COMPUTE!'s DATA FILE HANDLER

for the Commodore 64

Blaine D. Standage
John L. Darling
Kenneth D. Standage

A sophisticated electronic data base manager for the Commodore 64. Includes a data base processor, a sequential data file editor, and a disk operation support set. Will also work on the Commodore PET/CBM computer.

A **COMPUTE! Books** Publication

$12.95

COMPUTE!'s DATA FILE HANDLER for the Commodore 64

Blaine D. Standage
John L. Darling
Kenneth D. Standage

COMPUTE!™ Publications, Inc. abc
One of the ABC Publishing Companies
Greensboro, North Carolina

Printed in the United States of America

ISBN 0-942386-86-8

10 9 8 7 6 5 4 3 2

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Commodore 64, PET, and CBM are trademarks of Commodore Electronics Limited and/or Commodore Business Machines, Inc.

Contents

Foreword

Organizing information for fast retrieval and quick sorting is one of the things computers do best. Business computers have been doing it for years. So why not have a powerful data base manager for your PET or Commodore 64? That's exactly what *COMPUTE!'s Data File Handler for the Commodore 64* is—a series of integrated programs that give you a sophisticated and efficient data-handling system.

The Data File Handler will store, sort, merge, split, extract, and print records you've created. It allows you to organize your data in records with up to 20 fields. Sorting is allowed on any field. You can create print formats and save them to disk for use at any time. And the entire system works on either the Commodore 64 or PET computer.

Although the focus of this book is data base management, the book is much more than that. The authors have also included a sequential file editor that can be used with any sequential file. You can edit and resave word processing files or look at those mysterious SEQ files. Also included are four machine language routines, instructions on how to use these routines in your own programs, and more than a dozen useful disk commands.

Since all the BASIC programs are listed here, you can learn from the techniques by studying the listings. If you plan to type in the programs, we've included an error-checking program, "The Automatic Proofreader," that will verify your typing line by line, making perfect program entry easy.

COMPUTE!'s Data File Handler for the Commodore 64 includes everything you need, from the system itself to vital information. With this package and your computer, you'll be organized like never before.

Preface

For every good solution there must be a problem which caused it. In this case the problem was "How do I manipulate large data files?"

I wanted to create some large files and do some manipulating of those files. I wanted to be able to sort, merge, and print the files. This led to a single BASIC program which did some of the easier tasks, showed me a lot of other features I needed, and started a long evolutionary process of developing the programs of the Data File Handler (DFH) set. Along the way I have had a lot of help.

The DFH Editor was conceived and written by my close friend, John L. Darling. He also contributed the machine language sorting and partitioning subroutines. These routines were originally written to be used as independent programs, and that capability has been preserved. John also served as my chief technical editor and critic. I was careful to return that favor when working on his programs.

My brother Kenneth was the first to see a need for the file-splitting and data record extraction features of DHF Split. As a result, he was selected to create the program and write about it.

The result of these efforts is a coordinated set of programs for data file handling. Although I didn't know it then, these are the programs I wanted in the beginning. They are ones I can really use!

The DFH programs are efficient when used on small or large data sets. (The largest processed to date is a 48-file set containing over 25,000 records.) Because these programs employ a generalized approach, they have already been used by my associates for a wide variety of tasks.

I would be the last to suggest that there are no bugs in these programs. Quite the contrary, I feel there is no such thing as a perfect program, and these programs will be no exception. However, we have removed all the known bugs and the programs have been carefully tested. I believe they will do a good job for you. I hope you enjoy using them.

Blaine Standage

Chapter 1

Getting Started with DFH

Introduction to DFH

This book was written with two major goals in mind: to present and explain the operation of a coordinated family of general-purpose Data File Handler (DFH) programs; and to provide some insight into the overall procedures for working with data files, with or without these programs.

In keeping with these objectives, it is to your advantage to treat this book as a reference manual rather than a cookbook or a novel. Cookbooks are dictatorial and novels describe how someone else did it. Most of the material in this book is designed to help you create your own method of handling data. It will provide you with a powerful set of tools and instructions for using them.

This book is written primarily for the computer user rather than the programmer, and no programming knowledge is required. However, skilled programmers will find that the DFH programs are a solid base from which they can build and continue in many directions.

Program Protection

While we expect and trust you to honor the copyright laws, we do not want to prevent you from altering the programs to satisfy your own special needs. Therefore, none of the programs contains any copy-protection code or other programming techniques which would, in any way, restrict your handling of them.

The complete listings for all BASIC programs and the necessary data to create all machine language programs are included in the final chapters.

A disk containing all the programs, including the source code for the machine language programs, is available by calling COMPUTE! Publications toll free at 1-800-334-0868 or by using the coupon in the back of this book.

If you wish to enter the programs manually, you should refer to Chapter 9, "Entering the Programs," for details. There we have done our best to make the manual entry process as easy and error-free as possible.

Equipment Required

The Data File Handler (DFH) programs will work on any Commodore 64 computer with a single disk drive such as the 1541, and any PET or CBM computer with BASIC 3.0 or BASIC 4.0 ROMs. This includes the 2000, 4000, 8000, and 9000 series computers, but does not include the original version PET computers. When run on a PET or CBM, a Commodore dual disk drive is required.

A Commodore dot-matrix printer is recommended to fully utilize all printing features. However, only a few minor capabilities will be lost if a daisywheel printer is used.

Disk drives are expected to function as device 8. Printers are expected to function as device 4.

Other computer/disk combinations and different device assignments can be supported through program changes. Although no procedural details are given in this book, considerable care was taken during the programming to allow for such customizing changes. Pay special attention to the variable TY (TY stands for type) if you attempt a configuration change. This variable is used in all the BASIC programs to control special actions which are dependent on the computer type and disk drive type.

Program Organization

There are six BASIC programs and three machine language programs in the Data File Handler series.

BASIC programs:

DFH BOOT
DFH SORT
DFH PRINT
DFH MERGE
DFH SWAP
DFH SPLIT

Machine Language Programs:

DFH SUBS$79
DFH ED.C64$90
DFH ED.PET$70

DFH BOOT is the only program you need to remember by name. It provides overall executive control and allows you the freedom to select the functions you wish to perform, without regard to which programs are required to get the job done. This bootstrap determines which computer is being used and automatically makes any necessary adjustments to computer and program configuration. The loading and execution of all the other programs is controlled by DFH BOOT!

All of the BASIC programs reload DFH BOOT and return control to it when they have completed their tasks. Since these programs load and run each other, it is necessary to give each program the exact filename as it appears above.

Each of the BASIC programs uses the machine language subroutines contained in DFH SUBS$79. These subroutines are automatically installed and protected by DFH BOOT.

The machine language subroutines can also be used by BASIC programs of your own design. An entire chapter is devoted to the details of how this is done. The final machine language programs are two versions of the DFH Editor, one for the Commodore 64 and the other for PET/CBM computers. The DFH BOOT program can be used to load, protect, and activate the DFH Editor. However, the DFH Editor is a completely independent program which will not return control to DFH BOOT!

The DFH Editor provides the ability to directly access and manipulate sequential data files in much the same way you handle BASIC program files. A variety of new and powerful commands are provided, and many of them can also be used for handling program files.

The DFH Editor also provides a powerful Disk Operation Support (DOS) command set. The appearance (syntax) of these commands is very similar to existing Commodore disk commands. However, these commands provide many extended options which give increased power and flexibility in handling disk files!

Later in the book there is an extensive discussion of the features and operation of the DFH Editor. Also included are the procedures for loading, activating, and using the DFH Editor as a completely independent program.

Data Capacity

It would have been quite an impressive job of programming had we been able to write the DFH with unlimited capability. Unfortunately, this is not the case, and we will all have to live within the limits of the hardware and program design. But we believe you will find the capacity quite adequate for most applications.

- Data File Structure: Sequential data files containing single-field or multifield records. The field delimiter is selected by the operator.
- Maximum Fields: Up to 20 fields per record.
- Maximum Record Length: 74 characters. This may be the most significant limit.
- Maximum File Size: Up to 700 records or 14,000 characters for DFH SORT (Create, Edit, Sort, and List functions). For all other functions, file size is limited only by disk capacity.
- Suggested File Size Limit: 650 records or 13,000 characters, to allow space for editing and additions. This limit is used by the merge program as a default which the operator may ignore if he or she desires.
- Maximum Number of Files: 50 files for the merge program. No limit for other programs.
- Maximum Number of Disks: 50 disks for the merge program. No limit for other programs.

Program Features

The following features apply to the DFH programs as a group, except for the DFH Editor. The file-handling and DOS capabilities of the DFH Editor are detailed in a separate chapter.

- Sorting: Sort on any individual field or on complete data records. Sorts in ascending or descending order. The machine language sort routine will sort 650 records in about five seconds.
- Editing: Add, edit, and delete functions. Monitored to prevent errors, such as including delimiter as data, creating long records, etc. Previous content default prompting for add and edit.
- Listing: Lists file with line numbers assigned for editing reference. Lists in character case currently in use for screen display. Character case can be changed from within any program.
- Printing: Fully formatted printing per operator specifications. Page and column headings and controllable page numbering. Successive file printing with or without page breaks. Print formats can be saved to disk for later use. *WordPro* file cre-

ation of formatted pages on request, complete with global file linking.

- Merging: Merges up to 50 presorted files from up to 50 disks. Merges in ascending or descending order based on full record content or on the content of any single field. Merged output is saved in 1 to 99 files as requested by the operator. This function can be used to break large files into smaller, uniform-size files.
- Restructuring: Five record-reorganization modes are available for use on a field-by-field basis. This function can be used to change field order, concatenate fields, create new fields, add constant data, and delete fields.
- File Splitting: Files are split based on data changes in specified, full or partial, data fields. Files can be split automatically at predefined data change points, or the operator can maintain constant control of split/no-split decisions.
- Record Extraction: Records are selectively extracted and placed in a new file, based on operator-defined data content matches. This function can run automatically or under continuous operator control.

Simplified Operation

This section presents the minimum necessary operating instructions. It assumes you have all the programs properly saved on a disk. If you do not have the programs saved on disk and wish to use the Data File Handler as you read about the features and procedures, you will need to jump to Chapter 9 at this point.

The instructions provided in this section rely very heavily on the user-friendly nature of the DFH programs. With the exception of the editing and disk operation commands of the DFH Editor, the programs provide enough aid to the user that they are essentially self-teaching!

This self-teaching feature has been proved several times by giving the programs to a new user with absolutely no written instructions. A few words to indicate that the overall purpose of the programs is to create and process data files were all that was necessary.

Beyond the fact that learning is slower without written instructions, there is a serious disadvantage to the simple "load and run" method. Even after a few weeks of using the programs, you will still not be aware of many powerful capabilities of the DFH programs.

On the other hand, maybe you don't want to know "all about data files," but just have a specific job to do.

Just Load and Run

For a really quick way to get started creating and processing data files, simply insert your program disk and type:

```
LOAD"DFH BOOT",8
```

Then press RETURN, type RUN, press RETURN again, and follow the instructions. No kidding—that's all there is to it.

Except for the utility commands of the DFH Editor, all of the DFH programs are almost completely self-guiding. It is hard to make an error, and even the errors you make are usually no problem. The programs will almost always assist you back into normal operation.

When you load DFH BOOT and run it, the machine language subroutines are automatically loaded and protected. Next, a program identification and summary information page is presented and will remain in place until you indicate you are ready to continue. The main menu will now appear from which you can choose one of the DFH processing functions to Create, Edit, List, Sort, Merge, Print, Restructure records, Split data files, or Extract records from files. You can also choose to activate the DFH Editor, but that is a separate topic, as we will see in a moment.

When you select any of the DFH processing functions, the appropriate processing program will automatically be loaded and run. Additional menus or instructions will be presented to guide you through the desired process and finally return you to the main menu for another selection.

If you are using a 64, the computer will be set into a PET Emulator configuration just before leaving the DFH BOOT program for the first time. The start of BASIC will be at $0401, and the start of screen will be at $8000. When you exit through the boot program, the computer will be returned to normal configuration, except that the top of memory will be left at $9000 to protect the DFH Editor in case it has been loaded. Each of these actions will happen automatically when necessary as you respond to the program prompts.

The DFH Editor

One item on the main menu is very different. Selecting the DFH Editor function will cause the DFH Editor program to be loaded and activated. The top of memory will be set to protect the program. Except for some memory reduction, the computer will be returned to a normal power-up configuration.

You might now continue to operate your computer in any way you wish and never realize that anything was different. Actually, you now have a complete set of disk operating system (DOS) support commands and sequential-file editing commands at your disposal.

The DFH Editor operations do not depend in any way on the rest of the DFH programs. Therefore, you can usually leave the DFH Editor activated at all times to take advantage of the added DOS and editing commands. There will be some exceptions to this rule. When another of your programs must use memory needed by the DFH Editor, they cannot coexist, so you must choose one or the other.

DOS Commands List

The following is a quick-reference listing of the DOS commands syntax. For details refer to the DOS Commands section.

Read disk error channel.

>

Display disk directory.

>$ dr:qualifiers

Copy disk file.

>C dr:newname=dr:oldname

Copy and Append files on disk.

>C dr:newname=dr:oldname1,dr:oldname2...

Duplicate disk. (PET only.)

>D dest dr=source dr

Initialize disk.

>I dr

New (format) a disk.

>N dr:diskname,id

Rename a disk file.

>R dr:newname=dr:oldname

Scratch a disk file.

>S dr:filename

Validate disk.

>V dr

Load a program or sequential file.

/ dr:filename,qualifiers

Load and run a program file.

↑dr:filename

Save a program or sequential file.

