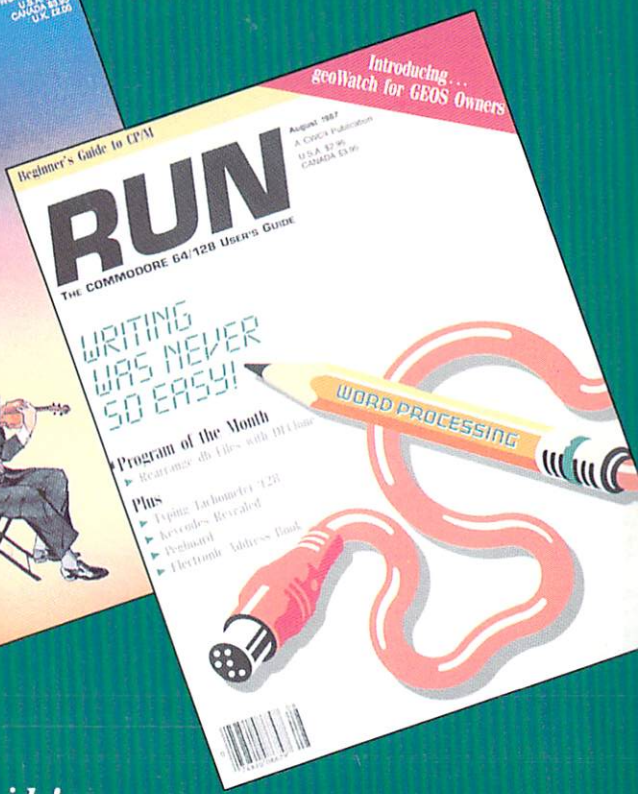
July/August 1987 Edition

RE RUN

RUN Programs on Disk

For the C-64 and C-128



Bonus Programs Inside!

Introduction

July-August '87 ReRUN

In much the same manner that the Lone Ranger and Tonto never seemed to lose the knack for finding excitement and mixing it up with bad guys, we here at ReRUN never seem to exhaust our programming resources. In fact, despite its having been on the market for over five years, the C-64 is still inspiring the creation of plenty of innovative programs, and the 1987 July-August ReRUN is proof of that statement. The C-64 and C-128 programs published in the July and August issues of *RUN* included some real show-stoppers.

I've received many positive comments concerning the programs published so far this year in ReRUN. Rest assured that I'll make every effort to continue to provide you, our ReRUN customers, with the best software products for your money.

Enough talk; let's get down to business. Foremost on my list of July programs is Tri-Solitaire. Now, I'm not particularly fond of computerized card games, so I had little interest in this one before I tried it. Surprisingly, it turned out to be so much fun that I found myself thinking of reasons to "test" it. And "test" it again. . . and again. I'll utter a few words of caution: Limit your playing time with Tri-Solitaire, or you also might spend long hours "testing" it.

Most grocery-shopping programs produced for the C-64 have proved to be ineffective. Using pre-defined prices and grocery items and confusing, complicated instructions, most were hardly worth using. Well, forget every bad experience you may have endured with computerized grocery lists—Attention, Shoppers! on this ReRUN disk revolutionizes the approach to computer-generated shopping lists. You input the groceries, their prices and location in the store and let Attention, Shoppers! itemize and produce a printed list.

Linker 128 eliminates the hassle of linking together long (or short) Basic 7.0 programs on your C-128. I've found Linker 128 to be a very handy utility.

Nearly everyone who has used

a computer for any length of time has been prompted by a program to enter data, such as field data within a database, only to have those nonsensical "extra ignored" or "redo from start" messages appear. Robert Senft's program, Input Sentry, solves that problem. It allows you to design programs that rigorously inspect input data and accept only the types of data you define.

Interaction with your elementary school-age children or grandchildren was never easier thanks to your Commodore, a joystick and our July Easy Applications program, Flash Cards. This educational program requires you to press the joystick whenever your child answers a math question that appears on the screen.

Wrapping up the programs from July is a Mega-Magic program that lets you create pie charts for the C-128, using Ultra Hi-Res. Those familiar with Ultra Hi-Res (available on the January-February 1986 and May-June 1986 ReRUN disks), will greatly appreciate the ability to create pie charts, using this program.

Moving on to the August issue, we proudly offer DFClone, for Datafile 3.6 users everywhere. It affords a way to alter the number of fields to be included in a file with Datafile 3.6. DFClone will prove to be a significant addition to the Datafile 3.6 lineup, because it opens up a new dimension in reducing, expanding and copying your present Datafile files. Anyone who has wanted to capture certain fields from one Datafile file to use in another one will find DFClone to be indispensable.

For C-128 programs, there's Typing Tachometer, an application program for 80-Column-mode only, which measures your typing speed in words per minute.

Electronic Address Book is the August Easy Applications program, designed to eliminate the need for those little black address books. The program lets your computer file away all those names and addresses on disk and to print them out when needed.

Next, there's Keycodes Revealed, which displays the value of every C-64 key on the screen whenever you want to reference those values. Run it, then enter a five-digit SYS number to activate it.

Pegboard 64 is next on the list of C-64 August programs. If you played Pegboard when you were a kid, you'll realize that this computerized version is every bit, if not more, challenging than the original.

The last of the August programs is that month's Mega-Magic, appropriately called Locator 64. It lets you search for a character, group of characters or

keyword, then scans and displays every line in your Basic program that contains those characters or that keyword. Think of the possibilities! Whether you use it as a debugging tool or just for fun, I'm sure you'll find it to be both indispensable and fascinating.

Finally, for a Bonus program, there's Sprite Database. Use it to create, edit and save to disk as many sprites as your imagination allows. There is virtually no limit to what you can do with this program and a little ingenuity.

That wraps it up for this edition of ReRUN. Keep those letters coming, and I'll be back in two months with my next exciting edition of ReRUN. Hi-ho, Silver, awaaay!

Tim Walsh
Technical Editor
RUN *magazine*

ReRUN Staff

Technical Editor: *Tim Walsh*
Managing Editor/Production: *Swain Pratt*
Copy Editor: *Peg LePage*
Proofreader: *Harold Bjornsen*
Design and Layout: *Anne Dillon*
Typesetting: *Doreen Means, Beth Krommes, Ken Sutcliffe*
Special Products Director: *Paul Finch*
Special Products Assistants: *Debbie Bourgault, Robyn Johnson*
Direct Marketing Coordinator: *Debbie Walsh*

Directory

* C-128 program
£ Bonus program

How To Load

Loading from Menu

To get started, C-64 users should type LOAD "MENU 64",8 and press the return key. When you get the Ready prompt, the menu is loaded and you should type RUN to see a list of the programs on your disk. C-128 users need only press the shift and run-stop keys. When all the programs are displayed on the screen, you can run the one you select by pressing a single key.

