Fall Edition 1985

RE RUN

The Best Programs From *RUN*

*For the C-64 and C-128**

PLUS:

Several New Program Releases!

*In C-64 mode only

Introduction

Welcome to the ReRUN Fall Edition. In this edition, we give you a wide variety of programs, many of which appeared in *RUN*'s June, July and August issues. Three have never before been published. All programs will work on the C-64 and on the C-128 (in C-64 mode). Directory titles are capitalized below and are usually not the same as the article titles. You must, of course, use the directory titles when loading programs.

This edition of ReRUN offers you three powerful disk-utility programs. AUTO MENU simplifies disk usage by loading and running programs with just a single keystroke. This program appears first on your disk, and when it's run, it will display the entire ReRUN disk directory. From there, you simply press the letter corresponding to the program that you want to run.

Call the Doctor (DISK DOCTOR) offers you a remedy for just about all of your disk problems. It is a valuable tool for regular disk checkups and includes options to list a directory, restore a file, pack a directory, list used blocks, view or modify blocks, format a disk and send a disk command.

The third disk utility is 64 Disk Directory (64 DIRECTORY). This will be one of your most-used programs, whether you are a programmer or strictly an applications user. Just load and run 64 DIRECTORY. Whether or not you're in the middle of a running program, you can see a directory of any disk in your drive with just the press of a function key.

We have two programmer's aids for you. Byte-Size Compiler (MICRO COMPILER) is a disk-based Basic compiler that will convert a Basic program into machine language. It is easy to use and allows you to take advantage of the speed of machine language programs. To accompany this compiler, we are providing TEST COMPILER and COLORS DEMO for your use in testing the compiler.

A Dozen Will Do It (FUNCTION KEYS) offers programmers a way to gain access to 12 function keys instead of eight. It preprograms the function keys to set background, border and character colors, to list a program and to run a program. It will also load a disk directory if the C-64 Wedge is in place.

A Dozen Will Do It also offers

you CALCULATOR, a program demonstrating the various methods of accessing preprogrammed function keys. It also acts as a valuable personal calculator.

C-64 Big Letters gives you three separate application programs for designing colorful screens full of large characters. BIG LETTERS is an editing program that lets you design your own big-letter screen directly from your keyboard. TITLE MAKER is a videotape title-making program that allows you to enter your titles as ASCII numbers within data statements, thus making it easy to save and reuse your designs. Finally, BANNER MAKER lets you type in a self-repeating big-letter banner across the middle of your screen.

High Performance Turtle (TURTLE GRAPHICS) adds ten new commands to your computer for using high-resolution turtle graphics. It is a Basic loader that installs the following graphics commands: Reset, Hires, Text, Colr, Left, Right, Move, Tailup, Taildown and Plot. TURTLE DEMO is an accompanying demonstration program that illustrates some of the amazing graphics designs you can achieve using these commands.

We haven't forgotten education. TEST MAKER lets you create, modify and save files of test questions and answers. Once a question file has been created, TEST TAKER will load your test-question file and give you the option of either testing your students or drilling them with random selections from the test file.

Lastly, we have two brand-new games. HOME RUN DERBY places you at a night game with baseball fans jamming the stands. You can hit and pitch, with the goal of hitting as many home runs as possible.

In GOLD GRABBER, you control a daring adventurer, Flynn, in his quest to catch bags of gold and hide them in his hilltop retreat. Watch out for the hungry eagle and falling ten-ton weights.

There you have it for another action-packed edition of ReRUN. We are looking forward to bringing you the PRODUCTIVITY PAK later this fall. It will be our second ReRUN Special Edition, and will be filled with productive applications for your C-64 and C-128 (in C-64 mode). We also have great plans for the ReRUN Winter Edition, which will contain the best programs from *RUN*'s September, October and November issues.

MARGARET MORABITO
Technical Manager
RUN *Magazine*

How to Load

DISK:
To load any of the programs, type:

```
LOAD "program name",8
```

then press the RETURN key.
The disk drive should whir while the screen prints SEARCHING FOR (program name). The screen should then print LOADING and then finally READY, with the flashing cursor beneath. Type RUN and press the RETURN key. The program will then begin.

CASSETTE:
Insert the cassette tape into the Datassette recorder, with the proper side facing up. Make sure that the tape is rewound all the way to the beginning. Type:

```
LOAD "program name"
```

then press the RETURN key. The screen will display PRESS PLAY ON TAPE. You should then push the play button on your Datassette recorder.

When the program has been found, the screen will display FOUND (program name). On some Commodore computers, you may then have to press the C= (Commodore symbol) key to load the program. On other Commodore machines, the program will load automatically. Check your owner's manual for specific loading procedures.
When the program has finished loading, you will see the READY prompt and the flashing cursor beneath. Type RUN and press the RETURN key to start the program.

NOTES:
You should use the entire program name exactly as listed to avoid loading programs that have similar titles.
Before loading a program, *always* refer to the article in the booklet for special instructions.
Be sure your C-128 is in C-64 mode before attempting to load these programs.

Directory

* You must load and run TURTLE GRAPHICS before you can load and run TURTLE DEMO.

+ You must create a questions file using TEST MAKER before you can use TEST TAKER.

Read the Introduction before trying to load any programs.

Auto Menu

Simplify disk operations by loading programs with a single keystroke.

By Joe W. Rocke

RUN It Right

C-64; C-128 (in C-64 mode)
Disk drive

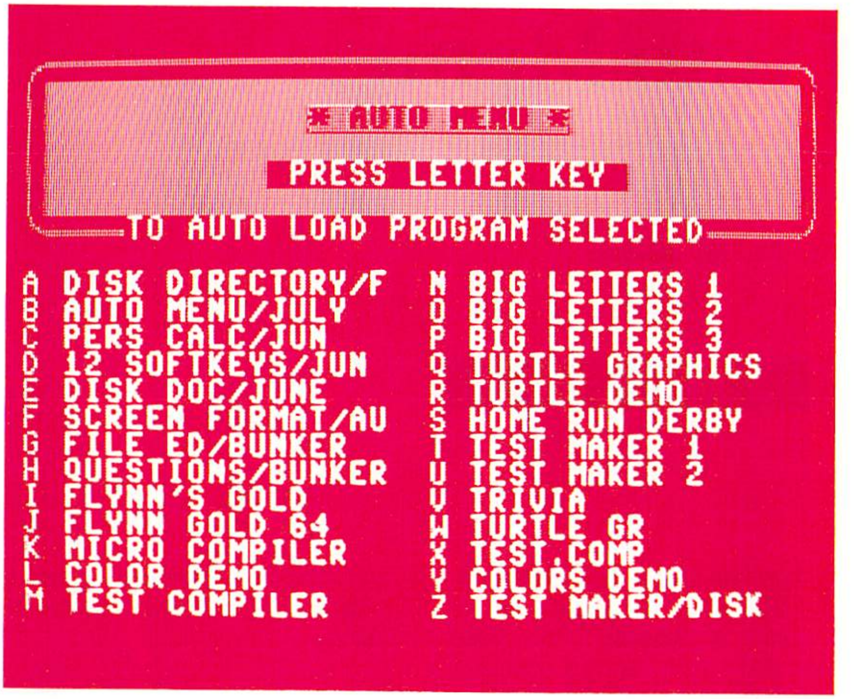
Auto Menu is a disk-based utility that takes the hassle out of loading a program. The Load and Run commands become a menu-driven operation. With a single keystroke, you select the program to be loaded. Your C-64 then takes over, automatically loading and running the selection. Even the most inexperienced newcomer can load a program.

The C-64 wedge and similar utilities provide shortcuts in typing the Load command. However, it's still up to you to remember and correctly type the program name. Everyone who uses a disk system has been confronted with a File Not Found message because of a typing error. Most of us have to load and read the disk directory unless a crib sheet of program names is handy.

SIMPLIFIED OPERATION

Auto Menu lists the disk directory in menu format. The disk directory is read automatically and listed in double-column form, which prevents all but the longest listing from scrolling off the screen.