←dr:filename,qualifiers

Verify a program or sequential file.

] dr:filename,qualifiers

Append a program or sequential file to the one in memory.

& dr:filename

File Edit Commands List

The following is a quick-reference listing of the file edit commands syntax. For details refer to the DFH Editor Commands section.

Add a character to end of records in range.

;AD character,range

Automatic line numbering by increment.

;AU increment

Change Screen display case.

;CS

Delete lines in range.

;DE range

Erase screen Down from cursor.

;ED

Erase screen Up from cursor.

;EU

Find and Change character strings in range.

;FC /oldstring/newstring/,range

Find character strings in range.

;FI /string/,range

Set BASIC Mode. (For program files.)

;MB

Kill Mode. (Disable the DFH Editor.)

;MK

Set Text Mode. (For sequential files.)

;MT

Insert Quote at start of records in range.

;QT S,range

Renumber lines in range.

;RN incr,newstart,range

UnNew (cancel a NEW command).

;UN

Display DFH Editor commands menu.

; (or ;+invalid command code)

Data File Structures

The DFH programs are all written to handle data files. To do this they rely on a well-defined and uniform file structure. This is similar to a group of people intending to have a conversation—everything will go well if they all speak the same language.

The file structure is the backbone of any file-handling system and influences all aspects of the processing programs. Consequently, designing or selecting a file structure requires close attention. Obviously, a file structure has already been picked for these programs, but we think you should know how and why it was chosen. This should be of particular interest if you decide to write programs to extend the power of these programs to meet your own unique needs.

First, there is no such thing as the best data file structure. That would be like saying that a luxury sedan is the best vehicle. It might be in some cases, but not on the motorcycle trail. In short, the *best* can't be determined until you define the job to be done.

So, where does that leave us? The bad news is that no file structure can handle all data file processing needs. The good news is that a well-chosen structure can support a large percentage of those needs.

The Most Common File

The most common data file has a structure so simple that we often don't even use the word *structure* when describing it. It is created by writing data items to a data file one after the other on a "one record = one data item" basis.

With a knowledge of how the data was saved in these simple files, a programmer can recall the data for processing. If numbers were saved, the processing program will input to numeric variables. If characters were saved, the inputs must be to string variables. Saving a mixture of numbers and characters can cause problems if the expected inputs get out of sync with the actual data types. It is one thing for a program to input data, determine it is incorrect, and let the operator take some appropriate action. But it's a disaster when the program detects an error and simply quits in the middle of the job.

This suggests that the first rule about data files is that they should contain only characters, never numbers. Numeric characters can always be converted back to numbers with the VAL command.

Please don't think there is anything wrong with a simple "one data item equals one data record" filing method. In many cases nothing more is needed. However, life starts to get more complicated when the data begins to come in sets. And as we will see, this is more the rule than the exception.

Data Sets and Records

A *set* of data is a number of data items which are closely related to each other. Sometimes they must occupy a fixed position in relation to other sets in the file. In other cases there is only a loose relationship with the other sets in the file.

In essentially all cases, the items within each data set must be kept together. A name and address is a common example of a data set. When you want to sort a name and address file, you may be concerned with the names, but you also don't want to break any of the sets in the process.

One way to keep data sets grouped together is to save all the items of a set in a single data record. This concept is built into the file structure we have chosen for the DFH programs. When you think of it, the great majority of all data you might want to process has a *set* quality about it. The sets may be small, like a simple expense record containing only dates and amounts, or they can be quite large, like recipe files for your kitchen.

Why Not Recipes?

The recipe file is an interesting example. Did you ever wonder why everyone talks about getting a computer to handle recipes, but no one ever seems to do it? It sounds reasonable that the computer should be able to prepare a shopping list and quantity-scaled recipes for an entire week if you give it meal-by-meal menus and the number of people to be present for each meal.

The problems here come more from the data storage and handling requirements than from how to program the calculations. The recipe program illustrates the need for a complex file structure. The requirements for a data set which has a title, an ingredient list with quantities, and a paragraph or two of instructions are not a trivial problem.

In case you are wondering, short of purchasing a professional program, the usual solution to this recipe problem is to give up and feed the recipes to a word processor program. Now, even though you still haven't automated your shopping list, you can at least tell your friends and neighbors that you have your recipes in the computer.

If we stop short of what might be called complex data sets, we find that we have already covered most of the normal situations. The typical data set contains a reasonable amount of well-defined data. It is here that we concentrated our efforts. We tried for a file structure that would provide maximum efficiency in this middle ground of data processing. At the same time, the structure allows us to work with complex data sets if we are willing to give up some efficiency.

Real-World Limits

To further define what we mean by "a reasonable amount of well-defined data," we must take a look at the realities of our computer. We have decided that there are advantages to storing data as strings of characters. What limits does that impose?

First, no string can be longer than 255 characters. Your computer simply won't handle longer strings. Also, if we plan to use the INPUT# command in our processing programs, we are faced with an even smaller number. No more than 80 characters can be input from a file. Another absolute computer limit.

Since the alternative to the INPUT# is the extremely slow GET# command, let's accept the 80-character limit. Actually, we are going to reduce the limit to 74 characters a little later, and you'll see that we can live with it quite easily.

It might be very difficult to design a good file structure with only 74 characters per record if we followed the old traditional approach of allocating a fixed number of character spaces for each data field. That method wastes a lot of space because the allocation must provide space for the longest data item. We can't afford to design that type of inefficiency into our programs.

We will stick with the idea of each record containing a set of data items, each in its own field. If we simply throw out all the unused spaces, we can pack a lot more data into the average record. But then how can we tell where one data item (field) ends and the next begins? Fortunately, there is a simple answer: We can use a *delimiter.*

A *delimiter* is any character that is used only to mark the boundaries between data fields. The price we pay to get rid of the wasted space in fixed-length fields is that the character selected as a delimiter can never be used as a part of the data.

The loss of any character can be a problem. If we try to choose a standard delimiter character, someone will have data that already contains that character. DFH avoids this problem by leaving the choice of the delimiter to the user.

Let's take a look at three records from a name and phone number file created with only the rules we have defined so far:

```
ED*113-525-7457
JOE*124-632-0808
SUSAN*012-415-9454
```

Editing and Line Numbers

At first glance the records in the preceding example seem to illustrate a reasonable file structure. In fact, some of our early experiments used a structure similar to this one. It didn't take long for the problems to show up.

We were using an editor program similar to the one included in this book and routinely used it to make changes in data files. To do its job efficiently, the editor needs a line number assigned to each data record. The editor assigns these numbers as the data records are loaded into memory, and removes them when the data file is saved to disk.

The function of the line numbers is similar to the line numbers in a BASIC program. They are used for editing the data in memory. Just as in a BASIC program, the line numbers must be counted in the 80-character per line editing limit.

This is why the rule of 74 characters per record was formed—to make room for four-digit line numbers. Remember that these numbers never exist on the disk. The DFH Editor is the only routine that uses them, but the editor functions are so valuable that it was well worth giving up these few characters per record.

Special Characters

When we introduced shifted characters into data files to get entries like "Bob" instead of "BOB", it was obvious that a leading quotation mark for each record would be a great help. Shifted characters will not list correctly unless they are inside quotes.

Adding the leading quote provided many other side benefits. In effect, it removed all restrictions as to what characters could be included as data. Special characters such as commas, colons, cursor motion characters, etc., could all be used as data. The quote itself is still a restricted character, but now it is doing something for us.

The leading quote also allows leading spaces in the first data field, not just the later fields. That idea was expanded to allow trailing spaces in the final field by adding a trailing delimiter character. Leading and trailing spaces are often quite useful when aligning some types of data for proper sorting.

Applying these rules, our previous example records might look like this:

```
1010 "ED*113-525-7457*
1020 "JOE*124-632-0808*
1030 "SUSAN*012-415-9454*
```

The line numbers are included only to show how the records would look when examined with DFH Editor control.

A Special First Line

Even with the file structure this well-defined, some small aggravations still remained. When using the DFH Editor to make changes, we would sometimes forget the name of the file we were editing. Also, it was inconvenient to have processing programs scan to the end of the first record just to find out what delimiter was being used.

Both of these problems were solved by adding a special first line to the files. This line (record) is never used for data. The first nonquote character in the first record is used as the delimiter for the file. This character is followed by file identification constructed in the form of a DFH Editor SAVE command. The following first line would work just fine for our example file:

```
1000 "*←@0:TELEDATA
```

The asterisk is the delimiter for the file. The next character is a left arrow. This is the single-character DFH Editor SAVE command. The @0: indicates that the SAVE is to be done in replacement mode to drive 0. The remaining characters are the filename.

This line can be used as a SAVE command by listing it; then typing spaces over the line number, the quote, and the delimiter; and then pressing RETURN. If it seems easier or more logical, you can accomplish the same thing by placing the cursor on the left arrow, deleting to the left margin, and pressing RETURN. In any case, using this line as a SAVE command eliminates the possibility of a typing error in the filename.

We have now defined our file structure: multifield records of up to 74 characters, each record preceded by a quote, each field ending with a delimiter, the delimiter specified by being the first nonquote character in the first record of the file.

If you still have both eyes open, you've noticed that we have not told you how to handle complex data sets with this file structure. On that point we ask you to have faith for a little while. The question is much easier to answer by example, and that will be done in the Applications chapter (Chapter 6). We're confident you'll be surprised by the complex data that can be handled with this simple file structure.

Error Sources and Handling

In the first part of this section we will discuss two sources of error inherent in the Commodore disk system. Either can be the source of a major disaster. The potential danger is such that these sources of error should be common knowledge to any Commodore computer user. However, we have found this is not usually the case.

The last part of this section will discuss common operator mistakes and the protections which are built into the DFH programs to prevent those mistakes from causing any damage.

There is no intent to present an in-depth study of error handling and error-protection coding. That could be the subject of an entire book.

No Duplicate Disk IDs

It is very important that every one of your disks have its own unique disk ID code. Duplicate IDs can cause contamination and loss of disk data. This results from the methods used to maintain the Block Allocation Map (BAM) information in the disk directories.

The BAM is a system of internal bookkeeping that the disk controller uses to remember which blocks contain valid data and which ones are available for new storage. In Commodore systems, a block is the same as a sector. When a disk is initialized, either automatically or by direct operator command, the BAM is read from the disk and placed in the internal memory of the drive controller.

Some years ago, it was reported that some disk file contamination problems were probably caused by an unidentified bug in the *replacement save* command code, and owners were advised not to use that command. We are absolutely convinced, through continual use of the replacement save, and from detailed examination of the drive controller code, that no such bug ever existed. Rather, we believe *duplicate IDs* have been and still are the major cause of disk data loss.

Whenever you command any writing operation, the ID on the disk is checked against the ID which was obtained during the most recent initialization. If these two IDs are different, a disk initialization must be performed before any writing can take place.

On model 1541, 4040 and early model 8050 drives, this initialization will be performed automatically. On model 2040 drives, the operator is required to command the initialization. On late model 8050 and 8052 drives, the situation cannot occur because initialization is performed automatically each time the drive door is pushed to the closed position.

If the ID on the disk and the ID in the drive controller memory are the same, the drive controller assumes (logically enough) that the disk has not been physically changed. It then proceeds to use the BAM that it obtained during the last initialization to determine where to write the new data. After writing the data, the updated BAM is written to the disk in the drive.

If the disks had, in fact, been changed but the IDs were the same, you are going to be in trouble. The disk controller, assuming that the disk has not been changed, will refer to the BAM in its memory to find open blocks where it can write data. Since the BAM in memory is not the correct one for this disk, the blocks which it shows as being open may or may not already contain data. If these blocks do contain valid data, it will be written over and lost forever.

Now, to make matters even worse, after the write operation is finished, the incorrect BAM from memory is written to the disk, forever destroying the correct BAM for that disk. In all future operations, the wrong BAM, now on the disk, will be used to guide any write operations. The data on the disk simply becomes more and more contaminated until it is utterly useless.

For the above reasons, on dual drive systems, the Copy command is much preferred over the Duplicate command. The Duplicate command duplicates everything on the disk, including the ID code, producing two disks with the same ID.

Don't Scratch Open Files

Probably the second most common cause of disk data loss is using the Scratch command to get rid of an open file. The Validate command is what you should use.

It seems quite logical that you would get rid of an open file the same way as you get rid of a closed one. There are very few warnings against it, but scratching a file that has been left open is one of the worst possible things you can do. (You can tell that a file is open by listing a directory of the disk. Files with asterisks next to their names are still open.)

To understand why, we must examine what happens to the Block Allocation Map (BAM) and the next-block pointers when a file is written and when it is scratched. When a file is opened for writing, two unused blocks are located by examining the BAM. The directory entry is written with a pointer to the first of these two blocks. This is where the first data in the file will be written.

The drive controller then begins preparing the contents of the first block in an internal buffer. The first two bytes in this block are a pointer to the second block. Preparation of the first block continues by adding the data that is to be saved to disk. When the buffer is full, the first block is written to the disk.

Then the BAM is searched for the third available block. The controller writes a pointer to the third block into the buffer where it is now preparing the contents of the second block. The pointers and the BAM are always working one block ahead of the current storage location.

If we close the file, the final pointer will be replaced by an end-of-file code. But if we simply interrupt the process, we are left with a pointer pointing to a block that has not actually been used. Hold that thought for a moment while we look at the Scratch process.

When a Scratch is commanded, the *next-block pointers* are traced and the corresponding BAM entries are marked *unused* until the end of file is found. The process is completed by marking the directory entry for that file as deleted. Notice that the Scratch command must find an end-of-file code to finish its process, but a file that has not been closed does not have an end-of-file code. So what happens?