Loading from Keyboard

If you do not wish to use the menu program, follow these instructions:

C-64:

To load a C-64 program written in Basic, type:

LOAD "DISK FILENAME",8

and then press the return key. The drive will whir while the screen prints LOADING and then READY, with a flashing cursor beneath. Type RUN and press the return key. The program will then start running.

To load a C-64 program written in machine language (ML), type:

LOAD "DISK FILENAME",8,1

C-128:

All C-64 programs can be run on the C-128 as long as your computer is in C-64 mode.

All C-128-mode programs are clearly labeled on the directory page. Your C-128 *must* be in C-128 mode to run these programs.

To load a C-128-mode program, press the F2 key, type the disk filename and then press the return key. When the program has loaded, type RUN.

Making Copies of ReRUN Disks

Many of the programs on your ReRUN disk have routines that require you to have a separate disk onto which the program writes or saves subfiles. In order for you to use these programs, you will first have to make a copy of the original program onto another disk that has enough free space on it to hold these newly written subfiles.

If the program is written in Basic, it is simple to make a copy of the program. Just load the program into your computer following the procedures outlined above, and then save the program back onto a separate disk that has plenty of free space for extra files.

If the program is written in ML, copying is not so simple. You cannot simply load and save an ML program. In this case, you'll need to use a disk-backup utility program, such as the one on your Commodore Test Demo disk.

Tri-Solitaire

By Jim and Deborah Chambers

RUN It Right

C-64

Tri-Solitaire is a fairly simple card game that still can be challenging, even to solitaire aficionados. The program uses a regular 52-card deck with all the face cards removed, leaving 40 cards to play. Sixteen of the cards are dealt face up in a four-by-four pattern. (These are referred to as table cards.) The remaining 24 cards make up your hand.

The object is to play each card in your hand on a table card so that the total of the other three cards in the same row or column equals the value of your card or its value plus a multiple of ten. For example, if you play a seven, the total of the other three cards in that row or column must equal seven, 17 or 27. Also, you can't play your card on a table card of the same value. Card suits don't matter, and aces always count as one.

HOW TO PLAY

At the start of each game, the program automatically shuffles the cards, then displays the table cards in the four-by-four pattern. The first card in your hand and a Play To? prompt are displayed to the right of the table cards. To play the card, you enter a two-digit command consisting of a letter (A–D) and a number (1–4), such as B3, which represents the location of the table card you want to replace.

If you make a good play, the card in your hand replaces the designated table card, and your score, displayed in the upper-right corner of the screen, is increased by one. The next card in your hand is then displayed.

If you make an incorrect play, an Illegal Move message flashes in place of the Play To? prompt, and everything on the screen remains the same. You can either try to play the card again or examine the next card in your hand by entering a ← (the left arrow key at the upper-left of the key-

board). By repeatedly entering this key, you can flip through and examine as many of the cards in your hand as you like.

You can play the cards in any order. The game is over when you reach a score of 24 or exceed the eight-minute time limit. To adjust the limit, just change the value of LM (line 130) and the word "eight" in line 346.

Note that this program won't run with some ROM cartridges installed, notably the disk fast-load type. If you have a problem running the program, try disconnecting these cartridges. ■

Attention, Shoppers!

By Bob Kodadek

RUN It Right

C-64; C-128 (in 40- or 80-Column mode); printer

Doing the weekly food shopping at the supermarket can be a long and tedious task, but now my Shopping List program can make your trips to the store faster and more efficient. It will let you "browse" through the aisles before you even leave home, compiling a complete shopping list arranged according to where products are located in the store.

To use the program, you establish, in the Data statements beginning at line 1000, a basic list of all items you might want to buy during any shopping trip. Each time you run the program to make a specific list, all these items appear in a succession of screen displays, and you specify the ones you want to buy on that particular day. The program saves those items in a sequential file you can call up to start your next shopping list. You can alter the basic list in the Data statements during any Shopping List session.

The printout you take to the store lists all the products you intend to buy, along with the quantity, unit price and total price for each, the aisle number where it's located and the total you'll have to pay at the cash register. If you enter products in the order you come to them in the store, they'll be listed that way, thus saving you considerable time in trips back and forth across the supermarket.

COMPILING THE LIST

When you run Shopping List, a sample list will appear. I've included some preliminary Data statements in the proper format to show you how to set up your own. You'll have to make additions, deletions and alterations to

them to match your needs. After making any changes to the data, make sure you have a formatted disk in the drive.

In the first couple of weeks of using Shopping List, take notes on prices and aisle locations as you do your shopping so you can add more items to the Data statements. Thereafter, you'll also need to keep track of price and location changes and alter the Data statements accordingly.

ADDING DATA STATEMENTS

As you add Data statements, keep line 1000 as the starting point for items located in aisle 1 at the store, line 2000 as the starting point for items in aisle 2, and so on for all the aisles in the store. There are four data items necessary for each product, the first two being name and price. Although I've included some leading zeroes in the prices to make the display neater, neither they nor trailing zeroes are necessary. The third data item is the quantity of the product you want, and the fourth is the number of the aisle where the product is located.

As you're working with the Data statements, keep in mind that line 40000 must always remain in the program. It signifies the end of the data to be read.

MAKING A SHOPPING LIST

When you run Shopping List, first it will ask, "Have you made any additions, deletions, or corrections? (Y or N)." This refers to the Data statements. If you have altered the Data statements, type Y to save and replace the program listing with the new Data statements. If you haven't changed them, type N.

Next, the first screen display of items in the Data statements will appear. The product list occupies the top of the display, and across the middle of the screen you'll see a line that includes the program title, messages about program operation, the "page" number of that screen display and the aisle in the store where the first item on the page is located. Note that the page number refers to the screen display only, not the printout; I assumed most shopping lists would fit on only one printout page. At the bottom of the screen there's a menu of keypress commands to use in creating your list.

You'll notice that some of the products are displayed in white, some in green. Those in white list a quantity of at least one. When you load the sequential file holding your previous shopping list, any items in it that are in addition to the Data statement items will turn white also. All the

white items will appear in the printout, unless you decrease their quantity to zero (at which point they'll turn green).

The green items in the display have zero for a quantity and won't appear in the printout unless you change their amount to something other than zero. The maximum quantity for each product is nine.

The highlighted item in the display is where the cursor is resting.

LIST-MAKING COMMANDS

To *increase the quantity* of the highlighted product, press the *cursor-right* key. If the starting quantity is zero, increasing it will add the item to your shopping list.