Each filename is preceded by a letter that is assigned by Auto Menu. The letter serves as an identifier for program selection, and filenames are listed in the order the programs are stored on the disk. To load a program, press its filename's corresponding letter. This will also clear the



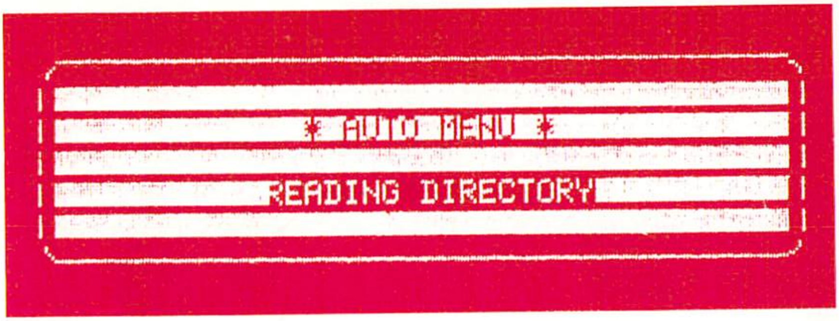
```
* AUTO MENU *

PRESS LETTER KEY

TO AUTO LOAD PROGRAM SELECTED

A DISK DIRECTORY/F      N BIG LETTERS 1
B AUTO MENU/JULY        O BIG LETTERS 2
C PERS CALC/JUN         P BIG LETTERS 3
D 12 SOFTKEYS/JUN       Q TURTLE GRAPHICS
E DISK DOC/JUNE         R TURTLE DEMO
F SCREEN FORMAT/AU      S HOME RUN DERBY
G FILE ED/BUNKER        T TEST MAKER 1
H QUESTIONS/BUNKER      U TEST MAKER 2
I FLYNN'S GOLD          V TRIVIA
J FLYNN GOLD 64         W TURTLE GR
K MICRO COMPILER        X TEST.COMP
L COLOR DEMO            Y COLORS DEMO
M TEST COMPILER         Z TEST MAKER/DISK
```

screen and display a loading message. A typical message display is as follows:

```
LOAD "PROGRAM NAM*",8,1
SEARCHING FOR PROGRAM
NAM*
LOADING
```

No further keyboard input is necessary after the initial menu selection. The loading operation takes place automatically, and, upon its completion, the computer automatically initiates a Run command. Finally, the menu program is removed from memory with a New command.

To facilitate the auto-run operation, the asterisk (*) pattern-matching format is used in Auto Menu's loading instruction, as described in the 1541 user's guide. Pattern matching simply means that the drive will load the first program that has a name matching the letters in the Load instruction. In this program, the


```
* AUTO MENU *

READING DIRECTORY
```

first 12 letters of program names are used. As it's unlikely that a disk will have two programs with the same name, this pattern-matching format should not pose a problem.

Save Auto Menu on each of your disks. When you want to use it, load and run it. The automated operation sure beats typing in loading commands! R

Call the Doctor

This disk utility has a remedy for just about all your disk problems and is a valuable tool for regular disk checkups.

By John Tanzini

RUN It Right

C-64; C-128 (in C-64 mode); disk drive

Oops! You have just scratched, or erased, an important program from your disk. Now, what are you going to do? Before you leap out that window, call the doctor—Disk Doctor, that is.

Disk Doctor will restore your senses by restoring those programs or files you've scratched. And that's not all. Disk Doctor is a disk-utility program with many useful functions.

PROGRAM RESTORATION

When you run Disk Doctor, the following menu will appear on the screen.

```
select a number
1. list directory
2. restore a file
3. pack directory
4. list used blocks
5. view or modify block
6. format disk
7. send disk command
press E to end
```

To select an option from the menu, simply press its corresponding number key.

Most of you will be interested in the ability to restore programs that have been scratched. When a program is scratched, nothing is actually erased. The directory slot is merely reserved until a new program is saved. At that time, the new program replaces the scratched program in the directory.

If many of your programs are scratched at one time, it will be a while before they are all written over by new programs. Sometimes a very old program can be recovered.

The first selection in the menu is provided to aid you in restoring programs. The listing will appear different from the usual directory listing. All items in the directory will be displayed, including scratched files, which will be displayed in reverse video.

If a file no longer remains in the directory, it will be impossible to restore. If a file still appears (in reverse video) when the directory is listed, you may attempt to bring it back.

Even though a program name remains in the directory, the program itself may have been overwritten. There is, however, no harm in trying to restore a program. You will be informed whether or not the attempt was successful.

As the directory is listed, items will scroll off the top of the screen. To freeze the screen, press any key. To resume the listing, press any key again.

Press the E key if you wish to end the listing, and the menu will reappear on the screen. This technique for ending an option or freezing the screen is the same in other parts of this program where items scroll off the screen.

When you want to try restoring a program, select number 2 in the menu. You will be asked for the filename. After you enter it and press the return key, a new menu will appear:

Is the file a
1. program
2. sequential file

Program files and sequential files are the only types of files that can be successfully restored. Make the appropriate selection and wait for the program to do its work. When it is finished, you will be given one of three possible messages: Cannot Restore, Successful Restore or Partial Restore.

It is possible that part of the file is recoverable while the rest of the program has been written over. Even when the Successful Restore message appears, you should verify that the file is the one you expected. Though very rare, it is possible that a new file has overwritten the old one and then been scratched. Disk Doctor has no way of knowing that it has latched onto the wrong file.

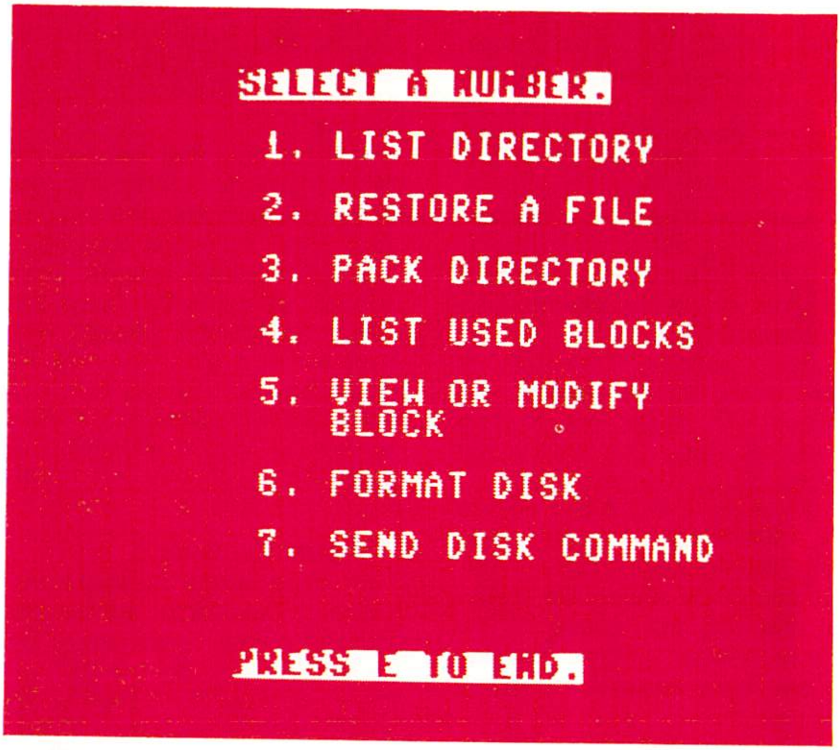
MODIFY A BLOCK

A very powerful capability provided by Disk Doctor is the ability to modify data stored anywhere on the disk. That's how this program is able to restore a file that's been scratched. The appropriate bytes in the disk directory are modified to indicate that the program still exists.

Of course, if you wish to modify the directory, you must understand the format and conventions used by the disk operating system. If you do not, you risk corrupting the entire disk.

Don't be afraid to fool around with a practice disk, since you cannot physically damage the disk in any way. Plenty of directories were rendered unreadable while debugging Disk Doctor.

You can use Disk Doctor to



```
SELECT A NUMBER.
  1. LIST DIRECTORY
  2. RESTORE A FILE
  3. PACK DIRECTORY
  4. LIST USED BLOCKS
  5. VIEW OR MODIFY
     BLOCK
  6. FORMAT DISK
  7. SEND DISK COMMAND

PRESS E TO END.
```

help you learn about the format of the directory. It provides you with a way to examine the data contained on any block. Try selecting item number 5 (view or modify block) from the main menu, and you will be presented with a new menu:

1. read block
2. view block
3. modify block
4. write block back

press E to exit

Let's read a block from the directory. Select item number 1 (read block). When the screen prompts you for the track and sector, type in 18,1 and press the return key.

The disk directory is stored entirely on track 18. Sector 0 is the first sector in the directory; however, sector 1 is the first sector containing filenames.

A few seconds after reading in a block, the menu shown above will reappear on the screen. Select item number 2 to view the contents of the block.

Since the screen is not large enough to display the entire block at once, only the first 64 of the 256 bytes in a block will appear. The bytes are numbered from 0 to 255. This number is shown in the left-hand column.

The next four columns are data from the block, shown in decimal. The four columns after that are the ASCII characters represented by the data.

If you have any programs stored on the disk, you will see their names on the screen. To view the next 64 elements, simply press any key. As usual, pressing E will end the display and return you to the menu from which you came.

If you feel daring, try changing one of the letters in a program name stored on this block. View the block again and note the location of the letter you want to change. Return to the menu and select item number 3.

You will be prompted for the location of the byte and the data you would like to put into that location. The location is the number from 0 to 255, shown in the first column when the block is viewed; it is not the track or sector. You are always modifying the track and sector that you last read in.

For example, enter 5 as the location and 65 as the data. The 5 is the location of the first letter of the first filename in the directory, and 65 is the ASCII code for the letter A.

You'll continue to be prompted for another location and data, so that you may change as many bytes as you wish. To return to the menu, simply press the return key without entering any numbers.

If you view the block again, the first letter of the first filename will be A. But you haven't yet changed the disk. You may view and modify the block as many times as you like until you are sure that you have it just the way you want it. When you are ready to update the disk, select option number 4.