In a common case, the last block in the open file will be pointing at a block that was used at some previous time. Consequently, it will have a pointer, and the next block will also have a pointer. Somewhere down the line, one of those leftover pointers may point into a valid file which has been written more recently. This is where the problems really start.

As soon as the pointers link into a currently valid file, we find the disk controller marking those blocks unused, while the Scratch process continues to hunt for the end of file code. At this point we have an incorrect BAM, but no data has been lost. We lose the data as soon as we write another file to the disk. Seeing all those nice convenient unused blocks, the next writing operation will probably use them. Now we have lost part of an older file. If we were to load that file, we would see that part or all of the newer file appears attached to what is left of the old file.

Depending on exactly how the files are cross-linked and what operations are performed on them next, this problem can continue to grow until many files have been contaminated. Often the problem may not even be detected until a great amount of damage has been done.

If you ever find yourself with cross-linked files, the best course of action is to copy each file to a new disk. There they will at least be linked properly, and the extent of the damage can be assessed without provoking more problems.

Operator Errors

No program can protect against all operator errors, but we have made a sincere attempt in this direction. The DFH programs are very friendly. A good part of this friendliness comes from protecting users from the results of errors they might normally make. Of course, the best protection is careful operation, but none of us is perfect.

The idea behind built-in error protection is to keep the program running and prevent the loss or contamination of data. All of the DFH programs, except the bootstrap program and the DFH Editor, have built-in error protection.

Extensive error protection in the bootstrap program is unnecessary because none of your data or data files can be threatened while the bootstrap is running. For the DFH Editor, error protection of the type we are discussing is simply not appropriate. The DFH Editor is a utility program that must be able to perform according to operator direction regardless of the consequences.

Many of the error-protection features are discussed or illustrated in the chapter on detailed operation. Program responses to some of the most common mistakes are shown in the following list. (Some problems apply only to particular computers.)

1. Typing RETURN without any data during an INPUT operation will not cause the usual exit from program operation. If

a null input is valid for the current situation, the program will accept this action as a null. Otherwise, the input will simply be ignored, and the program will wait for a valid input.

2. Inputting an alphabetic character when a number is needed will not cause an error. This input is treated the same as a 0. If 0 is valid for the current operation, the program proceeds. If not, the input will be requested again.
3. Asking for a disk operation without a disk installed will only cause a disk error message to be displayed. The program will then prompt you through a series of questions/ actions to return to the operation you were trying to perform.
4. If the program cannot find a file or program that you have requested, it will advise you of the situation and request a disk change.
5. Inserting the wrong disk during a disk change will only cause the program to reprompt you for the correct one.
6. If you request data to be saved in a file that already exists, the program will warn you and will require confirmation that you want to replace the existing file.
7. All operations which will delete data will require confirmation from the operator.

In general, you will find it hard to make a mistake from which you cannot recover, but no program is entirely bombproof, so here are some *don'ts*:

1. The RUN/STOP key has been left active, so do not press it during program operation. If it is accidentally pressed, you may be able to recover by typing CONT and pressing RETURN.
2. Do not remove a disk when its drive active light is on (unless you have already bombed the program and have no choice). If you find yourself about to do this in response to a message, reread the message. It is probably asking for a change on the other drive.

The final, and easily the most important, error protection is making backup copies. No program can protect you from ultimate disasters such as power failures or spilled coffee.

Chapter 2

Operating DFH

The Bootstrap

DFH BOOT is a bootstrap routine which provides initial setup and overall executive control for the other DFH programs. It is the only program in the DFH series that you need to know by name.

The easiest way to use any of the DFH programs is to start by loading and running DFH BOOT. Simply insert your program disk in the disk drive and then type:

```
LOAD "DFH BOOT",8
```

Then press the RETURN key, type RUN, and press the RETURN key again.

The first time DFH BOOT is run, after turning on your computer, it displays a brief summary of overall program capability and loads the machine language subroutines DFH SUBS$79. These operations are controlled by a test to see if the subroutines are already in memory. On subsequent runs, when the subroutines are found already in place, the bootstrap will not display the features summary or reload the subroutines.

The bootstrap also determines what type of computer is being used and conditions it as necessary. For all computers, the top of memory is set at $7900 to protect the machine language subroutines. For 80-column PETs, the screen is condensed vertically for graphic display. For 64 computers, the start of BASIC is relocated to $0401 and the screen memory is relocated to $8000 to make it look internally like a PET computer.

Next the main menu for the DFH functions is displayed. From this menu you can select functions without needing to know the name of the program that will perform them. For example, the program DFH SORT performs four functions, three of which are not indicated by its filename.

The main menu presented by the DFH BOOT program will appear as follows:

```
DATA FILE HANDLER FUNCTIONS
 1 CREATE OR EDIT A DATA FILE
 2 LIST (HARD COPY FOR EDITING)
 3 SORT BY RECORD OR FIELD CONTENT
 4 MERGE SORTED FILES
 5 PRINT PER USER-DEFINED FORMAT
 6 SPLIT FILES BY FIELD CONTENT
 7 EXTRACT RECORDS BY FIELD CONTENT
 8 RESTRUCTURE DATA RECORDS
 9 ACTIVATE DFH EDITOR & DOS
10 QUIT
YOUR CHOICE ----- ? 1
```

The main menu shows only the major functions of the DFH programs. When one of them is selected, the appropriate program will be loaded and run. At that time a secondary menu will usually be presented to further determine exactly what task is to be performed. All of the individual DFH programs except the DFH Editor can be directed to return to the master menu when you have finished using them.

Selecting the DFH Editor will cause the editor to be loaded and activated. The top of memory will be set to protect the editor at $9000 (decimal 36864) in the 64 or at $7000 (decimal 28672) in PET computers. With the exception of the changed memory limit and the fact that a command intercept wedge is installed, the computer will be returned to its normal power-on condition.

Activating the DFH Editor leaves you with a computer that appears near normal. Actually, you have a powerful set of Disk Operation Support (DOS) shorthand commands and a wide selection of file-editing commands at your disposal. Detailed descriptions of the DOS and Editor commands are presented later in this book.

Notice the difference between the edit function (menu item 1) and the DFH Editor (menu item 9). Item 1 is used to create and edit the contents of data files under DFH program control. The DFH Editor, along with its DOS functions, is intended to be used as a stand-alone program which can directly access and manipulate sequential data files.

The bootstrap program also contains termination procedures which are designed to return the computer to very near its normal (power-on) state. The exception is that in the Commodore 64 the top of memory is left set at $9000 to protect the DFH Editor just in case it had previously been loaded.

For PET computers, the top of memory is returned to the normal $8000, and for 80-column models, the screen is restored to normal line spacing.

Create, Edit, Sort, and List

The DFH SORT program is called into operation by selecting any one of the first three functions listed in the bootstrap main menu. This program provides a method of creating, editing, sorting, and listing multifield sequential data files.

The DFH SORT program lets you create files with up to 20 fields in each record. The delimiter you select to separate the individual fields can be any character except a number, a space, or a quote. Obviously, the delimiter must not be a character used as data in the file.

All actions needed to set up a sequential data file are accomplished in response to a series of questions asked by the program. A variety of data entry and editing features are included to reduce the manual effort required to enter or edit data, to reduce the chance of data errors, and to insure a uniform and controlled data format.

Sorting can be performed on the complete data records or on any field within the records. Both ascending and descending order sorting are available. Files can be listed in much the same manner as BASIC program files. Line numbers are included in the listings to assist in subsequent editing efforts.

Individual files of up to 700 records or 14,000 characters (about 61 disk blocks) can be created and processed.

Operating DFH SORT

In the discussions that follow, all references to disk drive numbers apply only to PET systems where dual drives are normal. For Commodore 64 systems, the program will not produce such messages.

When you select any of the first three functions from the bootstrap main menu, DFH SORT will be loaded and run automatically, and a short menu of start-up options will be displayed:

DATA ENTRY AND SORTING FUNCTIONS

1 CHANGE DISPLAY/PRINT CASE

2 LOAD DATA FILE FROM DISK

```
5 CREATE A NEW FILE
7 INITIALIZE ANOTHER DISK
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------ ? 1
```

Option 1 allows you to change the case of the screen display and the printer to provide the most useful presentation for the data you are processing. The printer will always print in the same case as the screen.

Option 7 allows the installation of another disk anytime you wish. This option releases you from any requirement to have your data files arranged in a particular order before running the program.

The primary purpose of this first menu is to select whether you want to use option 2 to load an existing data file from disk for processing, or use option 5 to create a completely new data file.

Loading a File

Let's assume that you have a data file named TEST on the disk in drive 0 and wish to load it. Menu option 2 will result in a sequence similar to the following:

```
INPUT FROM DRIVE #   ? 0
DATA FILENAME        ? TEST

52 DATA RECORDS LOADED.
   ( 6 DISK BLOCKS)

 7 FIELDS PER DATA RECORD.
"!" IS THE FIELD DELIMITER.

PRESS ANY KEY TO CONTINUE
```

As the data file is being loaded, the count of *data records loaded* is continuously updated on the screen.

When loading is complete, the program displays the number of blocks the file occupied on the disk, the number of fields per record, and the delimiter used in the file. The delimiter is the first character in the first line of the data file.

The program determines the number of fields per record by counting the number of delimiters in the first data record (second line) of the file. If you are processing a file which was not created by one of the DFH programs and which does not have a uniform number of fields per record, you must be sure that the first data record contains as many fields as any record in the file. (The general procedures for converting files which were not prepared by a DFH program are covered in detail in Chapter 7, "File Conversion.")

With the file loaded, the program waits for you to PRESS ANY KEY TO CONTINUE. This allows time to review the results of the loading operation before proceeding to the next step.

Once you press a key, the program will display the complete options menu. This is an expanded version of the menu we just used, and we'll examine it later. Right now, let's back up and see what would have happened if we had decided to create a completely new file. After all, that may be the first thing you will need to do.

Creating a New File

When you select option 5 from the start-up menu, the program will first need some basic information about the structure of the file and will then begin accepting data:

```
# FIELDS PER RECORD      ? 2
DELIMITER TO BE USED     ? !

A ADD F FINISHED ? A

LINE# 1010, FIELD 1
"(field #1 data)

LINE# 1010, FIELD 2
"(field #2 data)

A ADD    D DELETE
E EDIT   F FINISHED ? A
```

You can specify any number of fields per data record from 1 to 20. Numbers outside this range will not be accepted and the question will be repeated.

The delimiter character you choose must be one that will not be used in any of the data in the file. Beyond that, the only restriction is that the delimiter cannot be a number or a quote. The program will not accept an illegal delimiter.

While you are entering data, the program will not allow you to enter your selected delimiter as a data character. A little later in this section, when we discuss sorting, we will explore some additional considerations in selecting a delimiter.

Since there is no data in your new file at this point, your first action is limited to only two choices: ADD a new line of data or indicate you are FINISHED.

After you enter data for the first line, your choices from the next action display will also allow you to EDIT or DELETE. Let's look at each of these four selections.

Add a New Line

The program will guide you through the new line data entry process by displays that show which record and field are being entered. The contents of the same field in the previous record are displayed as an operator input prompt. If this data is to be repeated in the current record, you can simply hit the RETURN. If most of the characters in the data field are to be repeated, the normal onscreen editing procedures can be used prior to RETURN. Both of these techniques can save a lot of time and improve the accuracy of data entry.

All data used for input prompting is preceded by a quote. This leading quote is needed if leading spaces, shifted characters, or special function characters are to be contained in the data. You may either retype the leading quote or cursor past it.

The only time a trailing quote is required is when trailing spaces are being entered. Otherwise, the trailing quote is acceptable but not required. If the data being entered contains only unshifted alphanumeric characters, you may choose to simply type over the leading quote.

Warning: If you use a leading quote and then embed a quote within the data, the program will probably quit and you will not be able to restart it without loss of all data in memory. We simply could not find a way to protect against this operator error.

Editing a Record

When you select the EDIT function, you will be asked which line you want to edit. The data records are always numbered beginning with 1010 and proceeding in increments of ten. If you request a line which does not exist, the program prints a warning message and asks for a new line number:

```
A ADD  D DELETE
E  EDIT  F  FINISHED ? E
EDIT LINE # ? 0520
          OUT OF RANGE
EDIT LINE # ? 1010
LINE# 1010, FIELD 1
" (field 1 data)
```

The first time you request the EDIT function, the default prompting will be for the last line in the file. Thereafter, the default prompting will be for the line following the one you most recently edited or deleted.

If your request is out of range and too low, the program will return a default prompt to the first line in the file. If your request is out of range and too high, the program will return a default prompt to the last line in the file.

If your request is for a line you have already deleted, the program will tell you the line is ALREADY DELETED and will present a default prompt for the next line in the file.

Once you select a valid line to edit, the process is almost like a new line entry. The only difference is that the default data prompts will be from the line you are editing instead of from the previous line.

When you finish editing a line, the default prompting (if used) will keep you in EDIT mode and will proceed to the next higher numbered record for editing.

Delete a Record

The third option is DELETE. When you request the DELETE function, you will be asked for a record number. The out-of-range warnings and reprompting work the same as for EDIT.

```
A ADD  D DELETE
E  EDIT  F  FINISHED ? D
DELETE LINE #    ? 7050
                 OUT OF RANGE
DELETE LINE #    ? 1240
1240 "E01!68-01-15!962!21.55!!!
ARE YOU SURE     ? Y
               DELETED
```

The requested record is displayed with its line number, leading quote, and all the delimiters exactly as it would appear in a listing of the file. To reduce the chance of accidental deletions, you will be asked ARE YOU SURE ? before the record is deleted. For this question the default prompt will always be N.