To *decrease the quantity* of the highlighted product, press the *cursor-left* key. Decreasing the quantity to zero will eliminate it from your shopping list.

To *browse* through the list of items on the current screen page by moving the highlight down, press the *cursor-down* key. At the bottom of the page, the highlight will wrap around to the top of the list.

To move to the *next screen page*, press the *return* key.

To display the *previous screen page*, press the *cursor-up* key.

To *print out* your list, press *P*.

To *load* into memory the sequential file containing your previous shopping list, press *L*.

To *save with replace* your current shopping list into the sequential file, press *S*.

To *view* the sequential file in memory, press *V*. As the list scrolls by, you can freeze it by pressing the space bar.

To *erase* the sequential file, press *E*.

OTHER CONSIDERATIONS

When using the Shopping List on a C-128 with an 80-column monitor, you can remove the first two Pokes in line 2 and use the space they leave to add the Fast command, without having to renumber the program. The Pokes are needed only to set the screen and border colors for the 40-column screen.

Also, when you use the Save and Load functions of Shopping List, be sure to make all additions to and deletions from the Data statements prior to starting a new shopping list. ■

Linker 128

By Dale S. Brown

RUN It Right

C-128

In the 1986 *RUN* Special Issue, Morton Kevelson's article, "C-128 Programmer's Aid," contained a sidebar entitled "Missing Link Uncovered!" (p. 66) that dealt with the C-128's lack of a Merge or Append command. The article was helpful, but the Appender program in it had some problems.

Prior to an "appending load," the program changed the start-of-Basic pointers at locations 45 and 46 to locations 174 and 175, because, as the article stated, "the end of the Basic program is stored in addresses 174 and 175." However, addresses 174 and 175 actually store the end address for the last load, save or verify operation. As a result, if you load and edit the main program and then try to append a program segment or subroutine to it, the address that locations 174 and 175 point to will be obsolete.

Fortunately, Basic 7.0 gave me the tools to build an append program that really works. Linker, as I call it, first checks to see if you're in 40- or 80-Column mode and formats the screen accordingly. Then it prompts you for the filename of your main program, the number and filenames of the program segments or subroutines you want to link to it and the filename you want the final program to have.

There are three rules to follow in using Linker:

1. The main program and all the files to be linked to it must be on the same disk.

2. Subroutines or program segments to be linked to the main program must have line numbers greater than the main program.

3. The filenames of the subroutines or program segments must be entered at the prompts or assigned to the array variables.

Linker can handle no more than nine linked files each time it's used. This ensures that vari-

ables won't quickly exceed the memory limitations of bank 1, and it permits easy checking for erroneous input data in line 240. If you need more variable capacity, you can increase the DIM of B$(X) in line 65 and remove line 240.

Linker consists of five short programs for demonstration purposes. Here are brief descriptions of them:

Linker is the "work" program that links files together. Be sure to save it with the filename LINKER, because the demonstration programs will be looking for it with that name.

LinkLDR is a short, main program to use for demonstration purposes. It prompts you to insert a disk containing Linker, then loads and runs it.

PressKey.400 is a subroutine that provides a colorful "press any key to continue" prompt once you've performed some other action the program has requested, such as inserting the proper disk as prompted by the subroutine in InsertDisk.401, which prompts you to insert a disk, previously defined in KD$, into the drive.

SoundPrompt.402 is a subroutine that plays a note or notes (depending upon the value in KS) to alert the user that something needs doing. Lines 40210–40225 set up the notes the first time the routine is called. Line 40245 contains the tune—in this case, the first few notes of Yankee Doodle. If you change line 40425, be sure to change the 7 in line 40240 to match your final number of notes.

USING LINKER

Load and run the actual Linker program itself.

At the first prompt, input LINKLDR, and at the second prompt, input 3, the number of subroutines to link. Then input the filenames of the subroutines in this order:

PRESSKEY.400
INSERTDISK.401
SOUNDPROMPT.402

At the final prompt, input a filename for the combined program. It needs to be less than 16 letters long, and it can't already be used on the disk.

You'll notice that the filenames for the three subroutines contain the first three digits of their starting line numbers. These digits are referenced by a Gosub during execution.

MINI-LINKER

The sixth program, Mini-Linker, is for advanced programmers who may want to keep Linker in memory as a wedge. It does no disk-error checking, and it's meant to be loaded prior to start-

ing your program.

To set up Mini-Linker, you must append your main program to it, starting at Mini-Linker's line 100. Also, you must declare any variables in your main program at the beginning of the main program's listing, as you would in Pascal. Finally, you must declare, before line 100 of Mini-Linker, the subroutines that you want to link to your main program.

When you use a Gosub to access a new subroutine not previously linked to your program, declare the subroutine filename as one of the B$(X) array variables. Then, when you're ready to link files, go back and rearrange the B$ array numbers (the numbers in brackets) to agree with the ascending progression of line numbers for your subroutines.

Next, update the value of X in line 10 to agree with the total number of subroutines you'll be linking and input RUN 5. When that's done, enter DELETE -99 to eliminate Mini-Linker, and you'll be able to use the Renumber command. ■

Input Sentry

By Robert L. Senft

RUN It Right

C-64

Has your computer ever locked up while you were entering data into a spreadsheet or other number-crunching program? The chances are good that it has, especially if your software contains Basic's Input statement.

It's discouraging to get an error message after you've entered large quantities of data. And it's tedious to fix data-entry errors after you've already pressed the return key. Fortunately, these and many other data-entry problems are preventable. *Un*fortunately, numerous input routines, commercial and otherwise, don't do the job very well.

Recently, a brand name financial software package was observed to halt execution when input values were not to its liking. Each time this happened, the poor user had to turn off the computer, reload the program and start from scratch. Worse yet, he was given no clue as to what sort of error was committed. Not exactly what you would call user-friendly. There's a crying need for reliable, error-tolerant and user-considerate data-entry routines.

Many articles have addressed the Input statement's well-publicized limitations, which include the forbidden characters (commas, colons and quotation marks) as well as the sometimes cryptic error messages that can ruin the input screen format. Another nuisance is the unrestricted use of the cursor-control keys, which can lead to data errors or screen-scrolling problems.

Some, if not all, of these limitations have been solved, but many more remain. For instance, there may be times when you would like to limit maximum or minimum input values or to prevent a zero entry that might produce a "division by zero" error.

It would also be desirable to catch the accidental entry of alphabetic characters in numeric fields. Of even greater value would be the ability to easily

identify and correct errors without leaving the data-field area. It is usually much easier to fix data problems at entry time than later, when the output is all fouled up.

At this point you may ask, "How can the Input statement, or any of its substitutes, possibly do all this?" It can't. But don't despair—here at last is Super Input, the bullet-proof utility that can and *does* do it all.