When you attempt to write a block back to the disk, you will be asked for the track and sector of the block. Notice, though, that the track and sector of the last block read in already appear under the cursor. If you want to write back to the same block, then simply press the return key. If, instead, you wish to write back to a different block, overstrike the old track and sector with the new numbers and press the return key.

You will be asked one final question before the block is written back to the disk—"Are you sure? (Y/N)." This gives you a way out if you selected option number 4 by mistake. Pressing Y will result in permanently changing the disk. Any other key will return you to the menu.

AND THERE'S MORE

You have probably noticed that each program you save on a new disk appears at the end of the directory. After you scratch a number of programs, however, successive program entries are no longer placed at the end. Rather, they replace scratched files, thus acquiring random locations. This makes it difficult to find the latest version of a program.

Selecting item 3 in the main menu will compress the directory, filling in any holes left by scratched programs. From then on, programs will be added to the end of the directory. (This, however, lessens the chances of being able to restore a scratched program.)

Selecting option number 4 in the main menu will list the track and sector of all used blocks on the disk. It does this by examining the block allocation map (BAM) stored on track 18, sector 0.

As files are stored on a disk, the disk operating system keeps track of which blocks it uses by marking those blocks in the BAM. One simple way of copy-protecting a disk is to store some information on a specific block and then to alter the BAM by freeing the block. Since the block appears to be unused, most disk-copying programs will not copy that block.

If you would like to experiment with this, try sending a Block Allocate command or a Block Free command. Option number 7 in the main menu is provided as a convenience, so that you can send disk commands without having to exit the program. You need only type in the actual command. All necessary files are opened and closed for you.

For example, to send a Block Allocate command that will allocate track 1, sector 2, first select the Send Disk command option. The proper format for the Block Allocate command is:

```
B-A:0,1,2
```

The format for the corresponding Block Free command is:

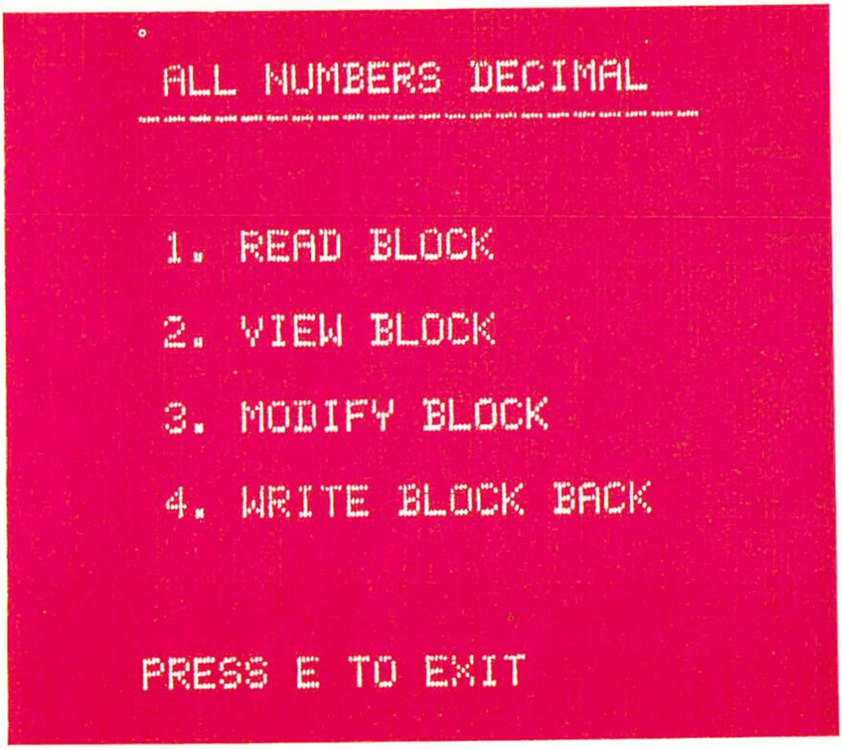
```
B-F:0,1,2
```

Files can be scratched using the Send Disk command option. To scratch a program named Disk Doctor, you would send the following command:

```
S:DISK DOCTOR
```

Any valid disk command can be sent in this way, but remember that Save and Load are not disk commands, but Basic commands.

Option number 6 is a simple subroutine, again provided for



```
ALL NUMBERS DECIMAL
------------------------------


1. READ BLOCK

2. VIEW BLOCK

3. MODIFY BLOCK

4. WRITE BLOCK BACK


PRESS E TO EXIT
```

convenience. It is used to format a disk. Since the New command is used infrequently, most people forget the exact syntax and must consult the disk manual. Disk Doctor will send the command for you, merely asking you for the disk name and ID.

FINAL WORDS

Disk Doctor is a fairly easy program to use. It may seem as though there's a lot to remember, but the program prompts you for any information that it needs.

Usually, when an option is selected, the program will open a disk file, then close the file when you return to the main menu. Since you should not change disks without closing all the files, I suggest you only change disks when in the main menu.

Extensive error checking is performed by reading the disk-error channel each time a disk operation is performed. Disk Doctor automatically prints out a message informing you of the error, should one occur. Reading the error channel clears the error so that you may proceed without exiting the program.

I am certain that you will find Disk Doctor a valuable addition to your utility programs. Even if you never need any of these functions to doctor your disks, you can use this program to learn how data is kept track of and stored on the disk.

The Key to Your Disk Directory

With this handy utility, accessing your disk directory is only a function key away, even while your program is running.

By Robin Franzel

RUN It Right

C-64; C-128 (in C-64 mode); disk drive

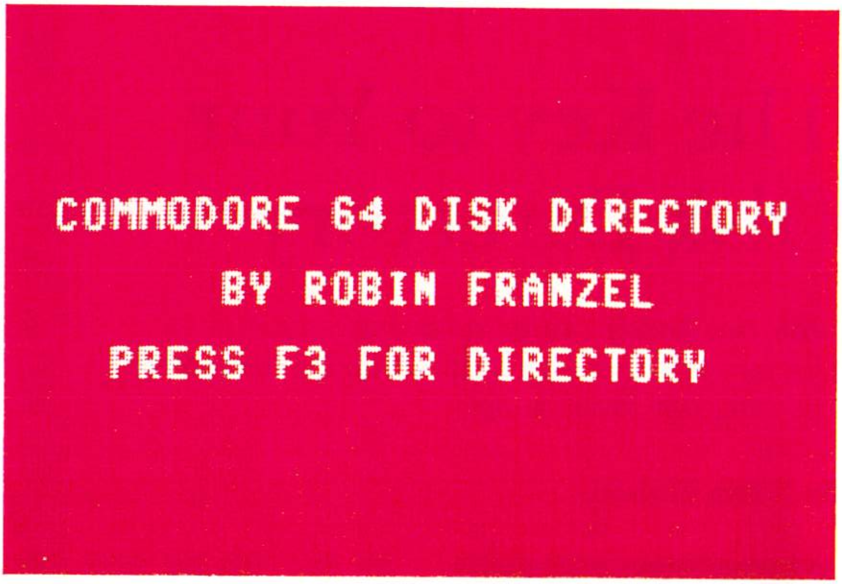
Have you ever been running a database program and forgotten a filename when prompted for it? You probably had to perform a number of inconvenient procedures (such as interrupting your program to enter "@$") in order to determine the filename. The accompanying machine language Disk Directory program enables you to display the disk directory simply by pressing a function key, even while your program is running.

When you press the f3 key, the directory will appear on the screen in two columns. To return to your previous display, press any key. This will work even when you are executing a Basic or machine language program.

Pressing the run/stop and restore keys will disable the directory utility; to re-enable it, simply enter SYS 49408.

If you want to list the directory using a key other than f3, just Poke in the appropriate key code for location 197. For example, to make the f7 key list the directory, enter POKE 49444,3.

I wanted a program that would use little memory, execute quickly and reside in memory with both the DOS wedge and the Screen Dump utility. (See "Print Your Screen," *RUN*, December 1984.) This program meets all those standards. It works with any program except those that use the



```
COMMODORE 64 DISK DIRECTORY
     BY ROBIN FRANZEL
  PRESS F3 FOR DIRECTORY
```

same memory space, which is $C100 to $C3DA (49408 to 50138).

It will even work with machine language monitors resident! (No more Formula-too-complex errors.) The program does not require any dedicated memory area outside the boundaries of the program itself, because all zero-page locations are saved and then restored at the end of the directory listing.

I keep the Disk Directory program in my computer just about all the time it's turned on! The more you use this program, the more you'll wonder how you ever managed without it. Have fun! R

C-64 Big Letters

Here are three utility programs that let you type onto your screen 48 colorful letters, numbers and other symbols, four times their normal size.