Because the default prompting causes the records to be displayed without deleting them, this function can be used to review the contents of a series of records. You simply continue hitting the RETURN key and look at the records as they are displayed.

Finished—Return to Menu

When you have completed all the data entry and editing you wish to do, simply select FINISHED and the program will return to the functions menu for your next selection. You might wish to do this quite often to store your newly entered data to disk.

If you have deleted any records, a special message will be displayed when you decide to return to the functions menu:

DUE TO DELETIONS, THE FILE IS NOW BEING
SORTED ON FIELD #0 IN ASCENDING ORDER.
-- YOU MAY RE-SORT AS DESIRED --

This automatic sorting operation is done to gather all the records into a compact group to insure that no null records are saved with the file.

Since sorting is a desired function in the great majority of computer-based data files, this automatic sorting step will usually be an acceptable procedure. However, there will be some exceptions where it is desired that the file be maintained in a different sort order.

For those cases, you should create one extra field in the data records. This field will contain sequential line numbers or other alphanumeric codes which, when sorted, will produce the desired order for the data records.

When the program displays the functions menu, you will see that it has been expanded and now contains some new items that were not meaningful until you created or loaded a file:

DATA ENTRY AND SORTING FUNCTIONS

1 CHANGE DISPLAY/PRINT CASE

```
2 LOAD DATA FILE FROM DISK
3 SORT THE FILE
4 SAVE THE FILE
5 ADD, EDIT, OR DELETE RECORDS
6 LIST THE FILE
7 INITIALIZE ANOTHER DISK
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------ ? 1
```

Also, note that option 5 is now for adding or editing records, not for creating a new file. Except for the new file start-up questions, option 5 works exactly like the earlier version of option 5 which we have just discussed.

Sorting the Data

One of the most valuable features of this program is its ability to selectively and very rapidly sort the data in a file. Maximum sorting time for large data files is usually under ten seconds. Sorting can be in ascending or descending order and can be based on the complete data records or on any individual data field.

The following is a typical sequence which could occur when we select the sorting option:

```
FIELD TO BE SORTED ? 4
  A   ASCENDING OR
  D   DESCENDING ORDER ? A
  352 TOTAL DATA RECORDS
  324 DATA RECORDS SORTED
  28  RECORDS WITH NULL IN FIELD 4
  PRESS ANY KEY TO CONTINUE
```

In this example we asked for a sort in ascending order on the data in field 4 of each record. The program found that there were 352 records in the file, but only 324 of them contained data in field 4. The remaining 28 records are retained in the data file, but they are in random order and not immediately available for editing or listing because there was no basis on which they could be sorted.

The data fields are numbered beginning with 1. This allows the number 0 to be used for a special purpose. When field 0 is specified, the entire data record is sorted. Since all characters, including the delimiters, are used in a field 0 sort, there is no possibility of a null result and all records will be sorted.

Sorting on field 0 can require special consideration in the choice of a delimiter. As an illustration, the following two results could be obtained by sorting identical data with different delimiters (! and >) in ascending order:

```
2!   200>
20!  20>
200! 2>
```

Because the first delimiter, !, has a character code value which is less than any numeric character, we can see that 2! is less than 20 and that 20! is less than 200. (The sorting evaluation takes place character by character from left to right.) Likewise, we see that 200 is less than 20> and 20 is less than 2> because the > has a character code value greater than any numeric character.

If you find that you need to change the delimiter used in a file, you can do it easily by the DFH SWAP program. That program is called by selecting the restructure option from the bootstrap main menu. Alternate methods are also available through the use of the DFH Editor.

Saving the File

When you have finished creating or editing a file, you will want to save it on a disk. The program will guide you through this process, explaining your options (and there can be some interesting ones) as you go. Let's look at the simplest case first:

```
ORIGINAL FILENAME   ? TEST
NEW FILENAME        ? TEST
OUTPUT TO DRIVE #   ? 0

325 BLOCKS FREE
REPLACE EXISTING FILE ? Y
SAVING -- PLEASE WAIT --

PRESS ANY KEY TO CONTINUE
```

The program first displays a reminder of what file (if any) was originally loaded and asks for the filename to use for saving. We choose the same name and are then asked for a drive number.

The program indicates how many free blocks are on the disk, and would insist we change disks if there are not enough blocks free. Because the file already exists, we are asked about replacing it, and finally the file is saved. If we choose not to replace the file, we have a chance to install a new disk and continue or return to the functions menu.

A more interesting situation is presented when the file has been sorted on a data field which contains some nulls. In that case the conversation would start with a display similar to this:

```
1 SAVE COMPLETE FILE
2 SAVE ONLY THE 24 RECORDS
  WITH DATA IN FIELD 3
YOUR CHOICE ----- ? 2
```

This display indicates that the file has been sorted on field 3 and that only 24 of the records have data in that field (implying that there are additional records in the file with nulls in field 3).

Save the Complete File

If you decide to save the complete file, an advisory message will be displayed as follows:

```
FILE WILL BE ERASED FROM MEMORY
DURING THIS SAVE. PRESS M FOR
ANOTHER MENU SELECTION, OR --
PRESS ANY KEY TO CONTINUE
```

In this situation, due to the nature of the sorting subroutine, the program does not know the exact locations of the records which had nulls in the sorted field. Thus the sorted records will be saved first, and then the remaining records will be sorted on field 0 and saved.

To prevent their being saved twice, the original sorted records must be deleted from memory as they are sent to the disk so that the field 0 sort can be done on the remaining records. If you wish to continue working on the same file after this type of save, you must reload the file.

The other choice is to save only the sorted records. When this is done, the entire file will still be contained in memory after the save. With this feature we see that the sorting/saving process can be used to isolate and save selected parts of a file based on the null or not-null condition of any data field.

Here's an example of the use of a partial save. Let's say your company makes a number of products which use a lot of the same parts but in different quantities. You get a request for one of the products and must order the required parts. You can extract data for the order list by using a file where each record represents an individual part with field 1 containing the part number, and each following field containing the quantity of that part which is used in each of the products.

PN 5425-A*2*9**
PN 5681-A*1*5*7*
PN 6004-D*14***

In this example the product represented by field 3 uses nine of the first part number, five of the second part number, and none of the third part number. A sort on field 3 followed by a partial save will create a file that contains only the records for the parts needed to build that product.

When we discuss the DFH printing functions, you'll see how to produce the parts order list by selectively printing only the desired fields, which in this example would be the part numbers from field 1 and the quantities from field 3.

Listing the File

When preparing to edit a file, it is often easier to have a printed copy of the file exactly as it appears in memory. The listing option is used for this purpose.

The file listing will include line numbers. These numbers are not a part of the file data. They are assigned to the records as the file is being loaded or created to provide a method of identifying the individual records.

The list produced on the printer will be in the same case as the screen display (lowercase and uppercase, or uppercase and graphics). You can change the case at any time by selecting option 1 in the functions menu.

When listing begins, the program will display

PRESS ANY KEY TO PAUSE, THEN --
Q TO QUIT LISTING OR ANY OTHER
KEY TO CONTINUE.

This feature allows you complete freedom to pause or stop the listing at any time you wish. For PET users, a momentary touch of a key will be sensed, but on the Commodore 64 the key must be held down until the listing stops.

Maximum File Size

There is obviously a limit to how much data can be contained in memory. Consequently, there must be some file-size restrictions. Since this program, unlike others in the DFH series, must contain a complete file in memory, it dictates the maximum individual file size.

The DFH SORT program can load or create files of up to 14,000 characters (about 61 disk blocks) or 700 data records, whichever occurs first.

The merging program will encourage the creation of individual files under 13,000 characters (about 57 disk blocks) or 650 records. This provision can be overridden to create large files for special purposes, but the intent is to allow some space for additions and editing by DFH SORT.

Printing

The DFH PRINT program provides an efficient and flexible method of printing the data contained in multifield sequential data files.

The major features of the DFH PRINT program are:

- Multifile and multidisk linking.
- Page headings with resettable page numbers.
- Page length and positioning control.
- Printing case control (UC/GR or LC/UC).
- Individually justified field headings.
- Print all fields or only selected fields.
- Reorder field positions during printing.
- Left/right justification for each field.
- Page images saved as *WordPro* files on demand.
- *WordPro* files linked for global operations.
- Printing format specification saved on disk.
- Self-guiding operation.

There are too many options and variations in this program to describe them all. However, this is a very user-friendly program and it will easily guide you through any variations you may want to use. The program is designed so that you can easily recover from almost any operating error you might make. A typical program operation sequence will be used to illustrate the major features and general flow of the program.

Operating the Program

The DFH PRINT program is activated by selecting the printing function in the bootstrap main menu.

The first display is a short set of operating notes. These notes will not be repeated, so until you become comfortable with the program, you might want to keep a written copy handy.

All references to disk drive numbers apply only to PET systems where dual drives are normal. For Commodore 64 systems, the program will not produce such messages.

```
----------------------N O T E S----------------------
ALL SOURCE DATA AND PRINT FORMAT FILES
WILL BE IN DRIVE #0.
```

```
ANY WORDPRO OUTPUT FILES CREATED WILL
BE SAVED ON DRIVE #1.
OUTPUT OPERATIONS CAN BE:
FROZEN BY PRESSING ANY KEY
-THEN-
ABORTED BY PRESSING Q
 -OR-
RESUMED BY PRESSING ANY OTHER KEY.
SET PRINTER TO TOP-OF-FORM AND
PRESS ANY KEY TO CONTINUE.
```

The program will maintain paper-position control during all operations following this step. However, there will be several opportunities for paper-position adjustment in case you forget and move the paper by hand during program operation.

When you press a key to continue, the program displays the first of four menus that will be used to guide you in producing the output you want:

```
FORMAT SOURCES
1 CHANGE SCREEN/PRINTER CASE
2 LOAD FORMAT FILE FROM DISK
3 DEFINE THE PRINTING FORMAT

9 QUIT OR GO TO MASTER MENU
YOUR CHOICE -----  ? 3
```

The options to quit (option 9) and to change the case of the screen and printer (option 1) are presented in all four menus so you can switch case or quit whenever you want. The case of the output to the printer and the *WordPro* files will always be the same as the current case of the screen.

Selecting option 9 will cause an orderly shutdown, closing all open disk files, etc. To complete the orderly shutdown and restore the computer to normal configuration, you should return to the main menu and "quit" from there. The quit option in each of the individual DFH programs is for the convenience of programmers who may want to examine the code while it is in an operational condition.

The primary purpose of this first menu is to select a method of defining the printing format. You can either load a format that you defined and saved during some previous operation of this program, or you can define a completely new print format. If you load a print format file, you will be given opportunities to modify it later in the program. This can be a real timesaver for similar formats.

Defining the Print Format

If you want to define a new print format, the program will guide you through the process, working down from the top of the page and left to right across the print fields. Such a conversation might start as follows:

```
DEFINE THE PRINTING FORMAT
# BLANK LINES ABOVE HEADING ? 3
ENTER PAGE HEADING LINE
USE TWO ENTRY LINES TO FORM A
COMPLETE PAGE HEADING LINE.
USE '<>' TO SHOW PAGE NUMBER LOCATION
DON'T DISTURB THE QUOTE OR THE END OF
LINE MARKER.
" (left half of page heading)
" (right half of page heading)
STARTING PAGE # ? 1
# BLANK LINES BELOW HEADING ? 1
# OF PRINT FIELDS ? 2
```

As shown here, you can specify the number of blank lines to be printed on each page above the page heading line. The technique of entering the page heading in two parts allows the program to be used with equal success on computers with either 40- or 80-column displays. The end of each page heading entry line is marked by what appears to be a reverse field space character. This marker character (not shown here) allows trailing spaces to be preserved without a closing quote and also indicates the proper right-hand limit for characters being typed in.

You can specify the number of blank lines below the page heading to separate it from the individual field headings.

Your entry for the number of print fields is used by the program to provide default prompting as the individual print field formats are defined. You can override the effect of this input if you change your mind while defining the print field formats.

By defining the individual print field formats, you can print the data in any order you wish, independent of the order of the data fields. The difference between a *data field* and a *print field* is important in understanding this process. A data field is a part of a data record, and it occupies a fixed position relative to the other fields in the record. The data in a print field is a part of a printed line, but it does not have a fixed position until the line is printed.

With this program, you have total control of what data fields get printed, where they get printed, and the order in which they are printed. The data from any data field can be printed in any print field.

Let's continue the conversation with the computer and examine the options available in defining a print field.

```
FOR PRINT FIELD # 1
# SPACES AHEAD OF FIELD ? 0
PRINT DATA FIELD # ? 1
LEFT OR RIGHT JUSTIFIED ? L
# OF COLUMNS IN FIELD ? 9
FIELD HEADING
" head 1 "

NOW AT COLUMN # 9
MORE FIELDS ? Y
```

In defining a print field you first specify the number of spaces which will be printed to the left of the print field. For the first print field, these spaces establish the left margin for the printed data. For all other fields, these spaces provide a uniform and guaranteed open area between successive print fields.

Next you select which data field you want printed in the print field you are defining, indicate whether you want the data to be left- or right-justified within the print field, and specify how many columns (character spaces) are to be used in the print field.

Finally, you enter the field heading. The field heading will be justified left or right, the same as the data in that field. For added control of heading placement, leading and trailing spaces can be included. To maintain these spaces during revision cycles, the previously entered field headings will be used for default prompting and will always be displayed with leading and trailing quotes.

If you specify more than 38 columns in a field, you will be allowed two lines of 38 characters each for the field heading. Again, this is to allow the program to run on both 40- and 80-column computers.