WHAT IT OFFERS

Super Input is a data-validation subroutine that solves all the problems mentioned above and provides many additional features, as follows.

- Decimal justification in numeric fields.
- Controlled field length and field-positioning.
- Field area underlining, if desired.
- Audiovisual alert of improper entries.
- Error message and correction on bottom line of screen.
- Cursor on-off control.
- Crash-proof input sequence.

Here's how it works. First, nine control variables that define the data-input parameters are passed to Super Input. You must assign appropriate values to these variables prior to accessing the subroutine for the first time. You can do this on a variable-by-variable basis or with a Gosub 70, which sets them all to default values. (The values in line 70 were chosen to work with the demo program and can be changed to suit your specific needs.) See Table 1 for the list of control variables and their functions.

Super Input was designed to be easily merged with any Basic program. All internal variable names consist of two alphabetic characters that begin with the letter *X*. Be careful not to use the field-control variables outside the subroutine.

The demo program illustrates some of the possible entry formats. All data is printed to the screen at the end of the input sequence, or whenever the CTRL and left-arrow keys are pressed. Don't forget to delete the demo lines when merging Super Input with your own software.

POINTERS TO HEED

When using Super Input in conjunction with any of your programs, keep the following points in mind.

1. Make sure you initially assign acceptable values to the appropriate control variables. If you omit this step, all variables will be equal to 0, which is unacceptable for DP, FL and FP.

Also, since some control variables are interdependent, restrictions are placed on certain

combinations. These restrictions, plus acceptable ranges for all control variables, are stated in Table 1. A Gosub 70 in your Basic program, prior to accessing Super Input, will assign acceptable default values to all variables.

2. Make sure your data field does not attempt to wrap around to the next screen row. An example of this would be a field position, FP, of 20 and a field length, FL, of 30. Hence, the (FP + FL)<40 restriction.

3. Whenever you print an input prompt message, Super Input will attempt to place the data field on the same screen line, at whatever column your FP or DP assignments dictate. Make sure these control variables are sufficiently large to prevent the data from overwriting your prompt.

When your data field is too long to fit on the same line with the prompt, insert one or more blank Print statements, before the Super Input Gosub statement, to move the data field down a corresponding number of lines.

4. All control variables retain their current values until new ones are assigned. It is therefore unnecessary to re-assign values

Variable	Function	Range	Data Type
CU	Cursor switch (1 = on, 0 = off)	0 or 1	all
DD	Maximum number of decimal digits permitted	0–36	numeric
DP	Decimal position (column number)	2–38	numeric
FL	Field length (maximum number of characters)	1–37	string
FP	Field position (start column number)	2–38	string
HI	High input value limit	*	numeric
LO	Low input value limit	*	numeric
NZ	Zero input allowed (0 = yes, 1 = no)	0 or 1	numeric
UL	Underline data field (1 = yes, 0 = no)	0 or 1	all

Restrictions: (FP + FL)<40 and (DP + DD)<39
DP> (number of HI or LO significant digits) + 1
*HI and LO limit range is −999999999 to +999999999 and HI>LO

Table 1. Control variables and their functions.

each time you access Super Input, as long as the data items are similar. Change these variables only when the nature of the expected input requires it.

5. Super Input will accept an open quote in a string data field. However, if you later attempt to print this field from your Basic program, it's possible to run into some problems. Insert a POKE 212,0 immediately after your Print statement to terminate the Quote mode and prevent the printing of garbage following the data field.

Super Input Demo, a home inventory Basic program format example, illustrates how to integrate Super Input into your program.

USING SUPER INPUT

Once you've loaded and run Super Input, print your Input prompt message and enter GOSUB 10 (for string inputs) or GOSUB 11 (for numeric inputs). When entering strings, the cursor will appear at the specified column position, FP, on the same row with your prompt message. The field underline, if on, will extend right from the cursor by an amount equal to FL.

The numeric input is similar, except that the cursor appears at the specified decimal position, DP. All digits will scroll left until the decimal key is pressed. However, characters that would otherwise force the entry to extend beyond the underlined area will not be accepted. One exception is a leading plus or minus sign, which can extend one column left of the normal field area. This accommodation prevents sign prefixes from occupying a normally available digit position.

As you input information, each character is checked for proper type and, if correct, accumulated in a string variable. The entry is terminated with the return key. In the event of an input type or parameter error, you'll be notified visually (border flashes red) and audibly (short beep). A message describing the error and needed corrective action is displayed, after which the cursor returns to its last occupied field position and awaits your response. Also, the return key is temporarily disabled to prevent the accidental entry of incorrect data.

At this or any other point during data entry, you can edit with either the insert/delete key, which functions as a backspace, deleting characters as it moves left, or the left-arrow key, which erases the entire field. Either one of these two editing actions reenables the return key, which permits entry of corrected data.

In the event that Super Input excluded the error-causing keystroke, but you wish to enter the data "as is," press the return key

twice, and—assuming no further errors—your entry will be executed. Note that under no circumstances can you move the cursor outside the designated field area. The normal cursor keys are inoperative.

Upon return from Super Input, the variable I$ contains a string representation of your data. If the input was numeric, variable IV contains the floating-point value. At this time, it's up to you to determine where and how to store the data. This detail must be taken care of, as both I$ and IV will have new values when the next data item is entered.

Note that each data item is limited to one screen line (a maximum of 37 characters). Normally, this is more than adequate for most database applications. Graphics characters will not be accepted at any time, and only numbers or leading plus or minus signs will be permitted in numeric fields. In instances when your input value cannot be 0, you should set NZ equal to 1.

You can set the range of input values by assigning the desired high limit to variable HI and the low limit to variable LO. The left-of-decimal field length is automatically set to accommodate the larger of the absolute values of the HI or LO limits. The remaining control variable functions should be relatively self-evident from Table 1.

To escape the data entry sequence, press the control and left-arrow keys together; this sets internal variable XA equal to 6 and returns you to your main program. You may then branch to wherever you desire with an "IF XA = 6 THEN whatever" statement following a Gosub to the Super Input routine. If this check is omitted, then the control and left-arrow combination simply imitates the return key.

Super Input eliminates the normally time-consuming and error-prone aspects of data entry. Take advantage of it. ■

Flash Cards

By Barbara Schulak

RUN It Right

C64

During summer vacation, reviewing school subjects with your children can be a good idea, but you'll want to make it fun. Flash Cards is a program I wrote that fits the bill by turning arithmetic review into a game. You'll find it useful not only for vacation-time review, but to reinforce your child's learning throughout the year. Perhaps you'd enjoy practicing on it, too, if your calculator has erased the multiplication tables from your memory.