By Jimmie W. Bernard

RUN It Right

C-64; C-128 (in C-64 mode)

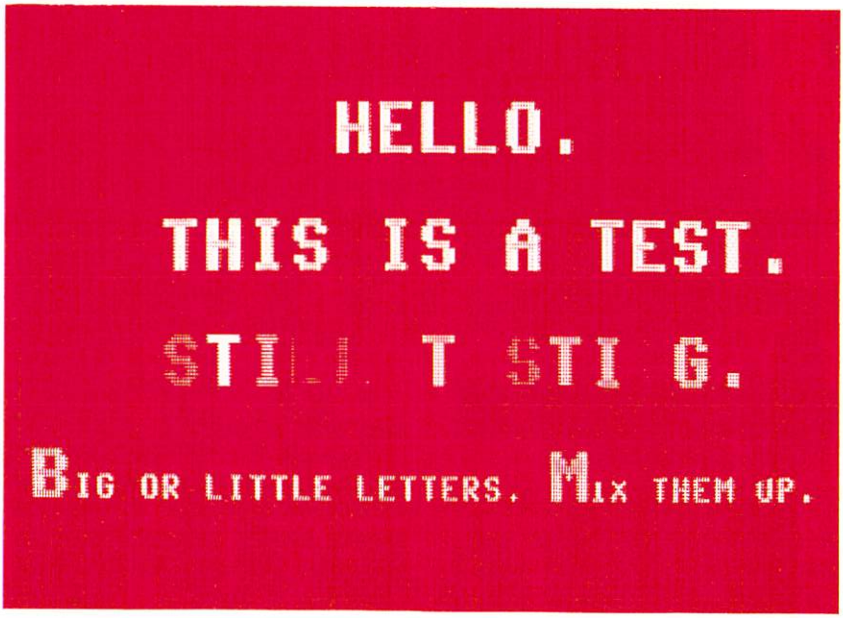
Big Letters lets you design your own big-letter screen right from the keyboard, with nearly complete editor functions and the capability to mix standard-size characters with big ones. If you have a VCR, this is an excellent way to create your own titles one page at a time.

Using a machine language subroutine to expand the normal character data, the program actually composes each giant letter of four reconfigured standard characters Poked together. In Big Letters and Title Maker, you may use the first 64 standard characters along with 48 big characters—A through Z, numbers 1–9, plus # $ % & ' + – , . ? * and the heart character. You can use the letter O to represent a 0, and the remaining character is the space, which is needed to print a blank or to overprint or erase an error.

Most of the instructions can be found on the title page. There are 12 screen lines in the program, with a possible 20 characters on each line. Unfortunately, there is no cursor, so you will have to pay attention. But the home/clear key, the cursor keys and the space bar all operate normally, with one exception: cursor up or cursor down will get you to the beginning of the line you move to.

The delete/insert key does not function, and the cursor-left key will only move you left to the beginning of the line you are on. Print, border and background colors can be changed instantly by tapping the left-arrow, up-arrow or pound keys, respectively.

Title Maker is similar to the first program, except, rather than being operated from the keyboard, it is completely preprogrammed through data entries


HELLO.

THIS IS A TEST.

STI   T STI G.

BIG OR LITTLE LETTERS. MIX THEM UP.

derived from the ASCII appendix, including the editor and color-changing functions.

For example, the program contains a test message that uses big letters, small letters, editor controls and color changes. Since the program scrolls, there is virtually no limit to the length of the titles you can make.

If you look at the portion of the listing beginning with data line 710, you can see what will happen by looking at the data in terms of ASCII. The 17s are cursor downs, so the video title presentation begins with a scroll from the bottom of the screen; the 32s are spaces to center text on a line. Then comes the text ASCII: 61 is a built-in, one-and-a-half-second timer delay you can use anytime and as often as you wish; 92, 94 and 95 are the color changers—the color is bumped up by 1 each time.

In line 760, the number 200 is a code by which the program does a Gosub to line 1000, which adjusts the actual cursor to the desired location and then prints a message in regular small letters. You will have to work at getting the small letters exactly where you want them.

Finally, use code 255 as the

last data entry and there will be no Out-of-Data message. Just hit the shift key when you have finished videotaping your creation.

The third program, Banner Maker, is a little different. You can use all 64 characters, up to 3200 of them, to create a self-repeating big-letter banner right in the middle of the screen. As the program begins, you enter your message in regular two-line segments, up to 40 inputs. You can't insert colons or commas into your sentences.

Then, when you press the shift key, your message will stream from right to left at a very readable speed, in big orange letters against a black background. (Other colors produce flaws or flashes in the scrolling chip!)

You could use the program to leave an attention-getting message for someone else right on your monitor or TV screen. You could also make a terrific window display for a business or social function. R

High-Performance Turtle

Slow and steady will win the race every time. Just type in this Basic program to add nine new commands for using high-resolution turtle graphics on your C-64 or C-128.

By Richard Holleran

RUN It Right

C-64; C-128 (in C-64 mode)

Turtle Graphics adds ten new graphics commands to Basic. You can use these commands in a program as easily as any other Basic command. Turtle Demo will give you an idea of how to use them in your own programs. The routines are written in machine language and do, indeed, execute quickly.

THE COMMANDS

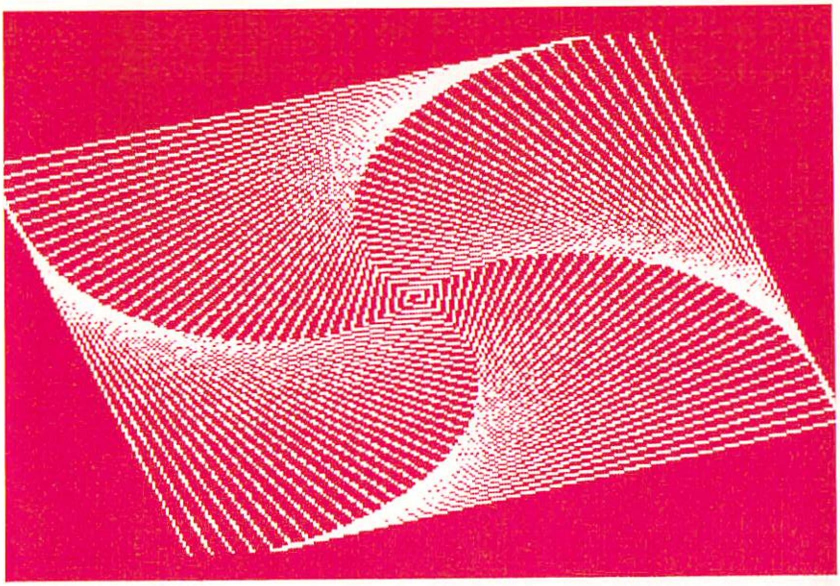
A short explanation of each new keyword follows. For fuller illustrations of their use, see the Turtle Demo program.

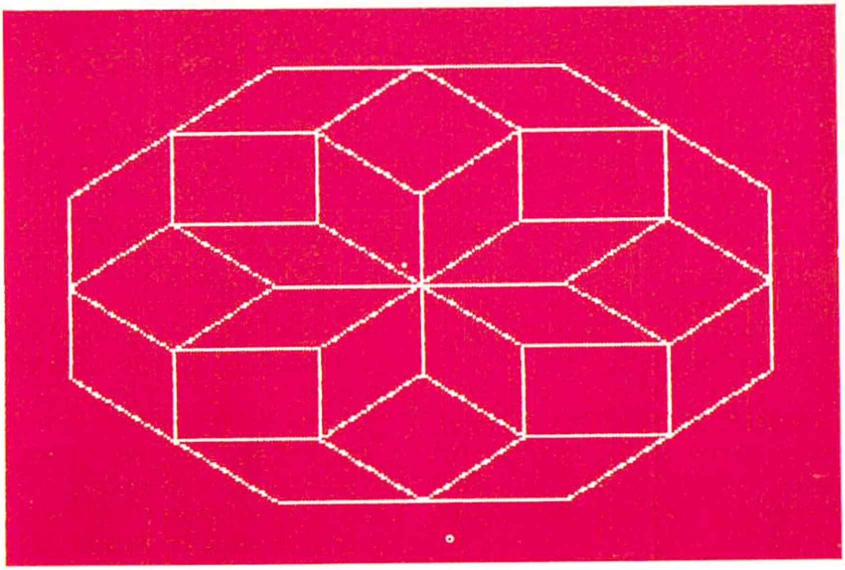
RESET is the initialization command; it centers the imaginary turtle on the hi-res screen and clears and turns on the hi-res screen.