If you enter a field heading that contains more characters than you asked for in the print field, a warning message will tell you how many extra characters there are, and the program will ask you to redefine the format for that print field. To correct the problem, you can change either the number of columns in the print field or the number of characters in the field heading.

As each print field is defined, the program shows the total number of columns which have been used and asks if more fields are to be defined.

When you indicate that all print fields have been defined, the program will proceed as follows:

```
#BLANK LINES ABOVE DATA   ? 1
DATA LINES/PAGE (MAX 54 )   ? 54
```

In this sequence you specify the number of blank lines between the field headings and the first printed data line. Now, having all the heading information, the program calculates the maximum number of data lines that can be printed on each page. This value is displayed and also used as an input prompt. You may request any number of lines up to the maximum. Higher numbers will simply not be accepted.

Setup Options

With the print format now completely defined, the second menu will be presented.

```
S E T U P   O P T I O N S
1 CHANGE SCREEN/PRINTER CASE
2 LOAD FORMAT FILE FROM DISK
3 MODIFY THE PRINTING FORMAT
4 TEST HEADINGS TO SCREEN
5 TEST HEADINGS TO PRINTER
6 SAVE PRINT FORMAT FILE
7 OUTPUT OPTIONS
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE -----  ? 6
```

An important function of this menu is to give you an opportunity to change your mind about anything you have done up to this point. By using this menu you can examine all the headings by printing them either to the screen (option 4) or to the printer (option 5) to make sure they are correct.

If you don't like what you see, you can either modify the format (option 3) or load a completely new format (option 2). If you modify the format, all your previous entries will be used as default prompts so that you can simply hit RETURN for items that don't need changing.

This entire sequence of test printing and modifying can be repeated as many times as necessary to get the headings just right. If you are printing the headings to the printer, the program will conserve paper by suppressing the paging feature. However, all printed lines are counted so that the program can automatically reestablish the paper position to top of page when you are ready to continue.

When you have the print format defined the way you want it, you can save it to disk for later use (option 6). The conversation following your decision to save the format might appear as follows. (To illustrate an error-protection feature, let's make some mistakes this time.)

```
READY TO SAVE PRINT FORMAT FILE
DELIMITER TO BE USED ? 7
ILLEGAL DELIMITER 7
DELIMITER TO BE USED ? E
CHARACTER IS USED IN HEADINGS
DELIMITER TO BE USED ? !
FILENAME ? FM-TEST
REPLACE EXISTING FILE ? Y
```

Our first error was choosing an illegal delimiter—numbers are not allowed as delimiters. We then tried an E, which had already been used in one of the headings and also produced an error message. The program accepted the ! as a delimiter.

Notice that the delimiter used in the format file has no relation to the delimiter used in any of your data files. It is selected independently and is used to separate the data fields within the format file records.

The choice of filename was not an error, although it might have been, depending on our intentions. We simply selected the name of a file that already existed, and in the next line, confirmed that we wanted the new print format file to replace it. If we had chosen not to replace the file, we would have been given an opportunity to install a different disk and then return to the setup options menu.

Output Options

With the printing format checked, you are ready to select option 7 and go to the output options menu to specify the output process:

```
OUTPUT OPTIONS
1 CHANGE SCREEN/PRINTER CASE
2 SCREEN OUTPUT ONLY
3 PRINTER OUTPUT ONLY
4 WORDPRO FILES ONLY
5 WORDPRO AND PRINTER
6 WORDPRO AND SCREEN
7 RETURN TO SETUP OPTIONS
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------------- ? 5
```

If you have an 80-column screen, you might want to try option 2 to send some completely formatted data to the screen to see how it looks. The output can be stopped at any time (press Q to stop, any other key to pause, and another key to unpause). With a 40-column screen you may have more success doing this test with the printer option.

In either case, the program will ask for the name of the file you want printed, and will send the formatted output to the device you select. Once you like the way the output looks, simply select the option you want, tell the program what file to print, and wait for completion.

To illustrate as many features as possible, let's look at what happens when we ask for a combined *WordPro* and printer output (option 5). This selection will create *WordPro* files that are exact images of the printed output. Our conversation with the program might be:

```
IS PRINTER AT TOP OF FORM ? Y
FOR WORDPRO FILES
FILENAME ? WP-

409 BLOCKS FREE ON DRIVE #1
FOR WORDPRO FILE WP-.001
```

The first question in this conversation provides an opportunity to readjust the position of the printer paper in case you had moved it by hand. If you answer no to this question, you will be asked to adjust the paper and tell the program when you are ready to go again.

Next you need to tell the program what name you want to use for the *WordPro* files. The program will modify any name you give it by adding a period and a three-digit number to create the filenames actually used to save the *WordPro* files. There will be as many *WordPro* files as there are pages of printed output.

When printed under *WordPro* control, each of the *WordPro* files will produce an exact image of the corresponding page printed by the DFH PRINT program. The first file will contain *WordPro* commands to set the paper size, number of printed lines per page, and the left and right margins. Each file will begin with a comment line containing the name of the file. Inserted at the end of each file (except the last one) are a *force page* command to insure proper paging and a *next file* linkage command to allow global printing.

After you enter the filename, the program will check to see if there is enough room on the disk to save the first file. If there is not enough free space, you will be asked to install a different disk in drive 1. This checking process will be done for each *WordPro* file that is created.

If you are using a Commodore 64, the basics of the space-checking procedure are still followed, but an impossible situation is reached if you run out of space on the disk. You can't change disks without removing the input data file, so you simply *must* have enough space for all the *WordPro* files or you can't complete the task.

Disk space will not be a problem if you start out with a moderately empty disk. You can use DFH SORT (items 1–3 on the bootstrap main menu) to move a file to a new disk. For worst-case estimating, assume that each printed page will require 19 disk blocks for *WordPro* file storage.

If a file with the same name as a new *WordPro* file is found at any time during the output process, the program will ask if you wish to replace it with the new file. If you do not want to replace the file, you must install a different disk (except on Commodore 64, where you either replace or quit). The program will guide you through this process.

Some minor confusion may result if you create a series of *WordPro* files which overlap into a longer series of files with the same name. Because of the next-file linkage lines, a *WordPro* global printing will end correctly at the last of the new files and a global copy will also end the copy at the correct point, but you will not be able to identify the last of the new files by examination of the disk directory. It is much better to use a different name or scratch all the old files before creating the new ones.

With the output setup now completed, the program requests the name of the first data file to be printed:

```
READY FOR FIRST DATA FILE
FILENAME ? EX68
END OF SOURCE FILE
PRESS ANY KEY TO CONTINUE
```

Continue Options

When printing and *WordPro* file creation have proceeded to the end of the first data file and you indicate you are ready, the continue options menu will be displayed:

```
CONTINUE OPTIONS
1 CHANGE SCREEN/PRINTER CASE
2 CONTINUE—NO PAGE BREAK
3 CONTINUE—AT TOP OF PAGE
4 CHANGE PRINTING SETUP
5 PRINT OPERATIONS SUMMARY
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------------ ? 2
```

As you can see, there are several ways to continue. Probably the most often used of these is option 2, which links in the next data file without any indication in the output as to where the transition took place.

Option 3 begins printing the next file at the top of the next page. The same print format is used, but you can select the page number of the next page in case you do not wish to continue numbering in sequence.

If you want to change the printing setup (option 4), the program will close any open *WordPro* files, eject the printer paper to the top of the next page, and return you to the setup options menu, where you can take any action desired to prepare for added printing.

Printing an operations summary (option 5) can be done at any time. It will interrupt the printing like option 4 by closing any open *WordPro* files and ejecting paper to the top of the next page. When the operations summary has been printed, the program will return to the setup options menu.

The operations summary provides a complete record of how the file(s) were printed and shows if there was any data that would not fit in the defined print fields. The number of any such field overruns, and the largest number of excess characters are listed for each print field.

Chapter 3

File Manipulation

Merging Files

The DFH MERGE program allows you to merge presorted multifield data file sets of nearly any size. The files may be large or small, on one or many disks. A maximum of 50 files can be merged in a single operation. Up to 99 files can be produced in a single operation. An individual file can be as large as the capacity of a single disk.

Why Merge?

Have you ever started a project involving data storage and found you could not complete it because not all the data would fit in memory at one time? Or maybe you got past that problem and then gave up when the data would no longer fit on one (or two) disks.

If you are working with data that requires sorting, merging is necessary when all the data cannot be contained in the computer's memory. As the data base size grows and you create more and more files, another limit is reached when all the files can no longer be stored on a single disk.

The following example shows three data records of a file named "idfile" which uses the * as a delimiter to separate the three fields in each record:

```
1000 "*←@0:idfile
1010 "Brown*Jack*45721*
1020 "Jones*John*15113*
1030 "Smith*Susan*23442*
```

The line numbers are not a part of the data file and are not stored on disk. They are added where needed for referencing purposes by the DFH programs.

To visualize the concept of merging, assume that the left column of the following illustration represents records from one data file and the right column represents records from another file. Both files have been sorted so that the names are in alphabetical order. Merging produces a file containing both sets of data in the order we would get if we simply pushed the two columns of the illustration together from left and right. It is a single set of records still in alphabetical order.

```
"Brown*Jack*45721*
                        "Edwards*Sam*22705*
                        "Gray*Donna*70442*
"Jones*John*15113*
"Smith*Susan*23442*
                        "Wilson*Alan*10046*
```

File Size Management

The DFH MERGE program will merge from 1 to 50 files which are stored on from 1 to 50 disks. Any number of output files up to 99 can be created by the merging process.

The program will recommend the minimum number of output files that should be requested. This number is based on input file sizes obtained by the machine language Spool subroutine as the files are located. During output, the program counts both characters and records as they are written to disk and closes the file when either count reaches its precalculated limit.

If you follow the recommendations of the program, the size of the merged files will be limited to 650 records or 13,000 characters (about 57 disk blocks), whichever occurs first. These limits insure that the merged files can be handled by programs like DFH SORT which must load a complete file into memory. It also provides a reasonable amount of space for editing and additions to the files before the maximum capacity of DFH SORT is reached.

If you ask for less than the recommended minimum number of files, the program displays a warning message and asks you to confirm your request. Thus, the program allows you complete freedom to create large individual files, but issues warnings when it appears that you may be creating a problem for yourself.

Running the merge program on a single input file is one way of breaking a large file into approximately uniform-size, smaller files. This, or some equivalent method of reducing individual file size, is necessary if you have files that are too large to load into the computer.

Of course, it is also possible to create a few large files from many smaller files. You may even wish to create a single, very large file to be used for special purposes such as long, unattended printing operations.

Operating the Program

Once the MERGE function is called from the bootstrap master menu, it is completely self-guiding in its operation. Consequently, a single example will be used to illustrate the general procedure. The sample problem will involve merging three short files that are located on two different disks. The conversation between program and operator might proceed as follows. (All references to disk drive numbers apply only to PET systems where dual drives are normal.)

```
READY TO MERGE FILES

HOW WERE THE SOURCE FILES SORTED ?

SORTED ON FIELD # ? 1

A ASCENDING OR
D DESCENDING ORDER ? A
```

It is important for each of the input files to have been sorted using the method specified for the merging. If the files are not properly sorted, the output files will not be merged in the correct order. If you have any doubt about the condition of the input files, you should re-sort them before merging.

```
ENTER NAMES OF UP TO 50 SOURCE FILES

NAME OF SOURCE FILE # 1 ? EX68
ANY MORE FILES ? Y

NAME OF SOURCE FILE # 2 ? EX70
ANY MORE FILES ? Y

NAME OF SOURCE FILE # 3 ? EX69
ANY MORE FILES ? N

ALL FILENAMES OK ? Y
```

Up to this point you have specified how the input files were sorted, entered the names of the input files to be merged, and indicated that all filenames are okay.

If you had answered that the filenames were not okay, you would have been asked:

```
REDEFINE THE MERGE ? Y
```

A yes answer would allow you to correct any errors, while a no answer would allow you to quit or return to the master menu.

The conversation continues:

```
READY TO LOCATE FILES & CHECK SIZES
```

```
SOURCE FILES IN DRIVE ? 0
OUTPUT FILES TO DRIVE ? 1
INSTALL SOURCE DISK #1
IN DRIVE #0
THEN, PRESS ANY KEY TO CONTINUE
FOUND EX68 6  BLOCKS
FOUND EX70 10 BLOCKS
STILL LOOKING FOR:
EX69
INSTALL SOURCE DISK #2
IN DRIVE #0
THEN, PRESS ANY KEY TO CONTINUE
FOUND EX69 8 BLOCKS
ALL FILES LOCATED
```

As shown above, the program first asks where you intended to install the source and destination disks.

Next you are asked to install a disk that the program will later refer to as *input disk 1.* This arbitrary identification simply provides a way for you to know what disk to install when the program asks for that disk later during the merging. If you wish, you may mark the disk label, but don't worry about later inserting the wrong disk. At that time, the program will continue to prompt you until you get it right.

When you indicate that the input disk is installed, the directory is searched for any of the previously named input files. Two of them are found, but EX69 is not located, so the program asks for *input disk* #2 to be inserted so that the search can continue.

Getting the correct disk installed is important during this setup phase because the program is also testing to see that none of the named files appears on more than one disk. If a duplicate is found, you must correct the problem and redefine the merge process.