Flash Cards provides flash card-type practice on the computer, eliminating the need to buy or make sets of paper cards. The program is designed for interactive use between a child and parent, but youngsters can use it alone to a certain extent. A joystick plugged into port 2 is required for operation.

When you run the program, it will first ask for the type of problem your child wants to practice: addition, subtraction, multiplication, division or an assortment of all four.

Then you must select the range, 0–9, of numbers to be used. For example, if you choose 0 as the lower limit and 5 as the upper limit after selecting multiplication, you'll get problems with a multiplier in the range 0–5 and a multiplicand (the number being multiplied) in the range 0–9. If you specify the same number for both the lower and upper limits, your child can practice just one multiplication table, such as 5 times 0 through 9. This flexibility in choosing numbers makes Flash Cards appropriate for children at all levels.

Your final choice is how long the program should run. You can select any length of time from one to nine minutes, depending on your child's attention span.

After you've responded to all the options, Flash Cards will begin displaying problems in large numerals on the screen. When your child gives a correct answer, push the joystick fire-button to advance

to the next problem. If the child doesn't know the answer to a problem, push the joystick in the up direction to display the answer. Then, push the fire-button to move to the next problem.

Continue this process until time runs out. At the end of the program, the time elapsed and number of problems answered correctly are displayed. Press Y to play again.

I deliberated long and hard on whether to make Flash Cards respond to keyboard replies or to require a person-to-person verbal response. I finally settled on the verbal response, even though the computer can't check the correctness of the answers, because I felt typing might slow a child down in a speed drill. Also, verbal responses get parents involved in the learning experience.

My children have enjoyed Flash Cards, and their math skills have improved rapidly. I hope you and your youngsters enjoy it, too. ■

Mega-Magic—July

By Cameron Goodair

RUN It Right

C-128

I've added a pie-chart maker routine to *RUN*'s Ultra Hi-Res graphics program that appeared in the February and May 1986 issues. UH.Pie Charts works with version 1.1 of Ultra Hi-Res, which includes the @Fill and @HCopy commands.

To create a pie chart, boot up Ultra Hi-Res V1.1 and load and run the pie chart program. You'll be asked to give your chart a name, the number of sectors you want in the chart, the area of each sector as a percent of the entire circle and a short description of each sector to print in the legend area below the chart. The routine won't fill a sector with a pattern if it occupies an area less than 1.5 percent of the chart, so if you have more than one such sector, you might want to group them into an "other" category. ■

DFClone

By Mike Konshak

RUN It Right

C-64; disk drive; printer

Most users of the Datafile 3.6 database program, published in *RUN* in February 1987, will eventually want to alter the structure of a record file so they can use it for a new purpose. The February article mentioned that files created with Datafile 3.6 could be modified with the DFRestructure program *RUN* published in November of 1985. However, DFRestructure was designed to work with the original version of Datafile, and it has proven ineffective with Datafile 3.6, so I've written a new program, called DFClone, to take its place.

DFClone lets you create a new structure for a Datafile file, then choose the record data to be cloned—or duplicated—into it. You copy only those fields you need for your new file.

The program options are:

1. Copy all or part of an existing record file into a new record file that has the same structure (that is, number, lengths and names of fields).

2. Copy all or part of an existing record file into a new record file having a different structure (that is, different fields, field lengths, names and order).

As an example of the flexibility the second option offers, you could clone the records from a file having only six fields into a new file with ten fields, or vice versa. This option also lets you combine up to three fields into one and specify the positions of the new fields.

The following logical operators tell DFClone which records to transfer:

=	Field data equals search criteria
>	Field data is greater than search criteria
<	Field data is less than search criteria
<>	Field data is not equal to search criteria
??	Field data partially matches search criteria

You can search up to three

operators, fields and criteria (data items) to ensure that only the data you want gets transferred. AND and OR operations can be used within criteria to make a search more selective.

Let's look at a file of students in an elementary school. The file contains each student's name, grade (1–6) and latest test score (0–5, with A = 5 and F = 0).

Field #	Field Name	Record Data
1	NAME	Bill, Mary, etc.
2	SEX	M or F
3	GRADE	1–6
4	SCORE	0–5

Using DFClone, you can, for example, search fields 2, 3 and 4 to find all the boys in grades 5 and 6 who got A (5) on their last test. Here's the search input for this case:

Search Field	Operator	Data (Criteria)	Continue?
2 (Sex)	=	M	AND
3 (Grade)	>	4	AND
4 (Score)	=	5	END

The search input for finding all the third-grade girls whose last names begin with K–R is:

Search Field	Operator	Data (Criteria)	Continue?
1 (Name)	>	J	AND
1 (Name)	<	S	AND
3 (Grade)	=	3	END

DFClone can divide up a file that exceeds memory capacity. For example, suppose your club's membership list has grown to where it fills its file, and still more new members are anticipated. You could provide for the additional growth by dividing the file into several new files with the same structure, but dedicated to separate groupings of members, such as these:

Old File	New Files
MEMBERS	MEMBERS A–J, MEMBERS K–M, MEMBERS N–R, MEMBERS S–Z
COMP CLUB	CLUB C-64, CLUB C-128, CLUB AMIGA, CLUB PC
AUTHORS	FICTION, NONFICT, MYSTERY, SCI-FI, HISTORY

DFClone can extract fields for use in creating documents such as mailing labels. Say each addressee's first and last name, house number and street, city, state and zip code need to appear on the mailing label, in the three usual lines, and your mailing list record file contains eight fields. You need to transfer all the records in the file and fit them into three fields. Here's a comparison of the old and new files:

Old Structure	New Structure
#Field	# Field
1 LAST (Name)	1 NAME
2 FIRST (Name)	2 STREETBOX
3 CODE	3 CITYSTATE

4 STREETBOX
5 CITY
6 STATE
7 ZIP
8 PHONE

In the new structure, Field 1 contains old fields 2 and 1 (first and last names), Field 2 is old #4, and Field 3 covers old fields 5, 6 and 7. After defining the structure, you can transfer all the records or just those found to have certain data in common. Not every field moves to the new file, but you can use the fields in the old file to search for the records you need.