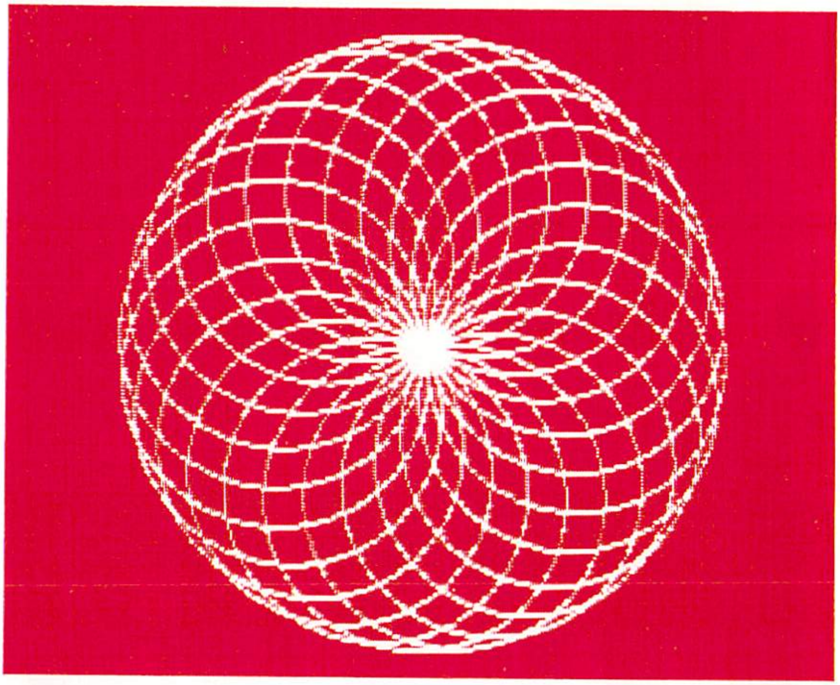
HIRES allows you to switch directly to the hi-res screen without clearing it or repositioning the turtle.

TEXT switches you from the hi-res to the text (normal) screen.

COLR allows you to select the hi-res colors. For example, COLR 0,6 sets the screen to black (0) and the turtle lines to blue (6). The numbers of the colors are the same as those found in the C-64 owner's manual. (It's unfortunate that this command cannot







be spelled COLOR, but the embedded OR keyword prevents it.)

LEFT X alters the turtle's heading in the counterclockwise direction. X is any number or numeric expression and is measured in degrees.

RIGHT X alters the turtle's heading in the clockwise direction. X is any number or numeric expression and is measured in degrees.

MOVE X is the command that propels the turtle, causing it to draw a line on the screen. X is any number or numeric expression. However, if X is negative, no move will take place.

TAILUP causes the turtle to lift its tail, allowing it to move without drawing.

TAILDOWN sets the turtle's tail down so that it will leave a line when it's moved. The tail is always set down by RESET.

PLOT plots any specified X,Y point on the hi-res screen. X values are limited to the range 0–319, Y values to 0–199.

If the turtle's travel takes it off the screen, no harm is done. The point or line is simply not Poked into memory. However, in the case of the PLOT command, a Y value greater than 255 will cause an Out Of Range error.

I hope you have fun with this program. If at any point you find the computer rejecting the turtle commands, POKE 1,54 should correct the problem. R

A Dozen Will Do It

The C-64's eight function keys can become 12 by means of some programming magic.

By Ronald Greenberg

RUN It Right

C-64; C-128 (in C-64 mode)

With a little creative programming, you can turn your C-64's eight function keys into 12. This article shows you how to find those four extra function keys and how to apply them for Basic and machine language programming.

Through the use of the hardware interrupt and a machine language subroutine, these 12 keys can function in the background as part of the operating system even while a Basic program is running.

The Commodore function keys are typically used in Basic in response to a statement such as:

GET A$: IF A$ = "F1" THEN action

The "action" can be a Goto command, a Print statement, a mathematical expression or other command. The expression above can also be written using a numerical representation for a character rather than a character name. For example:

GET A$: IF A$ = CHR$(133) THEN action

The expression CHR$(133) is the computer's numerical representation of f1. The CHR$ values of all eight function keys are given on page 136 of the *Commodore 64 User's Guide.*

FOUR MORE FUNCTION KEYS

The eight C-64 function keys are actually four physical keys, shifted or unshifted. For example, shifting the f1 key gives f2, which has a different CHR$ value. To gain access to four extra function keys, you use the Commodore key as a second type of shift key. However, this cannot be done in a straightforward manner, since shift f1 and

Commodore f1 both give the same CHR$ value.

To differentiate between shift f1 and Commodore f1 within a program, you must examine how the computer handles input data from the keyboard. Whenever a key is pressed, memory location 197 ($00C5) registers a key number. This number is different from the CHR$ value, as it represents a keyboard key rather than a character or function.

The computer generates a CHR$ value by coupling the key number with the value of memory location 653 ($028D), which is the shift flag. This location shows a value of 0 if no shift-type keys are pressed, 1 if the shift key is pressed, 2 if the Commodore key is pressed and 4 if the control key is pressed.

While the computer cannot generate different CHR$ values for shift f1 and Commodore f1, you can differentiate between the two by examining memory locations 197 and 653.

The logic that expands the number of function keys then becomes:

```
GET A$
IF PEEK(197)=4 AND
  PEEK(653)=0 THEN action 1
IF PEEK(197)=4 AND
  PEEK(653)=1 THEN action 2
IF PEEK(197)=4 AND
  PEEK(653)=2 THEN action 3
```

In this example, hitting f1 will result in action 1, hitting shift f1 (f2) will result in action 2 and hitting Commodore f1 (f1′) produces action 3. This logic can extend to the other function keys if you use values of 5, 6 and 3 in memory location 197 for f3, f5 and f7, respectively.

This results in a total of 12 user-defined function keys. One interesting point is that, in this type of logic, the control key cannot be used to try and pick up an additional four function keys.

FUNCTION KEYS AT WORK

Function Keys demonstrates the various methods of accessing function keys in Basic. In this program, the function keys are used twice—first for setting up the screen colors and then for providing the choices of mathematical calculations.

Additionally, two techniques are used for highlighting the function key menus on the screen: Extended Color mode and reverse printing. Highlighting a menu makes for a neater-looking display as well as making the program easier to use.

Calculator is a Basic loader for a machine language program that utilizes the function keys in the background. This program defines eight function keys, while leaving the four unshifted keys available for Basic programming.

The actions resulting from pressing the function keys are as follows:

Commodore f1 sets the screen colors to the Commodore blue.
Shift f1 sets the screen to black on grey, which is much easier to see than blue on blue.
Commodore f3 cycles through the background colors without having to Poke any values into memory.
Shift f3 cycles through border colors.
Commodore f5 cycles through cursor colors.
Shift f5 loads the disk directory if the wedge is being used.
Commodore f7 lists your present Basic program.
Shift f7 runs your Basic program.

When you are cycling through the colors, you can continuously hold down the function key until the desired color is obtained. This program will keep running in the background until you hit the run/stop and restore keys. To resume this program, just type SYS 49152 (return). R

Test Maker

Here's a chance to make your own tests. You can quiz yourself or your friends on any topic, and the number of questions you ask is limited only by your computer's memory.

By Thomas Bunker

RUN It Right

C-64; C-128 (in C-64 mode)

This program works as a sort of sophisticated electronic flash card, with the computer showing your questions, repeating the ones you get wrong and keeping score.

The program has two sections—Test Maker, to assist you in writing a question-and-answer file, and Test Taker, which uses these files to test your knowledge of a subject.

After you have created and saved your file of test questions and answers, using Test Maker, load and run Test Taker. When it has loaded your file, the questions will be printed on the screen one at a time. To determine if the responses to the questions are correct or incorrect, the computer compares the answer input from the keyboard with the correct answer in the file.

Test Taker has two modes of operation: Test and Drill. In the Test mode, the questions are presented once, in the same order in which they were written, with the computer showing a running total of right or wrong responses and a summary of results upon completion of the test.

In the Drill mode, the questions are randomly selected from a pool that initially contains all the questions in the file. A correct answer removes a question from the pool while an incorrect answer leaves the question to reappear until it is answered correctly. You must answer all the questions correctly to finish the drill.

In either mode, if you answer incorrectly, the computer retrieves the correct answer from the file and prints it on the screen. The Drill mode allows you two attempts to answer a question each time it appears, but only the first attempt counts. Pressing the return key without entering anything will immediately print the correct answer.

Load and run Test Maker first, as you cannot check out the main program without first writing a question file. Since the file is empty, the menu will only offer two selections: Load and Write. Press W for the latter, and the computer will prompt you to enter question #1. (If you are not starting a new file, the number will be the next available position in the file.)

Enter your question and place a question mark at its end. Space it properly. Press the return key and enter what you want the computer to recognize as a correct answer. Be careful to use either upper- or lowercase, not both, unless you tell your student to use both.

When you are through entering questions, respond to the question prompt by typing in the word menu by itself (the computer interprets this as a command and will not enter it in the file). The file is no longer empty, so the menu should now show a list of four options: Write, Load, Edit and Save.