As the input files are located on the disk, the machine language Spool subroutine is used to find the number of records and data characters in each file. This information is used for internal memory and file-size management during the merge. A summary is also presented for your information:

```
MERGE INFORMATION SUMMARY
3 TOTAL DATA FILES
216 TOTAL RECORDS
23 TOTAL BLOCKS
```

```
READY TO DEFINE OUTPUT FILES
1 OUTPUT FILES ARE SUGGESTED.
HOW MANY DO YOU WANT  ? 2
A SEQUENCE NUMBER WILL BE ADDED TO
EACH OUTPUT FILENAME. WHAT NAME DO
YOU WANT TO USE  ? TEST
```

As shown above, you can choose to create more output files than are actually needed.

In this example, the general name for the output files will be TEST. The program will add a two-digit sequence number to this name for each file created. Thus the two output files will be named TEST.01 and TEST.02.

The size of these two files will be approximately the same, but significant variations are possible depending on how uniform the record lengths are. In extreme cases it is possible to produce one more output file than the number requested.

Now the data is to be loaded:

```
READY TO LOAD INITIAL FILE SEGMENTS
73 RECORDS FROM EX69
INSTALL INPUT DISK #1
IN DRIVE #0
THEN, PRESS ANY KEY TO CONTINUE
52 RECORDS FROM EX68
91 RECORDS FROM EX70
```

When the initial loading sequence begins, the program examines the disk currently in the source drive, looking for the first of the source files, and loads from it if it is found. The program will request whatever disk it needs to continue the loading.

For large files, this sequence would have loaded only the first part of each of the input files, but in this example the files were small enough to be completely loaded during the initial pass. Here, the number of records loaded from each file is displayed in reverse video to indicate that all records in the files have been loaded. If only a partial loading had been done, the number of processed records would have been displayed in normal video.

When all of the initial file segments have been loaded, they are sorted in preparation for the first merge output. The sorting takes about seven seconds and is relatively independent of file size. Sorting activity is indicated by a flashing reverse video character.

The remainder of this example is

```
INSTALL OUTPUT DISK #1
IN DRIVE #1
THEN, PRESS ANY KEY TO CONTINUE
161 BLOCKS FREE FOR TEST2.01
122 RECORDS OUT TO TEST2.01
148 BLOCKS FREE FOR TEST2.02
94 RECORDS OUT TO TEST2.02
MERGE COMPLETED
MORE FILES TO MERGE ? N
PRESS Q TO QUIT OR ---
ANY OTHER KEY FOR MASTER MENU
```

In this example, the output disk was not blank. In fact, it had only 161 blocks free when the first output file was opened, and 148 blocks free when the second output file was opened. If there had not been enough free blocks for any output file, the user would have been asked to INSTALL OUTPUT DISK #2. The newly installed disk would also have been checked for free blocks.

When the merge is complete, you are asked if there are more files to merge. A yes answer would return you immediately to the setup process, avoiding a round trip to the master menu.

Finally, you are asked if you want to quit immediately or return to the master menu for another function selection. If you quit immediately, it will leave your computer with an unusual memory configuration, so this exit should be used only when you intend to power down anyway. Its real purpose is for the programmer who might want to quit, make a program examination or modification, and then rerun or resave the program.

Hints for Merging

This program allows you freedom to solve your data-merging problems in the way you think best. However, this freedom can be misused to your own disadvantage.

For example, we recently tried two methods of merging a data set containing over 25,000 records. The data was in 48 files on three nearly full disks (over 1600 disk blocks). One approach completed the task in less than two hours. We abandoned the other approach when our timing estimates showed that more than 26 hours would be required to complete the job.

We offer the following suggestions for merging very large data sets (more than one full disk) and especially those contained in a large number of files.

If there are some small individual files, they should first be merged into larger files. The merging runs faster as the number of files decreases, and small files don't take long to merge. The time you spend to merge the small files will be more than recovered during later merging because of the smaller number of files involved at that time.

Merging runs faster, and with less work, when you don't have to waste time changing disks, so you may want to arrange a series of single-disk merges rather than trying to do the entire job in one pass. Moving files from disk to disk is quite simple with a dual disk drive. But even with a single drive you can use the DFH SORT program to move moderate-size files.

Here is an example of the type of file moves that could help: If you have two disks of data, you could merge all the files on each disk into four files. With a dual disk drive, these merges can run unattended once you get them started. Next, move the files so that the first two files from each merge are on one disk, and the last two files from each merge are on another disk. With this arrangement, the number of disk changes during the final merge will be greatly reduced.

Don't be concerned that the files you create during intermediate merges might be very large. They can be split into as many files as you want during the final merge.

As you work with large files you will surely develop your own techniques for saving time. These suggestions are intended to get you started. Merging files is a time-consuming operation, so it is important to spend a few minutes prethinking the process in order to save execution time.

Restructuring Files

The DFH SWAP program provides the capability of uniformly restructuring the data fields within a multifield data file.

The ordering of the fields can be changed. New fields can be added at any place in the records. The data in existing fields can be concatenated into new fields. New constant data can be added (either leading or trailing) to an existing data field.

All of these operations are available for each field you wish to define in a new file. Sound like a lot? It is, and yet it is very simple once you see the reasons for restructuring and the logic behind the various options.

It is not unusual to decide, somewhere in the middle of a large data-handling project, that the data structure you chose for the data files was not really the best one. Or, worse yet, maybe it won't even work. Or perhaps you just need the same data organized in more than one way. DFH SWAP was written to get you out of such difficulties.

The program allows five types of reorganization to be performed on the data fields of multifield sequential data files. Each of these types will be discussed as we examine the operation of the program.

Program Operation

This program is called into action by selecting the restructure function in the bootstrap master menu. The operation begins with a program/operator conversation similar to the following. (Remember, all references to disk drive numbers apply only to PET systems where dual drives are normal.)

```
READY TO RESTRUCTURE DATA RECORDS
CHANGE DISPLAY CASE ? N
SOURCE FILE IN DRIVE ? 0
SOURCE FILENAME ? DATA1
OUTPUT FILE TO DRIVE ? 1
OUTPUT FILENAME ? NEWDAT
343 BLOCKS FREE
REPLACE EXISTING FILE ? Y
```

The first question allows you to change the case of the screen display. Depending on the data in your file, you may need to change the case to be able to read the data when it is displayed later in the program. If you are not sure what case you need, don't worry—you will be able to return to this option from several points in the program.

Next, the program needs the name of the data file you intend to restructure and where it is located. With this information it will immediately check that file to find the delimiter used and the number of fields per record.

When you enter the name and location of the new file you wish to create, the program checks the disk to see if there are enough free blocks to store the converted file.

If a file with the requested name already exists, the program asks if you want to replace the existing file. If you decided not to replace the file, the program would ask you to:

PRESS E TO EXIT, OR --
ANY OTHER KEY TO REDEFINE

If you request the exit, the program will ask if you wish to quit immediately or return to the bootstrap master menu.

The redefine option first asks if you need a new disk and then allows you to completely redefine the conversion. This time it remembers all the answers you have given and uses them as default prompts. This allows you to simply press RETURN for any answers that are still correct.

This general redefining procedure is used in almost all cases where you appear to be having problems.

When you have either selected an unused output filename or decided to replace an existing one, the program will ask what delimiter is to be used. The delimiter that it suggests by default prompting is the delimiter that it found when it opened the input file.

You should be cautious about changing the delimiter. The program will not allow you to use an illegal delimiter (a number or a quote). However, if you select a new delimiter, you must be sure it is not already used as a data character in the data file. The program can't protect you from this mistake, and it can really mess up a data file.

When the input and output files have been defined, the program clears the screen and presents a display like the following:

```
FIRST RECORD ( 5 FIELDS) IN
SOURCE FILE DATA1 IS:
JONES!TOM!123 MAIN!SOMEWHERE,USA!!
NEW DATA FIELD [ #1 ] TO CONTAIN:
1 DATA FROM AN OLD DATA FIELD
2 NEW FIXED DATA
3 OLD DATA FIELD + NEW FIXED DATA
4 NEW FIXED DATA + OLD DATA FIELD
5 DATA FROM TWO OLD DATA FIELDS
----------------------------------------------------------------
9 REDEFINE OR EXIT
YOUR CHOICE --------- ? 1
```

The first lines on the screen remind you what file you are working with and the number of fields in the first record. The entire first record of the file is displayed to help you in deciding how to restructure the data.

Skipping ahead a little, the final item in this menu allows you to go back and redefine the conversion process. For example, if the displayed record has graphics characters where you expected uppercase, you might want to use option 9 to go back and change the display case.

This complete menu will be repeated for each field you decide to include in your new data file.

Menu Selections

Let's examine each of the menu options as they might be used to create field 1 in the new data file:

Option 1—DATA FROM AN OLD DATA FIELD. Data from any selected field in the input data file will be placed in the new data field you are defining.

For this option the program will ask

```
DATA FROM WHICH OLD DATA FIELD ?
```

and you respond by entering the field number.

Option 2—NEW FIXED DATA. New data which you will enter from the keyboard will be placed in the new data field you are defining. This new data will be repeated in every record of the new file.

For this option the program will ask

```
WHAT NEW DATA
?
```

and you respond by entering any characters you want.

Remember that some special cases such as leading or trailing spaces, commas, colons, and most shifted characters will need to be enclosed in quotations.

Option 3—OLD DATA FIELD + NEW FIXED DATA. New data which you specify will be added to the data from a field in the input data file and will be placed in the new data field you are defining.

For this option the program will ask

```
WHICH OLD DATA FIELD ?
WHAT NEW DATA
?
```

and you respond by entering the number of the old data field and then the characters you want added to it.

Remember that this adding is a string concatenation (attaching end to end) and not a mathematical addition.

Option 4—NEW FIXED DATA + OLD DATA FIELD. This function works just like option 3 except that the new fixed data is placed ahead of the old data.

For this option the program will ask

```
WHAT NEW DATA
?
WHICH OLD DATA FIELD ?
```

and you respond by entering the new characters you want in the field and then the number of the old data field you want added to them.

Option 5—DATA FROM TWO OLD DATA FIELDS. The data from two fields in the input data file will be combined in the order you specify and placed in the new data field you are defining.

For this option the program will ask

```
FIRST OLD DATA FIELD ?
SECOND OLD DATA FIELD ?
```

and you respond by entering the two field numbers.

Each time you finish defining the new contents of a data field, the program will ask if there are any more fields to be defined.

Redefining the Change

When you have finished defining as many data fields as you wish (up to 20) for the new file, you will have the option of continuing with the conversion as you have defined it or going back to change the definition.

```
READY TO CONVERT FILE:
DATA1-TO-NEWDAT
PRESS R TO REDEFINE OR
PRESS ANY KEY TO CONTINUE
```

If you press R you will be allowed to completely redefine the conversion, including the names and locations of the input and output files. The program will assist you in this process by providing default prompts which will be the answers you previously gave for the same questions.

When you decide to let the conversion take place, the program will present a running count of the records as they are converted and saved to disk:

```
524 DATA RECORDS CONVERTED
CONVERSION COMPLETE
ANY MORE FILES ? N
PRESS Q TO QUIT, OR --
ANY OTHER KEY FOR MASTER MENU
```

When the conversion is complete, you will be asked if there are any more files to be converted. This is useful when it is necessary to do the same conversion on a large number of files, because the program will also ask if the same conversion definition is to be used. If it is, you need to enter only the new input and output filenames to start the next file conversion.

If there are no more files to restructure, the program will allow you to quit immediately or return to the bootstrap master menu to select another function.

Restructure Applications

Probably the most common use of this program is to rearrange the positions of the data fields within each record of a file. This is often done so that a complete record sort (sort on field 0) will produce the results normally obtained by a true multifield sorting process.

We have had various other reasons to restructure data files, and we're sure you will find many new applications. To stimulate your imagination, consider these possible uses:

1. You have a data file which was not prepared using the DFH programs. Its format is correct except that it does not have a uniform number of data fields because trailing null fields and trailing delimiters were omitted. The restructuring program will automatically guarantee that each record contains leading quotes, add trailing delimiters, and will produce a uniform number of fields per record.
2. While getting ready to create a new data file, you notice that all of the entries for some fields either start or end with identical character groups (for example, part numbers that all start with P/N). Simply omit these characters when entering the data. The P/N can be added quite easily after the data file is created.
3. You want to add new data fields to an existing file. Use restructuring to create a new blank field at any position you choose. You can then use the editing function to place the new data in the newly created field. Or fill your new field with some fixed data if it will save any typing during the editing process.

Split and Extract

The DFH SPLIT program provides a method of splitting files or extracting parts of files. Perhaps you are saying, "That sounds nice, but where is the practical value?" Let's look at a couple of common applications. (You'll think of more on your own.)

Imagine a file of expense records for your automobiles—we'll let you have two of them. Each record contains four fields: date, vehicle name, cost, and type of expense. The file has been sorted so that the dates are in order.

For some reason, you want to split this single large file into 12 files, each containing the records for one month. The SPLIT function of DFH SPLIT will do this with ease. In fact, it would be almost as easy to do something weird like saving January to March in one file, not saving June and August at all, and saving the rest of the months in individual files.

For a second example, assume you want to extract all records for your Chevy and put them in a separate file. You could re-sort the file on the correct field and then use SPLIT, but the extra work is not necessary. This job is exactly what the EXTRACT function of DFH SPLIT is designed to do, without any preliminary sorting operations.

You can have EXTRACT look for a data pattern in a fixed location within a field or have it search the entire field—it's your choice. You tell the program what data to look for and what field to look in, and it will extract and save all records containing the specified data.

Both SPLIT and EXTRACT leave your original files just as they were. The data which is split or extracted from them is saved in new files.

The SPLIT Operation

This program is called into action by selecting either the SPLIT function or the EXTRACT function in the bootstrap main menu. The first decision you will make is to choose between SPLIT and EXTRACT as shown in the following screen display. (All references to disk drive numbers apply only to PET systems where dual drives are normal. For Commodore 64 systems, the program will not produce such messages.)