THE CLONING PROCEDURE

First, save DFClone to a work disk. Then, run Copy DOS 5.1, which lets you copy DOS 5.1 from your 1541 or 1571 Test Demo disk onto a DFClone work disk. Then run DFClone, and, after some introductory material, this prompt will appear:

```
PRESS CONTINUE, $ DIRECTORY OR
  QUIT
```

Pressing the $ key displays the disk directory, Q quits the program and C proceeds with cloning. If you press C, the following sequence of screens will appear. First:

```
  CLONE DATAFILE RECORD FILE

INSERT A DISK CONTAINING THE
DATAFILE RECORD FILE TO BE
CLONED. PRESS RETURN WITHOUT
ANY ENTRY TO EXIT.

NAME OF DATAFILE ? MAIL LIST
  <return>
```

The last line asks you to enter the name of the file you want to use as the source for cloning—our mailing list file, for example. After you type the filename, MAIL LIST, and press return, the program loads the structure of the file and then displays:

```
FIELD NAME(LENGTH) FOR MAIL
  LIST

1 LAST 15          5 CITY 23
2 FIRST 10         6 STATE 2
3 CODE 5           7 ZIP 5
4 STREETBOX 32     8 PHONE 12
192 RECORDS POSSIBLE IN OLD
  FILE
4 RECORDS CURRENTLY PRESENT

CREATE NEW FILE:
CLONE STRUCTURE AS ABOVE?
  YES NO EXIT
```

If you press Y, DFClone accepts the structure of this file as the structure for the new file as well. The program then advances to a screen that says TRANSFER RECORD DATA TO NEW FILE. You can choose to move all records or to find just certain ones to move. I'll elaborate on this option later.

If you press E, the prompt you encountered at the beginning of the program reappears, giving

you a chance to change your source file or quit.

Press N if you want to create a new structure with some of the fields from the Mail List source file. For example, let's use the following format to create a special file for sending form letters to any part of the world:

#	Field Name	Content	Length
1	NAME	First, Last	32
2	STREETBOX	Street or Box	32
3	CITYSTATE	City, State, Zip	32
4	COUNTRY	Country	16

The new file includes most of the Mail List fields, plus the addressee's country. Now that you've defined the new file structure, examine the original screen again. (Remember, you pressed N.)

```
FIELD NAME(LENGTH) FOR MAIL
  LIST

1 LAST 15          5 CITY 23
2 FIRST 10         6 STATE 2
3 CODE 5           7 ZIP 5
4 STREETBOX 32     8 PHONE 12

NUMBER OF FIELDS PER RECORD?
4   <return>

FIELD#1
TITLE ? NAME <return>
LENGTH ? 32 <return>
```

As the prompts appear, set up the initial structure for the new file by entering the field title and length you defined previously for fields 2–4. Then the following screen will appear:

```
YOUR SELECTION WILL ALLOW:

205 RECORDS IN A C-64 COMPUTER
468 RECORDS IN A C-128
  COMPUTER

PRESS REJECT OR ACCEPT 1 C-64
  2 C-128
```

If you're dissatisfied with the new structure and want to redefine it, press R. To accept the structure as it is, choose either C-64 by pressing 1 or C-128 by pressing 2. Of course, since we're using C-64 files, you should press 1. The old file-structure data will appear again, along with:

```
UP TO THREE OF THE OLD FIELDS
MAY BE COMBINED INTO ONE OF
THE NEW FIELDS.

ENTER FIELDS TO BE CLONED INTO
NEW FILE AS PROMPTED. ENTER 0
TO LEAVE NEW FIELD EMPTY.

NEW FIELD 1     NAME
OLD FIELD 1 ? 2 <return> etc.
          2 ? 1
          3 ? 0

NEW FIELD 2     STREETBOX
OLD FIELD 1 ? 4
          2 ? 0
          3 ? 0

NEW FIELD 3     CITYSTATE
OLD FIELD 1 ? 5
          2 ? 6
          3 ? 7

NEW FIELD 4     COUNTRY
OLD FIELD 1 ? 0
          2 ? 0
          3 ? 0
```

Note that, to leave the record field blank, you must enter 0s for the new fields.

Once you've entered all the data as shown, specifying which fields are to be combined and their proper placement, the program will ask:

```
IS THIS THE INTENDED CLONE? Y N

NEW FILE        OLD FILE
1 NAME          =[FIRST] [LAST]
2 STREETBOX     =[STREETBOX]
3 CITYSTATE     =[CITY] [STATE] [ZIP]
4 COUNTRY       =
```

If everything's all right, press Y to continue. If there's an error, press N. When you press Y, the transferring process starts, with the following screen:

```
TRANSFER RECORD DATA TO NEW
  FILE

CLONE ALL THE RECORDS FROM
  MAIL LIST
FIND RECORDS WITH COMMON
  DATA
EXIT

PRESS THE APPROPRIATE KEY
```

Pressing E sends you back to the beginning of the cloning process, A transfers every record in the Mail List file into the new file and F moves to the search routines, so you can choose field, data or search criteria and operators.

If you press F, the program presents a breakdown of the old file structure and prompts you for the first search field. Then a sequence of prompts appears, asking specifically what data you're looking for. Naturally, you may have to experiment somewhat to get the results you want.

The only logical operator that might confuse you is ??. It represents the same Find parameter that's used in other Datafile 3.6 programs, and it transfers all records in which the field you've specified begins with the characters you've specified. For instance, if the characters are S-M-I-T-H, all records beginning with SMITH will be moved.

As I mentioned earlier, DFClone can divide a file full of records into several smaller files. If you have two or more files filled with data, you can even divide each into smaller files, then sort all the smaller files together.

When you're working with DFClone, make sure the number of records in your new file doesn't exceed the number of possible records in your original file structure. Also keep in mind that DFClone uses arrays and memory similar to those that Datafile 3.6 uses, so you must be careful in cloning merged files. You could get an Out of Memory error. ■

Typing Tachometer 128

By Charles Kerr and Doug Larson

RUN It Right

C-128

"Don't look back; something may be gaining on you," quipped baseball's legendary Satchell Paige. Well, in this case, it's fine to look back, because what's gaining on you is both useful and fun. It's Typette 128, a program that will improve your typing proficiency, make you chuckle and help you memorize 12 witty quotations you can use on your friends.

The quotations appear one at a time at random. After you type each one as rapidly but accurately as possible, the program displays the percentage of the characters you got correct, your speed in words per minute and the number of mistakes you made. It also awards points based on a combination of your accuracy and speed.

When you run Typette 128, an attractive screen display will appear, with a quotation from a well-known personality. A rule above the quotation indicates its length in characters.

Type the quotation exactly as you see it, including punctuation, capitalization and character placement. As you type, your input will appear directly below the quotation, in reverse characters. Don't try to correct any errors you make, because we've disabled the editing keys to encourage accuracy.

The only way to escape from the quotation screen is to fill out the entire line with characters. If you don't want to type that particular quotation, you can fill the line quickly by repeating any character, but doing so will wreak havoc with your cumulative scores.