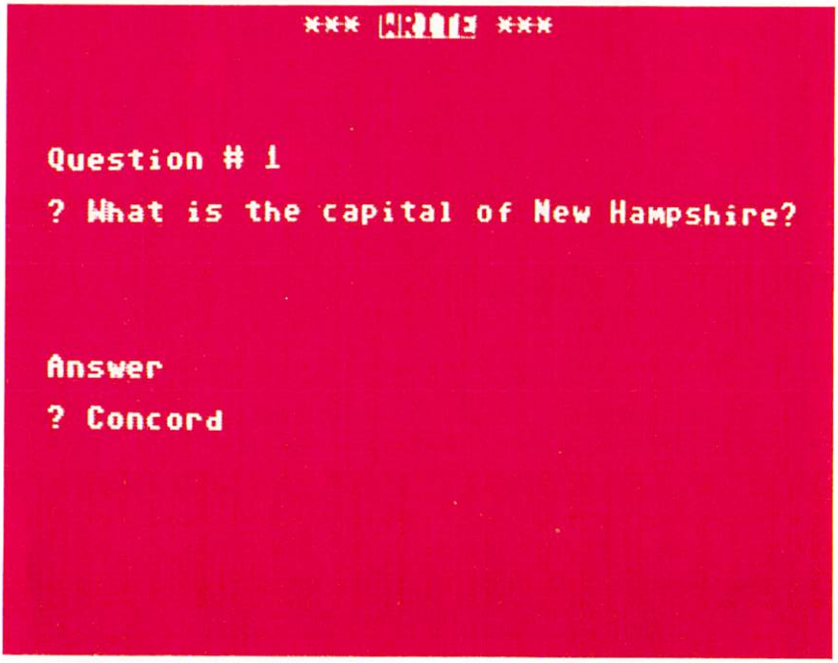
The Edit mode will print any question and answer set you select and offer four options: Next, which will print the next question in the file; Change, which lets you write a revised question that will replace the question shown on the screen; Drop, which will delete the question shown and shift all higher numbered questions in the file down to close the gap; and Menu, which will return you to the menu.

You can edit or save a file at any time and return to the Write mode without losing your place. The computer will always show the next available spot in the file whenever you enter the Write mode.

Tape operations are handled by selecting Load or Save from the menu. When directed by screen prompts, position the cassette tape to the place where you want to load or save. If there is a file in memory, the Load command will give you the option of either adding to or replacing this file.

This lets you combine a number of small files into one long file, but you must be careful, as this is one of the few places where the program cannot protect you from mistakes. If you do not have sufficient memory or if the total number of questions exceeds the size of the array, the program will crash. This is not a serious problem. Just make sure that you have saved the files separately before attempting to merge them.

It is very discouraging to spend an hour typing in a long



```
*** WRITE ***

Question # 1
? What is the capital of New Hampshire?


Answer
? Concord
```

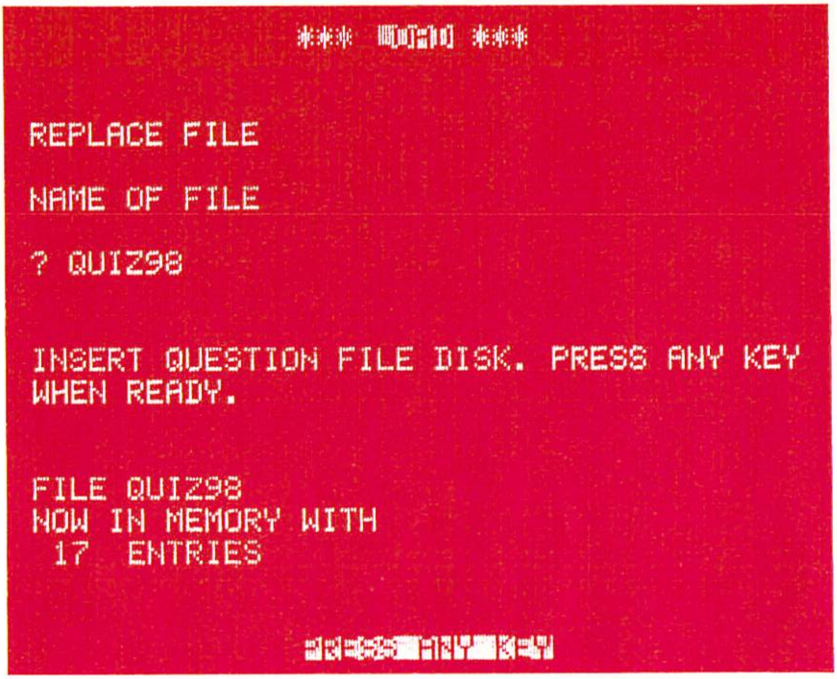
series of questions only to see an Out of Memory message and realize that all your work is gone forever. The program protects against this and other potential disasters, including many caused by operator errors, by checking the amount of memory remaining after every question is written to the file.

If less than 300 bytes remain, the actual number will be printed on the screen. This is a cautionary message only; you can continue entering questions and nothing will prevent you from going right off the end, so watch the memory remaining and save before you get too close to 0.

If the number of items in the file reaches the limit for which the array is dimensioned, a File Full message will appear. You can still edit and save, but you will not be able to enter the Write mode.

All the commands that can affect a file require your confirmation before execution. Pressing Y (for yes) will cause the operation to proceed, while pressing N (or any other key) will abort the operation.

Load and run Test Taker and enter a file tape (previously writ-



```
*** [illegible] ***

REPLACE FILE

NAME OF FILE

? QUIZ98


INSERT QUESTION FILE DISK. PRESS ANY KEY
WHEN READY.


FILE QUIZ98
NOW IN MEMORY WITH
 17  ENTRIES

[illegible]
```

ten and saved using the File Editor) as indicated by screen prompts. A menu will let you choose either type of test or load a new program. In either Test or Drill mode, the number of the question currently on the screen will be displayed in the upper left-hand corner. You can quit either test before finishing and jump back to the menu by entering the pound sign (£) in response to a question.

QUESTION GUIDELINES

The standard Commodore Input statement was used in the question file. This means that you cannot use commas, colons and cursor movements, but you may use periods, semicolons or quotation marks.

When you are ready to enter a question, the cursor will be flashing at the extreme left side of the screen; this allows you to enter four screen lines on the VIC or two screen lines on the 64. When you run Test Taker, questions will appear exactly as you type them into Test Maker so you must format your entry to avoid leaving part of a word at the end of a line. If you are in doubt, write a question and jump to the Edit mode to see what has actually been recorded in the file.

When writing your answers, remember that the computer requires an exact match to indicate a correct response. It will even make a distinction between a capital letter and a lowercase letter.

If the correct answer can only take one form—for example, if it's a year or a city—then that answer should be entered completely. However, if an answer could be stated in various ways, you might want to design a multiple-choice question, using a letter to indicate the proper answer.

It might be helpful to understand how the computer decides whether or not it has received a correct response. The number of letters in a file answer is the number of letters the computer looks at in the response.

For example, if you want the correct response to be California, and you are using lowercase letters, you might designate "ca" as an answer when writing the file. The computer would accept as correct ca, calif, california or any word in which the first two letters were "ca."

When you are entering your questions and answers in the Test Maker program, if you type the pound sign (£) immediately after the acceptable answer (don't leave a space), the computer will ignore the remainder of the entry when evaluating the response from the main program. You can also use the pound sign to further elaborate upon the answer or add a humorous remark that might be appropriate after a person has entered an incorrect response.

The number of questions you can write depends on the size of your computer's memory and the length of the questions. R

Home Run Derby

Koufax vs. Clemente, Seaver vs. Rose, Drysdale vs. Mays, Gossage vs. Rice. Experience the thrill and excitement of these classic pitcher/batter confrontations with this top-quality program.

By Mark Jordan

RUN It Right

C-64; C-128 (in C-64 mode)
One or two joysticks

Home Run Derby is a fun computer baseball game with two twists. One, instead of a regular baseball contest, it's just you against the pitcher, and—crack!—you're going for the fence. You can play against the computer or another person. The other twist is that the game is played at night.

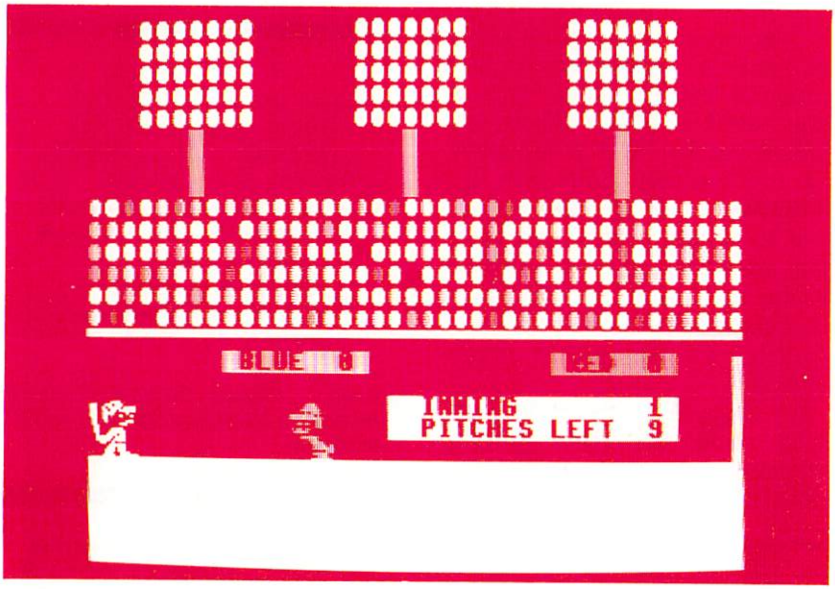
Once Home Run Derby loads, you'll be transported to a colorful, lighted stadium packed with enthusiastic fans. The lights are so bright you can see the ball's shadow follow the ball as it heads for the fence.