READY TO SPLIT FILES OR EXTRACT DATA

```
S SPLIT OR
E EXTRACT                ? S
PREVIEW TO PRINTER       ? N
SOURCE FILE IN DRIVE     ? 0
NAME OF SOURCE FILE      ? EXPENSE
```

The second question will allow you to find out what the results of any split might be without actually creating any new files. This preview can, in some cases, provide all the information you need. For example, you might simply want to know how many expense items there were in each month.

When you have named the first source file, the program will open the file, check the delimiter, and read the first record. This record will be used a little later to help you define the split.

When the program has found the requested source file, it needs to know where to put the output files that will be created.

```
OUTPUT FILES TO DRIVE  ? 0
```

If you select the same drive for both source files and output files, the program will assume that you want the output files on the same disk with the input files. This procedure will be followed unless there is no room on the disk. In that event the program will revert to a two-disk operation, but it will still use only the single drive you selected.

If you are operating on a 64, the preceding message would have no meaning. (The program assumes you only have one disk drive.) In its place you would see:

```
SOURCE AND OUTPUT FILES
ON THE SAME DISK  ? Y
```

A yes answer will allow the program to proceed in a more automatic mode, while a no answer alerts the program to prompt you for disk changes as necessary to save the output files on a separate disk.

Defining the Split Points

The program needs to know what conditions it should look for to decide where to split the file. In general, either this procedure can be defined so that the computer can proceed automatically through the entire job, or it can be set up to allow you to make all the major decisions as the job progresses.

Let's look at a typical display:

```
FIRST RECORD IN EXPENSE IS:
84-01-14*FORD*17.25*FUEL*
CHANGE PRINT CASE         ? N
SPLIT ON WHICH FIELD      ? 1
FIELD 1 = "84-01-14"

SPLIT AT CHANGES IN:
E ENTIRE FIELD, OR
S SELECTED POSITIONS      ? S
START POSITION            ? 4
# OF CHARACTERS           ? 2
SELECTED FROM FIELD 1: "01"
```

Notice you can split the file based on changes within any field, and on changes in the entire field or in any part of it. The result of each choice is displayed for your examination. Later in the process you will get a chance to change any incorrect selections.

Before the actual file splitting begins, you must decide how to save and name the output files.

```
SPLITTING/SAVING PROCESS TO BE:
A AUTOMATIC, OR
O OPERATOR'S CHOICE       ? O
SELECT OUTPUT FILENAMES:
I INDIVIDUALLY OR
S SEQUENTIALLY            ? S
A 3-DIGIT SEQUENCE NUMBER WILL
BE ADDED TO THE NAME YOU ENTER
OUTPUT FILENAME           ? EXP
SPLIT DEFINED OK          ? Y
```

If you want "automatic" file splitting, the file will be split at each change in the data in the selected field (or partial field). If you want to pick and choose from the possible splits, you should select "operator's choice." This will allow you to collect or discard groups of records and save the collected records to disk whenever you wish.

If you requested "individual" filenames, you would not be asked for a filename at this time. Rather, you would be asked to enter a new filename each time you are ready to save a set of collected records to disk.

If you choose the "sequential" naming procedure, as shown in the example, the program will know the correct filename to use when it is ready to save each new file. It creates these names by adding a three-digit number to the basic filename you enter. In this example, the first two files created would be named EXP.001 and EXP.002.

The last question allows you to start over if you have made an error or have changed your mind.

Details of the Split

When you indicate that the split is correctly defined, the program will begin the process of splitting the file. It will find each split point, and only needs to know if you wish to save or discard the records in each group.

The first display is short because there are not many choices you can make at the first split point:

```
SPLITTING FILE EXPENSE
BASED ON CONTENTS OF FIELD 1
0 RECORDS ( 0 BYTES) IN MEMORY

------------------------------------------------------------------------
NEXT RECORD GROUP IS:
 "84-01-14"
1 ADD NEXT RECORD GROUP TO MEMORY
2 DISREGARD NEXT RECORD GROUP
8 DEFINE NEW JOB SETUP
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------            ? 1
```

Options 8 and 9 will be included in all menu displays. Option 8 allows you to completely redefine the SPLIT or EXTRACT job setup. It is almost like a new run except that all your previous answers are used as default prompts.

Option 9 is your way out of the program, either by a complete quit or by going back to the bootstrap main menu.

Now, let's assume you picked option 1 (add to memory), and then made the same choice the next time the menu was presented. We are skipping one variation of the display, but you'll get the idea easily enough. The display will now be expanded to:

```
SPLITTING FILE EXPENSE
BASED ON CONTENTS OF FIELD 1
```

```
91 RECORDS ( 3172 BYTES) IN MEMORY
"84-01-14"
--THROUGH--
"84-02-27"
--------------------------------------------------------------------------------
NEXT RECORD GROUP IS:
 "84-03-12"
1 ADD NEXT RECORD GROUP TO MEMORY
2 DISREGARD NEXT RECORD GROUP
3 SAVE RECORDS IN MEMORY TO DISK

8 DEFINE NEW JOB SETUP
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------          ? 3
```

The expanded display now indicates that 91 records are being held in memory and shows the contents of the selected field for both the first and last of those records. We also have an added option—to save the present memory contents to disk as a separately named file. Let's try option 3:

```
283 DISK BLOCKS FREE
FILE "EXP.001" EXISTS
WANT TO REPLACE IT        ? Y
91 RECORDS OUT TO "EXP.001"

PRESS ANY KEY TO CONTINUE
```

If there were not enough blocks free on the disk, you would be asked to change disks. The question about file replacement will be asked only if needed. If you choose not to replace an existing file, you will get a chance to change disks or quit.

If you change disks for either of these reasons, the program will test the new disk for free blocks and try again to save the new file.

The splitting and saving process will continue under your control until the end of the source file is reached. But this may not be the end of the split.

Split Continuation

The end of a file may be the end of the job, or it might be only the end of one file in a multifile data set. The continuation menu gives you several ways to proceed.

```
SPLITTING FILE EXPENSE
BASED ON CONTENTS OF FIELD 1
15 RECORDS ( 827 BYTES) IN MEMORY
"84-12-02"
  --THROUGH--
"84-12-30"
----------------------------------------------------------------------
END OF FILE EXPENSE
3 SAVE RECORDS IN MEMORY TO DISK
4 CONTINUE TO NEXT SOURCE FILE
8 DEFINE NEW JOB SETUP
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ------          ? 3
```

Option 4, CONTINUE TO NEXT SOURCE FILE, requires the entry of a new source filename. If the split data pattern does not change at the beginning of the new file, and you were adding records to memory, records will continue to be added. If you were disregarding records, they will continue to be disregarded.

If you decide to save the records in memory to disk, that will be done in the usual manner; the menu will be repeated with no records shown in memory, and option 3 will no longer be shown.

Once the split is completed, option 8 allows you to set up a new job without going back to the bootstrap main menu. All of your previous setup answers will be used as input prompts to save time in case the setups are similar.

Extracting Selected Records

Option 8 of the above menu will give you the opportunity to use the EXTRACT function. The first few lines of an EXTRACT setup are very similar to SPLIT. The most obvious difference is that only one output file is created. The only records that are extracted are those containing data that matches your specifications.

As you can see in the following display, the program needs to know how to find the records you want to extract.

```
FIRST RECORD IN EXPENSE IS:
84-01-14*FORD*17.25*FUEL*
CHANGE PRINT CASE        ? N
```

```
WHAT DATA ARE YOU LOOKING FOR
AND WHERE IS IT LOCATED:
WHAT DATA STRING          ? CHEVY
IN WHICH FIELD            ? 2
FIELD 2 = "FORD"
SEARCH FOR STRING AT:
B BEGINNING OF FIELD
S SPECIFIED POSITION
A ANYWHERE IN FIELD       ? B
EXTRACT DEFINED OK        ? Y
```

The program will search only in the field you request. Within that field it can be directed to start the search at the first character or at any other character position. For this example we are looking for all CHEVY entries in field 2.

Another option is to search the entire field for any occurrence of the specified string. This form of search is slower and should be used only when it is really needed.

The last question allows you to redefine the extract function if you wish. All your previous answers will be used as prompts to save time. Just hit RETURN to reconfirm any correct answers.

When you indicate you are happy with the setup, the program will begin searching the source file for any extract records. Since this may be a long process for large files, an activity display is presented which shows a running count of records examined and records extracted.

```
EXTRACTING FROM FILE TF 2
RECORDS WITH "10" IN FIELD 1
50 RECORDS EXAMINED
10 RECORDS EXTRACTED
PRESS ANY KEY TO CONTINUE
```

At the end of each file the program will wait so you can examine the record counts. When you are ready to continue, a new display will be shown:

```
EXTRACTING FROM FILE EXPENSE
BASED ON CONTENTS OF FIELD 1
10 RECORDS ( 704 BYTES) IN MEMORY


------------------------------------------------------------------------
END OF FILE EXPENSE
```

```
3 SAVE RECORDS IN MEMORY TO DISK
4 CONTINUE TO NEXT SOURCE FILE
8 DEFINE NEW JOB SETUP
9 QUIT OR GO TO MASTER MENU
YOUR CHOICE ---------        ? 3
```

This display is very similar to the one you get at the end of a source file while doing a split. You can proceed directly to the next source file (option 4) and continue to accumulate extracted records. If there are no more source files, you should use option 3 to save the records which have been extracted.

When you have completed extracting and saving, you can define a new job setup (either SPLIT or EXTRACT), or quit or return to the bootstrap main menu.

Chapter 4

The DFH Editor

A Sequential File Editor

Have you ever seen a sequential file on one of your disk directories and wondered what was in it? Of course, you could write a program to display the file contents. But it's the old chicken and egg problem. If you knew enough about the file to write a good display program, you might not need to display it.

If you did write a display program, your efforts wouldn't stop there. When you can see the contents of your sequential files, you will, sooner or later, want to modify them. At that point you have just defined your need for an editor.

A Different Editor

Unlike program files, sequential files cannot easily be listed and modified. Years ago we asked, "Why hasn't someone already developed a utility program to do that?"

You could greatly simplify some data processing programs if you had direct access to the data files. One file of this type is your personal address book. Most of the processing is making new entries, deletions, and changes. If these operations could be done with an editor, the processing program would only need to read the file and print it in an acceptable format.

If you do much machine language programming, you already know about a form of direct access. Your source code files are probably prepared for assembly under the control of some type of editor program. (There is a good chance that the DFH Editor presented here can replace your present source code editor.)

The DFH Editor allows you to handle sequential files as though they were BASIC programs. With the DFH Editor you can load, list, modify, and save using the same procedures you use with program files. The DFH Editor also adds some powerful commands you probably haven't seen. We will talk about them later.

File Organization

To understand how and why the DFH Editor works, we need to quickly review the differences (and similarities) of program and sequential files and how the Commodore screen editor helps prepare program files.

As it's loaded from the disk into memory, a sequential file looks quite different from a BASIC program file. Except for the RETURN characters which are used to mark the end of each record, all the bytes in a sequential file are data characters. A BASIC program, on the other hand, contains two extra bytes at the beginning of the file and four additional bytes at the beginning of each line of BASIC code.

Starting at the beginning of the file:

- Bytes 1 and 2 contain the load pointer. This is the memory address where the file loading will start. These bytes are used only to direct the loading and are not stored in memory. The Commodore 64 does not use these bytes unless a trailing ,1 is included in the LOAD command.
- Bytes 3 and 4 contain the link pointer. This is the address where the next program line will start.
- Bytes 5 and 6 contain the BASIC line number.
- The following bytes (as many as needed) are the BASIC program line. A zero byte marks the end of the program line.

This pattern of link pointers, line numbers, and program lines is repeated to the end of the program, which is indicated by both link pointer bytes being 0.

It would seem logical that if we added link pointers and line numbers and changed the RETURN characters to zero bytes, we might be able to handle a sequential file just like a BASIC program. That is correct thinking, but it is not quite enough. The remaining problems are related to the actions of the Commodore screen editor and a process called tokenizing.

Tokens Versus Text

As you create a BASIC program, the screen-editing routine converts all the BASIC commands to single characters called tokens. Each token represents a complete BASIC command. This reduces the storage space required for a program file, both in memory and on disk.

In the reverse process, when a program is listed, the tokens are used to produce the printed BASIC commands you see on the screen. However, if a token character is found inside quotes, it will not be converted to a BASIC command, but will simply be printed.

The characters you put inside quotes are never tokenized because they are considered to be text. You can (and routinely do) put token characters inside quotes because the shifted version of almost every key on your keyboard is a token. You cannot accidentally place a token outside quotes. Even if you type the character on the screen, the screen editor will simply discard it when the line is transferred to memory.

For an editor to work with sequential files, this tokenizing effect must be disabled. We cannot allow the computer to alter any data. It must not create tokens, and it must accept any tokens (shifted characters) that we want to enter as data. We have now defined two more requirements for an editor to be used with sequential files.

These simple changes give you complete freedom to enter anything you want from the keyboard. Well, almost. They do if you understand how the quote mode works.

Quotes and the LIST Function

As stated previously, the normal LIST function includes provisions for changing tokens back to BASIC commands. For an existing file containing token characters, this was a real aggravation, and we considered modifying the LIST operation for use with the DFH Editor.

However, there was a much better way to solve this problem. A new command was created that installs leading quotes in every record in a file. With a leading quote, the LIST function will print all characters without any attempt at conversion.