Once you've filled the line one way or the other, the program automatically checks for mistakes, then indicates the incor-

rect characters with little red arrows and displays your scores for that quotation. After you've seen how you did, press the return key to go to the next quotation, F1 to clear your cumulative scores and start over, F3 to view your cumulative scores or F5 to leave the program. Ten quotations make up a complete exercise, but you can press F3 or F5 at any time.

When you press F3, the screen displays your final cumulative scores, a rating chart and a mistake tally. The rating chart is based on ten quotations. The mistake tally lists all the keyboard characters in columns, with the number of times you typed them incorrectly displayed in parallel columns. Numbers for characters you've mistyped five or more times appear in red. After checking your total performance, press F1 to start over or F5 to leave the program.

Typette values accuracy over speed. You'll achieve higher scores by typing slowly and correctly than by typing quickly but incorrectly. Fifty bonus points are added each time you type a quotation perfectly.

Eventually you may want to replace the quotations we've supplied with some that are of your own choosing. Try to select quotations that contain characters you mistype frequently, and be sure the quotations are no more than 79 characters long.

To insert the new quotations in the program, list lines 450–560, move the cursor to where you want the replacements to go and type them in. Make sure they're enclosed in quotation marks, but that they don't contain any quotation marks. When the replacements are correct, press the return key to fix them in the listing. Finally, save the revised listing to disk, so you don't lose all your work when you turn off the computer. ■

Electronic Address Book

By Charles Krumholt

RUN It Right

C-64; printer (optional)

"A short pencil is better than a long memory." You've probably heard that old adage, and if you follow it, you probably keep a little black book for jotting down addresses and phone numbers. Well, now that adage should say, "a C-64 is better than a long memory," because I've written The Directory, a program that keeps a list of addresses and phone numbers and provides printouts. The Directory has made my little black book obsolete, and it'll make yours out-moded, too.

I keep a printout of The Directory's alphabetical file by each of my phones for reference and for jotting down new information. Once a month or so, I collect the printouts and make changes to the file. Even if you don't have a printer, you'll find The Directory's list helpful when it's time to write holiday cards or send out invitations.

When you add an entry to the file, the program checks to make sure it's not a duplicate. You can access individual entries by name, and I've provided a quick and easy way to scan the entire file. Program options include hard copy, delete, update and more. ■

Keycodes Revealed

By Bruce Fellows

RUN It Right

C-64

Designed for C-64 programmers, Keycodes is a handy reference utility that displays the ASCII codes and screen codes for the Commodore keys. This eliminates the need for you to search printed tables or to interrogate the computer to find these values.

The program resides in 451 bytes of memory at the top of Basic and can be accessed from most Basic programs without affecting their operation. You'll find Keycodes especially handy when you're writing Data statements to draw graphics characters or replacing long lines of CHR$ functions with a character string in quotes.

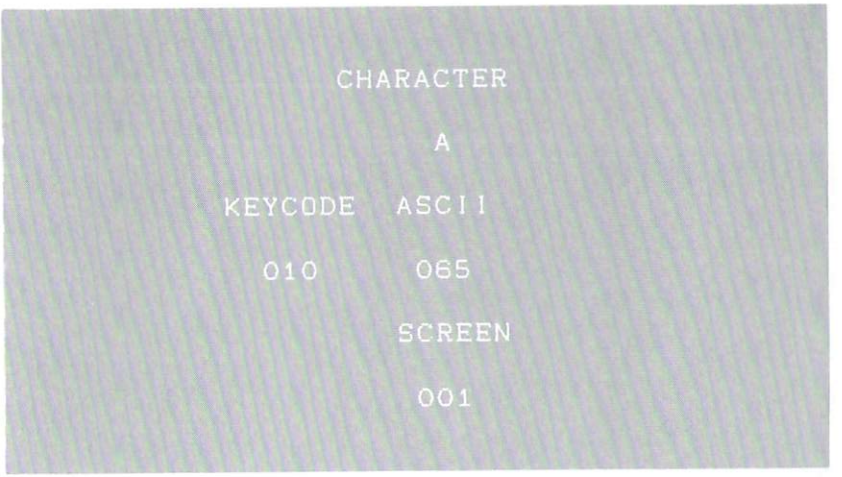
USING KEYCODES

First, type in and save Keycodes to disk. Then, before you begin a programming session, load and run it to install Keycodes in memory. When a five-digit SYS address appears on the screen, jot the number down. Then start your programming.

When you need to determine a key's code values, enter SYS and the address you wrote down. The screen will clear and display four headings: Character, Keycode, ASCII and Screen.

Now, press any key except restore or shift-lock (the only keys with no ASCII and screen-code values), and Keycodes will fill in information under the headings. Under the Character heading you'll see the character that would be printed in Quote mode by the key you pressed, and under the Keycode heading you'll see the keyboard matrix position of the key. Under the ASCII heading will be the ASCII value of the key, obtained from a table in Kernal ROM, and under the Screen heading will be the screen code of the key, as derived from the ASCII code by the screen editor routines.

Keycodes even displays values for the return key, which usually doesn't print, and the shift,



control and Commodore keys, which never print. It also reports values generated by a joystick in port 1.

Once you're done with that key, you can go on to others. When you've found all the codes you need, press the restore key to exit Keycodes and continue programming, or press the run-stop/restore combination for a Basic warm start. You can enter the SYS command again at any time to reaccess Keycodes.

In Keycodes mode, nearly all the normal keyboard functions are disabled. However, the shift, control and Commodore keys still work, so you can see the values of almost all key combinations, and shift/Commodore still toggles between the C-64's character sets.

One limitation of the program is that it reports a keycode of 15 for both shift keys, but the keycode value for the right shift key is actually 52. The ASCII and screen codes are the same for both keys. ■

Pegboard

By John Olsen

RUN It Right

C-64

The pegboard game has been around in various forms for years. It's played on a board perforated by 33 holes arranged in a plus-sign shape. At the start, pegs are placed in all the holes but the center one.

The object of the game is to remove from play as many pegs as possible by jumping over them, one at a time, with other pegs. Play continues until you can't move any more.

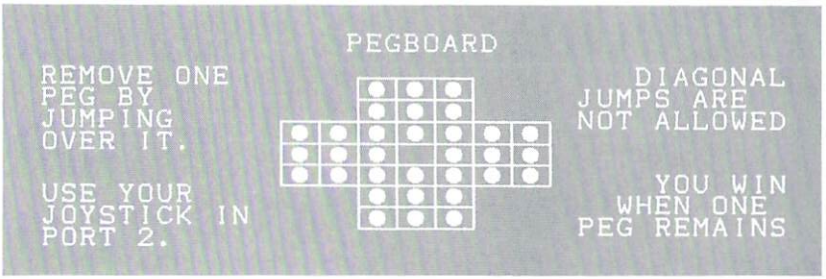
You win if you eliminate all the pegs but one—a task that's more difficult than it sounds. Usually you'll end up with several pegs stranded in different parts of the board with no adjacent peg to jump over. Avoiding this downfall takes a combination of strategy and luck.