The concept is simple. In each of the nine innings, you and your opponent get a certain number of pitches; you try to hit as many balls over the fence as you can. If you play alone, you can hit for both sides. To swing, you simply press the fire-button (joystick in port 2). You have the option of allowing the computer to do all the pitching, or you and your opponent, depending on who's pitching, may control the type of pitch thrown.

If you opt for player-controlled pitching, you'll need to plug a second joystick into port 1. Depending on joystick positions, you have nine pitches available (see Table 1).

To release your pitch, just point the joystick in the proper direction and press the button. By mixing up the pitches, the defensive player can effectively prevent the batter from getting into a "groove." If the computer is controlling the pitching, it will do the mixing automatically.

Hitting the ball over the wall may seem hard at first, but be patient;



by the end of one nine-inning game, you'll have the knack—somewhat. The computer looks for perfect timing. If you swing when the ball is within nine pixels of your strike zone (dead center of batter's body), you'll hit it somewhere. Of course, the closer to dead center, the farther the ball will carry. Another factor in the ball's carry is the speed of the pitch; just as in real baseball, a blazing fastball is easier to put into the seats than a change-up. (That also ought to keep Goose Gossage-style pitchers from throwing nothing but heat.)

The risers and sinkers are dandy pitches because, if they are out of the strike zone, they will result in nothing better than a sharp foul tip. However, some don't sink or rise quite out of the strike zone, and these can easily be hit. Not only that, but if the batter lays off the ones that look too high or too low, an automatic umpire will call balls and strikes. If the pitch was a ball, then it doesn't count against the batter. So, if you can just lay off those forkballs, sliders, knucklers and rising fastballs, you won't lose any of your nine pitches. That, of course, is easier said than done.

The rest of the game is completely self-explanatory, even if you know nothing about baseball. A complete game takes about five minutes, unless you go extra innings.

The game makes use of many of the 64's excellent sound and graphics features, including two-part harmony, sound effects, animated sprites and a smoothly arcing batted ball with its shadow. I used an interrupt routine to create a stylized crowd that seems to be quite involved in the ball game. All the action is controlled via machine language, so the game plays smoothly.

So, enjoy a nice summer evening under the lights playing Home Run Derby. There's a long drive! It's going, going... Ⓡ

LEFT	straight fastball
LEFT/UP	rising fastball
LEFT/DOWN	sinking fastball
RIGHT	straight slowball
RIGHT/UP	rising slowball
RIGHT/DOWN	sinking slowball
UP	medium riser
DOWN	medium sinker
NO DIRECTION (fire-button)	medium straight ball

Table 1. Pitches available to you in Home Run Derby, according to joystick positions.

Gold Grabber

Bags of gold are dropping from the sky, and you've got to be quick to catch and bury them in this nine-level game of fun and strategy.

By Pasquale Longo

RUN It Right

C-64; C-128 (in C-64 mode)
Joystick

In Flynn's Gold, Flynn is a daring adventurer. He hides behind a hill watching bags of gold falling down from the sky. An eagle soars, swoops and glides overhead.

Flynn must run out, catch the gold, bring it back to the hill and bury it.

This is a dangerous task. Giant ten-ton weights rain down on Flynn every time he leaves the safety of the hill. If the eagle is hungry, he might attack him.

Guide Flynn with a joystick plugged into port 2. To catch a bag of gold, position Flynn under it. Bury the gold by guiding Flynn back to the hill and pressing the fire-button.

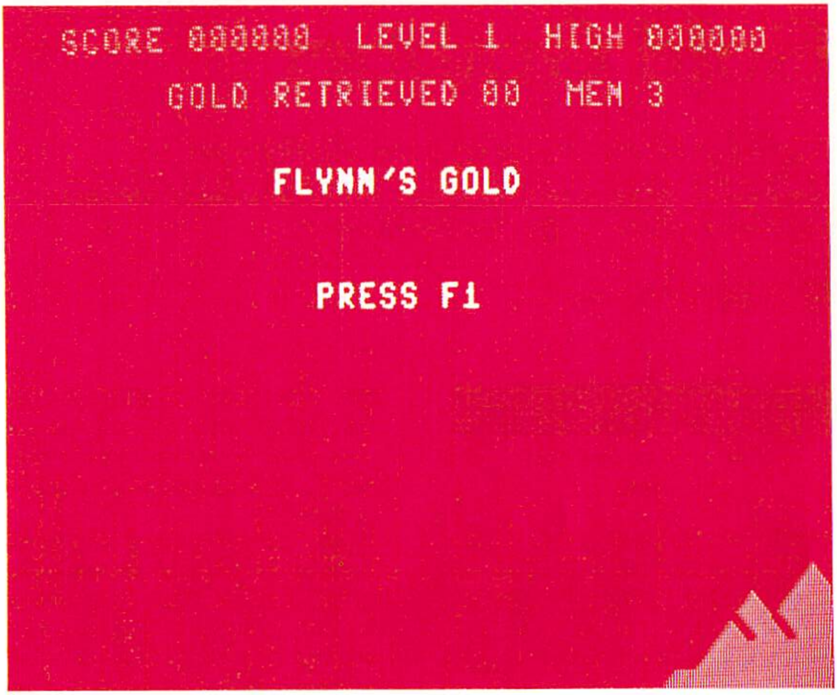
You receive points for each bag of gold caught. You also receive points when you bury the gold. The bags of gold vary in point value, with those greatest in value falling farthest to the left. Burying a bag gives you additional points.

Flynn can only carry and bury one bag of gold at a time—the last one caught. Therefore, when you are running to bury a more valuable bag of gold, you must be careful not to catch a bag of lesser value, as this bag will replace the one you were holding. You'll retain all the points you earned for having caught the more valuable bag, but because you lost the bag before burying it, you won't earn as many points upon burial.

The game ends when you have caught and buried 25 bags of gold.

To play the game, you're given three Flynns. The game ends when you lose all of them.

There are nine levels of play. You can select the level of difficulty at the beginning of a game


```
SCORE 000000  LEVEL 1  HIGH 000000
     GOLD RETRIEVED 00  MEN 3

           FLYNN'S GOLD


             PRESS F1
```

(when the game is waiting for you to press f1) by pressing f3. The higher the level, the more points you receive for catching a bag of gold and for burying it.

If you get off to a bad start, you can restart the game by pressing the stop key. To quit the game entirely, press the restore key.

The program takes about three minutes to load. 🅁

Byte-Size Compiler

Don't let your Basic programs slow you down. This short little compiler adds unbelievable execution speed to your programs, and it's easy to use.

By Victor H. Cortes

RUN It Right

C-64; C-128 (in C-64 mode)

Micro Compiler is an integer Basic compiler for the Commodore 64. A compiler converts a high-level language program, such as one written in Basic, into a machine language program. It allows you to take advantage of machine language speeds without knowing machine language.

Micro Compiler compiles a subset of the regular Commodore Basic, which I call Micro Basic, into machine language. Since it is a subset of the regular Commodore Basic, you can develop, test and save programs using the regular Basic interpreter.

To begin, load and run Micro Compiler. When the compiler is run, it first asks for the name of the Micro Basic program, or source program, to be compiled. Then it asks for the address at which to start placing the machine language, or object, code. If no address is entered, the compiler defaults to 49152. The source program is then read directly from disk and is listed line by line.

A special technique is used to convert the command tokens on disk to their expanded form and to print them. If any errors are encountered, a message is printed under the line in question. Since it is assumed that the source was tested with the regular Basic interpreter, a minimum of error checking is done. If an error is found, it is usually because of an invalid Micro Basic command. Once the program is compiled, you have the option of saving the code, executing the code, compiling another program or just terminating.

When saving the machine language code, the compiler will ask for the name of it. If no name is entered, the compiler defaults to the source name plus ".ML".

After the compiled code is saved, you can load it with a regular Basic program, using the statement: LOAD "program name",8,1. The code can be executed from Basic with the SYS command to the starting address of the code (usually SYS 49152). This can be done in Direct mode or in Program Run mode.

Since a full Basic compiler would be a very large program, this compiler was written to handle a subset of Basic. This subset has a number of limitations. No nesting of If or For statements is allowed, and string variables and variable arrays are not supported.

There are, however, many ways to get around these limitations. For example, instead of using a variable array to hold numeric values, you can Poke these numbers into an area in memory. You can accomplish essentially the same task by changing

```
FOR I=1TO5: A(I)=I: NEXT
```

to

```
FOR I=1TO5: POKE 828+I,I:
  NEXT
```

Usually, you'll just compile a Basic subroutine that needs speeding up. Then, instead of performing the subroutine with a Gosub statement, you can call the compiled routine with a SYS statement. You can also write and compile for execution a program written only with Micro Basic statements.

The following is a list of the Micro Basic commands the compiler can process:

```
{LET} V=Expr
PRINT {Expr} {CHR$
  (Expr)} {"string"} {;}
IF Expr Cmpr Expr THEN
  {statements or line#}
FOR V=Expr TO Expr
  {STEP Expr}
NEXT
POKE Expr,Expr
SYS Expr
GOTO N
GOSUB N
RETURN
END or STOP
REM {remarks}
```

{} indicate an optional item.

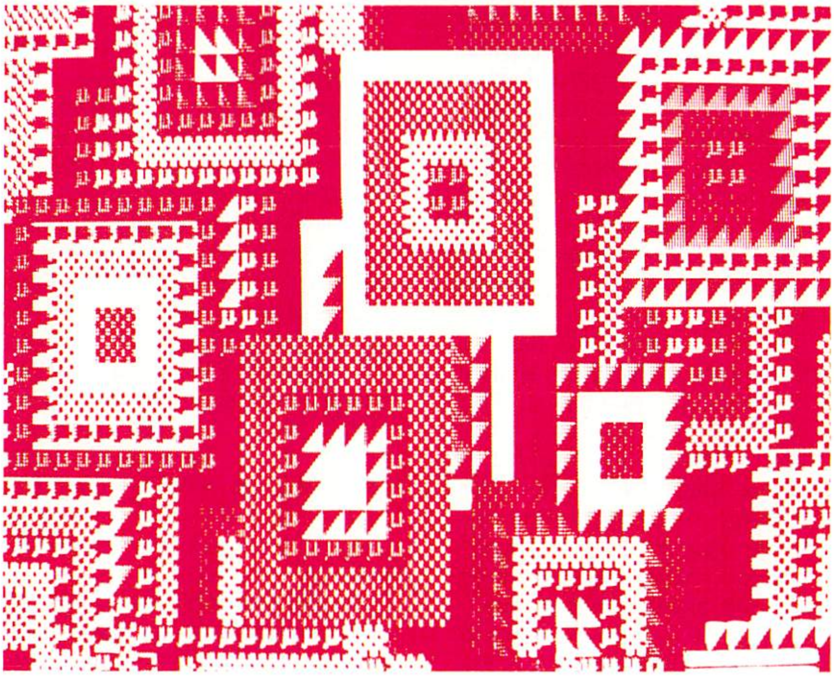
V is used to represent a variable name. The first character in the name must be unique (A–Z). These variables use memory locations 680 to 731.

N is used to represent a numeric integer literal. Its value can be from 0 to 65535.

X is used to represent a variable or a numeric integer (V or N).

Expr is a numeric expression beginning with X or PEEK(X) and optionally followed by any number of the following: {+ X}{− X}{* X}{/ X}{AND X}{OR X}.

Cmpr is a type of comparison



and can be one of the following: = (equals), > (greater than), < > (not equal to) or < (less than).

The following are examples of valid statements:

```
R = PEEK(A) * 100 / M
IF Y*40+X > 2023 THEN
  PRINT CHR$(147);
FOR I = 1 TO X + A :
  PRINT I + 64 : NEXT
SYS B+1024 : RETURN
GOSUB 500 : PRINT "TOTAL";T
GOTO 20
POKE A-I, J AND 15 : END
```

The following are examples of invalid statements:

```
R = COS(B)
PRINT TI$
GET X$(I)
OPEN 15,8,15
```

Like regular Basic, the "LET" is optional in the Let command. It is used to assign an expression to a variable. An expression must begin with a variable, a number or the Peek function of a variable or number. It can be followed by other variables or numbers, as long as they are separated by +, −, *, /, AND or OR.

The expression is evaluated from left to right, with no operator precedence and no parentheses. Because of this, you must place the multiplication and division operators first, the addition and subtraction operators next and the and/or operators last. This will ensure correct evaluation of an expression by the compiled program.

The Print command can be used to print a numeric variable, a PET ASCII character (CHR$) or a string. The optional semicolon, if used, will prevent a carriage return after the Print statement. The semicolon can also be used to print any combination of these. Only the Print command allows literal strings.

In the If...Then statement, Then can be followed by a line number or any other Micro Basic statement. Multiple statements can be put on one line as long as they're separated by colons. However, it's easier to correct errors if each command is on a separate line.

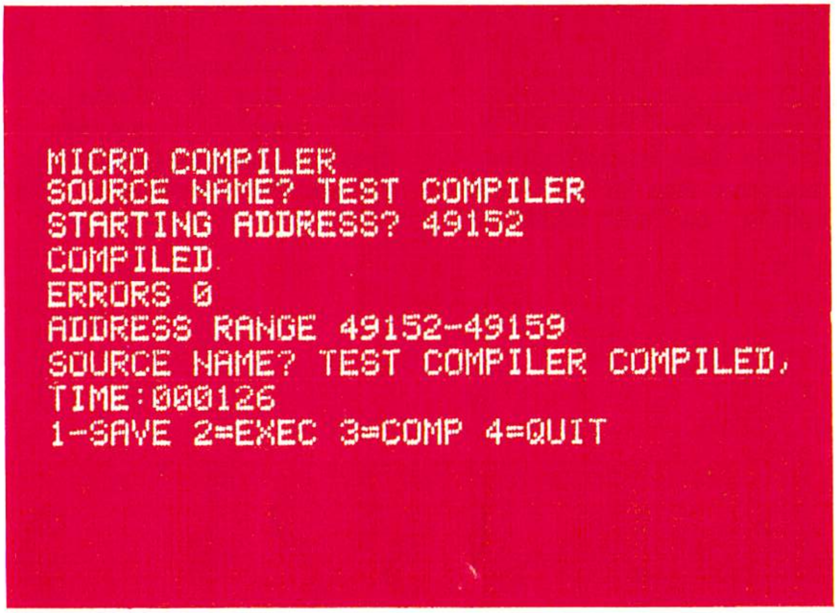
All the string-manipulation commands (LEFT$, MID$,...) are omitted because Micro Basic does not handle string variables. It can only handle numeric integer variables or literals in the range of 0 to 65535 (two bytes).

Some commands can be simulated; for instance, instead of the Get command, you can use PEEK(197) to read the keyboard. The value returned by the PEEK(197) can even be converted to its PET ASCII equivalent by using the internal ROM tables. Most of these restrictions were necessary to keep the compiler program to a reasonable size.

Test Compiler's main function is to test whether the compiler is working correctly. When you load and run it, the program first clears the screen and prints TEST.COMP. It then positions the cursor to the tenth line and prints TEST. Next, it prints numbers from 1 to 5. You should then press keys at random. The program reads the keyboard and prints the characters that were entered.

It then identifies the character as being equal to, greater or less than the character A. It also changes the screen border to green if the character entered is equal to an A, and to red if it is not. The program will terminate when the f7 key is pressed.

You can then load and run Micro Compiler. When prompted by the compiler for the source name, enter Test Compiler. Press the return key when prompted for the address to default to 49152. The test program will load and begin compiling. After the compilation is done, the compiler will display the address range (starting and ending addresses) required by the compiled code and also the number



```
MICRO COMPILER
SOURCE NAME? TEST COMPILER
STARTING ADDRESS? 49152
COMPILED
ERRORS 0
ADDRESS RANGE 49152-49159
SOURCE NAME? TEST COMPILER COMPILED,
TIME:000126
1-SAVE 2=EXEC 3=COMP 4=QUIT
```

of errors encountered. It will then display the following options:

1. Save—use to save the machine language code.
2. Execute—will execute the machine language code.
3. Compile—will allow you to compile another program.
4. Quit—will send you back to Basic.

Enter option 2 to execute the program. The results should be similar, except for the speed of the compiled version.

Colors Demo demonstrates the difference in speed between an uncompiled versus a compiled program. This program fills the screen with various color designs. First, type it in and run it, noting how long it takes to fill the screen with color patterns. Now load and run the compiler. When prompted for the source name, enter Colors Demo and enter a starting address of 49152. When the program has been compiled, enter option 4 to terminate the compiler.

Now enter SYS 49152 to execute the compiled machine language code. The screen should fill instantly, compared to the minute or more required for the

Basic version. This should convince you of one of the advantages of using a compiler program. The compiled versions of Basic programs are so fast that you will often have to insert a For...Next loop to hold a program to controllable speeds.

With this compiler, you'll be able to develop programs using a high-level language (a subset of Basic) that will give you, as a final product, a program in the machine's own language, helping you take advantage of the computer's full capabilities. 🅁

Notes

In This Edition . . .

►Games ►Utilities ►Education
►Graphics ►Applications

Exciting Programs
From RUN's June, July and August issues!

Including . . .

- ►Home Run Derby
- ►Big Letters
- ►Turtle Graphics
- ►Function Keys
- ►Test Maker
- ►Gold Grabber
- ►*And More!*

If any manufacturing defect becomes apparent, the defective cassette or disk will be replaced free of charge if returned by prepaid mail within 30 days of purchase. Send it, with a letter specifying the defect, to:

ReRUN • 80 Pine Street • Peterborough, NH 03458

Replacements will not be made if the cassette or disk has been altered, repaired or misused through negligence, or if it shows signs of excessive wear or is damaged by equipment.

These programs will not run under all system configurations. Use the RUN It Right information included with each article as your guide.

CW COMMUNICATIONS/PETERBOROUGH