My Pegboard program simulates the traditional board game. The playing surface is displayed on the screen, along with a summary of the rules. The program uses graphics, color and sound to enhance play.

You move the pegs with a joystick, plugged into port 2, that places a yellow cursor in any desired position on the board. When it's over the peg you want to move, press the fire-button, and that peg turns into an asterisk. Then use the joystick to place the yellow cursor on the position where you want the peg to go. Pressing the fire-button a second time moves the peg from its original position to the newly chosen one and eliminates the peg it jumped over. You can move horizontally or vertically, but not on a diagonal.

Pegboard checks your moves and permits only legal ones. It also scans the board continously to see when no more moves are possible. When that occurs, the game is over, and the computer counts the remaining pegs to see if you've won.

In writing Pegboard, I employed a programming strategy that doesn't require arrays, takes little memory and makes the pro-



gram run quickly. Since the game board is constantly displayed on the screen, I have the program Peek the screen memory to determine which holes contain pegs and which are empty. This Peek also tells the joystick-controlled cursor where the edges of the game board are. With this approach, the program needs to keep track only of the cursor position. Nearly everything else is kept in the screen memory.

When you play Pegboard, you'll find that it's fairly easy to leave fewer than ten pegs on the board, and if you're lucky, you may remove all but two or three. However, it will take real skill to remove all but one. Keep working at it, and good luck! ■

Mega-Magic—August

By Alton C. Williams

RUN It Right

C-64; printer (optional)

One feature possessed by screen editors on many computers more powerful than the C-64 is a locator function for finding all occurrences of a specific string in a listing. String Locator fills this gap by locating all the lines containing a specified string of characters and printing their numbers to the screen or printer.

Simply run the program to activate the Locate function. You can then type in or load a program and use this function to help you debug or analyze a program.

Each Basic command is stored in memory as a single character. If you could peek inside memory for the word PRINT, for example, you wouldn't find it, because the editor automatically tokenizes the word to one character. However, Basic doesn't tokenize any word enclosed in quotation marks.

It's obvious, then, that the locate function must be able to distinguish between tokenized and untokenized strings. If, for example, you type in LOC GOTO, the program locates all tokenized occurrences of GOTO, whereas typing in LOC †GOTO finds all instances of GOTO within quotes.

Your computer then asks if you want the line numbers where a string's been found to be sent to the screen (press S) or the printer (press P).

The function occupies memory locations 49152–49797 ($C000–$C285). So, if you're using a machine language routine with your Basic program, make sure that it doesn't occupy this area. If it does, deactivate the Locate function before you run your program by entering SYS 49155 and pressing the return key.

The more you use this program, the more you'll wonder how you ever got along without it! ■

Sprite Database

By Louis Wallace and David Darus

RUN It Right

C-64

Sprite Database is a special editor designed to make sprites you create (up to 1000 of them) easily available for later use—without having to remember their names and locations in memory. It also lets you create Basic Data statements to include in your programs and binary files to load directly into memory. The Data statements can start at any line number. The binary files can contain as many sprites as you wish, in any order, and they can have any load address.

With Sprite Database, it's also possible to produce meaningful printouts of your sprites on any printer. The program's powerful editor supports single and multiple colors, rotation, reversal and flipping.

To get started, save the program to a freshly formatted blank disk, then run it. The main menu screen appears, divided into three areas. In the upper-left is the sprite grid, which displays in a large format the sprite you're designing. The right side of the screen shows the actual sprite in its four possible sizes, along with its name and colors, plus the number of sprites in the database.

The bottom of the screen contains the main menu. To choose an option, use the left-right cursor key to move the highlight until it's on the option you want; then press the return key. On this menu only, you can also press the letter corresponding to the option you want.

The first time you use Sprite Database, you must create the main file. From the main menu, choose NewDB, which sends you to a second menu. There, choose CreateDB to create the relative file on disk that you'll use in all future sessions. The disk drive runs for a few minutes; then you are returned to the menu.

Start each session after the first by choosing NewDB, then OpenDB. Opening the database reads in the sprite index file and tells the program how many

sprites there are. After the database is open, you can create a new sprite or edit an old one, export sprites or perform any other of the program's functions.

Sprite Database comes with a built-in Help feature, available from the main menu, to supply you with information about any of the options. You can abort any function or exit any submenu and return to the main menu by pressing the run-stop key. ■

Please send me back issues of ReRUN

____ January/February 1986
____ March/April 1986
____ May/June 1986
____ July/August 1986
____ Productivity Pak II
____ September/October 1986
____ November/December 1986
____ January/February 1987
____ March/April 1987
____ May/June 1987
____ Disk version(s) at $16.47 each*

* *Price includes postage and handling. For foreign air mail, please add U.S. $1.50 per item. Prepayment only.*

☐ Payment Enclosed ☐ MC ☐ VISA ☐ AE

Card # Exp. Date

Name

Address

City State Zip

Signature

ReRUN • 80 Elm Street • Peterborough, NH • 03458

BEAT THE RUSH!

Please send me:

☐ **1 year (6 issues) for $69.97**
☐ **September/October 1987 ReRUN disk for $16.47.***

***Available in October 1987.*

Includes programs for C-64 and C-128 (in both 64 and 128 modes).

Price includes postage and handling. For foreign air mail, please add U.S. $1.50 per item and $25 per subscription. Prepayment only. Subscriptions will start with the current issue.

☐ Payment Enclosed ☐ MC ☐ VISA ☐ AE

Card # Exp. Date

Name

Address

City State Zip

Signature

ReRUN • 80 Elm Street • Peterborough, NH • 03458

13 *RUN* Programs Included on this Disk:

Programming Utilities ► Typing Aid ►
Shopping List ► Address Book ►
Education ► Games ► Graphics

From the July RUN:

- Tri-Solitaire
- Attention, Shoppers!
- Linker 128
- Input Sentry
- Flash Cards
- Ultra Hi-Res Pie Charts

From the August RUN:

- DFClone
- Typing Tachometer 128
- Electronic Address Book
- Keycodes Revealed
- Pegboard
- Locator 64

Bonus Program:

- Sprite Database

If any manufacturing defect becomes apparent, the defective 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 Elm Street • Peterborough, NH 03458

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

The programs in ReRUN are taken directly from listings prepared to accompany articles in *RUN* magazine. They will not run under all system configurations. Use the RUN It Right information included with each article as your guide.