The ability to insert and delete leading quotes gives you much more control in handling your sequential files. As an illustration of this flexibility, consider the following typical operations using the DFH Editor:

1. For files which do not contain tokens, you can load, list, modify, save, and verify just as though you were handling a BASIC program file.
2. For files which do (or might) contain tokens, you can load, and then install leading quotes. With the quotes in place, you can list and modify the file just like a BASIC program. When ready to save the file, you can either leave the quotes installed or, with a single command, you can discard all of them.

When you are creating a file from the keyboard, you have complete freedom to use quotes or not as you see fit. Just remember that if you type a shifted character in a line that does not have leading quotes, you can expect to see it listed as a BASIC command, or even worse, it may produce a SYNTAX ERROR during listing.

We have covered a lot of ground, so let's take a moment to look at some examples of file creation and handling. Sometimes an example, like a picture, is worth a thousand words. If you look closely, you may notice a procedural error in the program code. It is there to help illustrate the power of the DFH Editor, and will be corrected later.

The programs in these examples are shown as though the computer were set to display in uppercase/lowercase mode (as opposed to graphics mode). If you want to try these examples on a computer that defaults to uppercase and graphics, you should change the display case. For Commodore 64, use the shifted Commodore key to toggle the screen case. For small screen PETs, type POKE 59468,12 for lowercase, and POKE 59468,14 to return to uppercase.

The following program will create a sequential file, named test-1, which will contain the information in the program's DATA statements.

```
100 open 3,8,6,"@0:test-1,seq,write"
110 cr$=chr$(13): rem "carriage return"
120 qt$=chr$(34): rem "quote character"
130 : read a$
140 print#3,a$;cr$;
150 if a$="end" then close 3: end
160 goto 130
170 :
180 data "Ed*203 Grand, Anytown"
190 data "June*14 Birch, City"
200 data "end"
```

Let's try to read and display the sequential file with another program as follows:

```
100 open 5,8,4,"test-1,seq,read"
110 : input# 5,a$
120 if a$="end" then close 5: end
130 print a$: goto 110
```

The printed output from this program will be

```
Ed*203 Grand
June*14 Birch
```

This is not exactly what we wanted. The city names got lost somewhere. Also, look at what happens when we command the DFH Editor to Text Mode and load and list the file.

```
1000 vald*203 chr$rand, atnnytown
1010 mid$une*14 peekirch, lenity
1020 end
```

Here again we see something that looks like a problem. All the shifted characters are being displayed as BASIC commands. We know the word Grand is in the sequential file because our second program got it and printed it, yet it is listed as chr$rand.

However, if we use the DFH Editor to install leading quotes with its ;QT command and list the file again, we will see:

```
1000 "Ed*203 Grand, Anytown
1010 "June*14 Birch, City
1020 "end
```

You must use the ;QT command to insert the leading quotes. You must not try to insert them manually after listing. That would only preserve the characters that had been printed on the screen, and the actual data bytes would be lost.

If we now use the DFH Editor to save the file and then rerun the second program to read and display the file, we will see:

```
Ed*203 Grand, Anytown
June*14 Birch, City
```

Of course, we could get the same results by correcting that deliberate error in the first program. Simply change line 140 to read:

```
140 print#3,qt$;a$;cr$;
```

Adding the qt$ variable causes the program to insert a leading quote in every record. With the file now created properly, the display program will work, and the DFH Editor can load and list the file without any distortion.

The most important thing these examples have done is to demonstrate the importance of leading quotes in sequential files. I suspect that most existing data files do not contain quotes, but this simple change to the structure can add tremendous flexibility by removing almost all restrictions on what data characters can be saved in the file.

Remember that when the INPUT# command is used to read a data record, the leading quote is discarded. If you are using one of your own programs to handle data records, you must remember to reinstall the leading quote when you store the data records on disk.

Using the Editor

The normal method of activating the DFH Editor is to load and run the bootstrap program, DFH BOOT, and select the DFH Editor function from the main menu. The correct editor program for your computer (DFH ED.C64$90 for the Commodore 64 or DFH ED.PET$70 for the PET) will be loaded and activated. Total control will be returned to the keyboard with the DFH Editor in Text mode ready to work with sequential files.

If you know that the DFH Editor program is already loaded, you can simply execute a SYS to the activation address:

SYS 36864 to activate the 64 editor.
SYS 28672 to activate the PET editor.

Due to its location at $9000 (decimal 36864) in the 64, the DFH Editor can remain installed (not necessarily activated) at all times once it has been loaded. Thus, it will usually be available for activation with a SYS command.

In PET computers, with less memory available, the situation is quite different. There the DFH Editor is located at $7000 (decimal 28672), and shares memory space with the machine language subroutines used by other DFH programs. Therefore, the DFH Editor will only be present when it has just been selected from the DFH main menu.

Of course, if you wish to take total control, you can load the DFH Editor for your computer directly into memory and SYS to the activation address. When used this way, the DFH Editor becomes completely independent of the remaining programs in the DFH family.

During the activation routine the DFH Editor installs a command interception wedge in the computer's Character Get (CHRGET) routine. The DFH Editor sets top of memory to protect itself from strings created by BASIC programs. The amount of memory reserved for the DFH Editor should not affect the vast majority of your other computer programs.

Deactivating the Editor

One of the DFH Editor commands, ;MK, is the primary means of deactivating the DFH Editor. Deactivating the editor does not release the top of memory back to its normal address. The intent is that the editor will remain protected should you wish to reactivate it at a later time. If you wish to restore normal top of memory without cycling power, it can be done quite simply by typing

POKE 55,0: POKE 56,160: CLR (for the Commodore 64)

or

POKE 52,0: POKE 53,128: CLR (for the PET)

The DFH Editor can also be tested from within a BASIC program to see if it is activated, and can be deactivated by a SYS command.

For Commodore 64 computers:

If PEEK (36876) = 242 the editor is active.
If PEEK (36876) = 243 the editor is not active.
Deactivate by SYS 36867.

For PET computers:

If PEEK (28684) = 242 the editor is active.
If PEEK (28684) = 243 the editor is not active.
Deactivate by SYS 28675.

This method of checking and deactivating is intended for use by your own BASIC programs when those programs need to be loaded into the memory occupied by the DFH Editor. Note that BASIC programs will run a little faster if the editor is not active. This is because the Character Get routine does not have to check for the presence of editor commands.

The DFH Editor Commands

The DFH Editor provides 14 new file-editing commands. Five of them can be used only with sequential files (Text mode). The remaining commands can be used in Text mode and also with program files (BASIC mode).

Command	**Meaning**	**Use For**
;	Display menu & set repeat	Text & BASIC
;AD	Add a character	Text
;AU	Auto line numbering	Text & BASIC
;CS	Change screen case	Text & BASIC
;DE	Delete lines	Text & BASIC
;ED	Erase screen down	Text & BASIC
;EU	Erase screen up	Text & BASIC
;FC	Find and change	Text

;FI	Find string	Text
;MB	Set BASIC Mode	Text & BASIC
;MK	Kill DFH Editor	Text & BASIC
;MT	Set Text Mode	Text & BASIC
;QT	Insert quote	Text
;RN	Renumber lines	Text
;UN	UnNew	Text & BASIC

In the following paragraphs we will examine the functions of each of these commands. They are described in alphabetical order.

Display Menu & Set Repeat

Used in Text or BASIC mode.

Syntax: ; or ;(any invalid command code)

When the DFH Editor command is simply the edit prefix character (;) or the prefix character followed by an invalid command code, the complete DFH Editor menu will be displayed on the screen. The current operating mode, Text or BASIC, is also displayed.

The menu can be called to refresh your memory when you can't remember the code for a desired function. When the menu appears in response to an error, it serves as a reminder of what commands are available.

The Menu command also performs a hidden function. It reestablishes the repeating keys feature. The repeat will be lost on some PET computers when a tape LOAD or SAVE is performed or when any LOAD command is executed within a BASIC program. If you suddenly find yourself with a nonrepeating keyboard, just call the menu, and the repeat feature will again be operating.

Add A Character

Used in Text mode.

```
Syntax is: ;AD char,Range
                      ,R1
                      ,R1-
                      ,-R2
                      ,R1-R2
```

Where: char = Character to be added; R1 = start line number; R2 = end line number.

This command adds the specified character to the end of each line within the specified range. If a range is not specified, the character will be added to all lines in the file.

The message SYNTAX ERROR is displayed if the range is not specified properly or if more than one Add character is specified.

The message LINE >74 CHRS is displayed for any line that contains more than 74 data characters after the new character is added.

The character will not be added if the resulting line would contain more than 250 data characters.

Here are some suggested uses for this command.

- When adding fields to existing data records. Use it to install trailing field delimiters and trailing fixed data patterns to reduce your typing effort.
- If you suspect there may be trailing spaces in some records, install a presently unused character at the end of each record. The trailing blanks can then be seen, and the records can be edited without losing the trailing spaces.
- To add a string of characters, simply add a presently unused character and then use the ;FC (Find and Change) command to change the dummy character to the desired string. Use the ;FI (Find) command to check for unused characters.

Auto Line Numbering

Used in Text or BASIC mode.
Syntax: ;AU incr

Where: incr = line number increment value.

This command causes automatic printing of a correctly incremented next-line number during data entry or editing operations. The numbers are printed following a carriage return on a numbered line containing data. The new number will be the number of the entered or edited line plus the increment value.

Automatic line numbering is disabled if the increment value is less than 1 or if it is not specified. It is also disabled by running a program; by executing the Renumber command; or a DOS LOAD, Append, or SAVE command.

Change Screen Case

Used in Text or BASIC mode.
Syntax: ;CS

This command switches the display case of the screen. The contents of the computer's memory do not change.

Delete Lines
Used in Text or BASIC mode.
Syntax: ;DE Range

```
              R1
              R1-
              -R2
              R1-R2
```

Where: R1 = start line number; R2 = end line number.

This command deletes all lines within the specified range of line numbers.

The range syntax is the same as for a LIST command. The defaults are R1 = 0 and R2 = 63999, but at least one range parameter must be specified. When both range parameters are used, they must be in ascending order.

The line delete command should not be used to delete an entire file. Use the NEW command for that purpose.

The message SYNTAX ERROR is displayed if the range is not specified properly.

Erase Screen Down
Used in Text or BASIC mode.
Syntax: ;ED

This command erases the screen from the line containing the command down to the bottom of the screen. For 40-column screens, the existing screen line linking is retained after the erasure. The linking pattern will exist until the CLR key is pressed or until the linked lines are scrolled off the screen.

When a line of more than 40 characters is typed or listed onto a 40-column screen, the first and second physical lines are linked into an 80-character logical line. This linked structure is quite easy to see when the data that caused it is still on the screen. However, when the data is erased, using the ;ED or ;EU commands, the linking pattern still exists and it can cause unexpected results.

This usually happens when you type a command on what appears to be an empty line, when in fact it is logically linked to the line immediately above and that line already contains some characters. A simple rule will keep you out of trouble.

When you have recently executed an Erase command, always keep at least one blank line above the cursor. This way, there is no possibility that your newly typed input could be accidentally linked to characters above it on the screen.

Erase Screen Up

Used in Text or BASIC mode.
Syntax: ;EU

This command erases the screen from the line containing the command up to the top of the screen. Existing screen-line linking is retained for 40-column screens. (See the caution about linked lines under "Erase Screen Down" above.)

Find and Change

Used only in Text mode.

```
Syntax:  ;FC /old/new/,Range
                      ,R1
                      ,R1-
                      ,-R2
                      ,R1-R2
```

Where: old = String to be found; new = Replacement string; / = The string delimiter character (not contained in either string); R1 = Start line number; and R2 = End line number.

This command finds a specified "old" character string occurring in a range of line numbers and changes it to the "new" character string. If a "range" is not specified, the complete file will be searched.

All lines where changes are made are displayed. If more than one change is made in a line, the entire line is displayed once for each change.

Lines containing up to 250 total data characters can be modified by this command. The execution of this command can be paused or resumed by momentarily pressing the space bar. It can be halted by pressing the RUN/STOP key.

The message SYNTAX ERROR will be displayed if the range is not specified properly or if all three string delimiters are not the same.

The message DATA >74 CHRS will be displayed if a line contains more than 74 data characters after the change has been accomplished.

The message CAN'T ALTER NEXT LINE will be displayed, followed by the problem line, if the requested change would create a line having more than 250 data characters. The Find and Change operation is terminated by this error.

Character strings can be found and deleted by not specifying a "new" string, as in ;FC/ABC//.

Find String

Used only in Text mode.

Syntax< ;FI /string/,Range
```
                     ,R1
                     ,R1–
                     ,–R2
                     ,R1–R2
```

Where: string = String to be found; / = String delimiter character (not contained in the string); R1 = Start line number; R2 = End line number.

Use this command to find and display a specified character string occurring in a range of line numbers. If a range is not specified, the complete file will be searched.

If the string is found more than once in a line, the entire line is displayed each time the string is found. The execution of this command can be paused or resumed by momentarily pressing the space bar. It can be halted by pressing the RUN/STOP key.

The message SYNTAX ERROR will be displayed if the range is not specified properly or if both delimiters are not the same. The message DATA >74 CHRS will be displayed if a found line contains more than 74 data characters.

Set BASIC Mode

Used in Text or BASIC mode.

Syntax: ;MB

This command sets the DFH Editor to BASIC mode. This is the mode your computer is in when you turn power on. In this mode you are assumed to be working with BASIC program files. Machine language program files are also loaded and saved in BASIC mode.

The DFH Editor functions which work in both BASIC and Text modes are set to perform correctly with BASIC program files. The DFH Editor functions which work only with sequential files are disabled. The DOS commands for loading, saving, verifying, and appending are set for proper handling of BASIC and machine language program files.

The message BASIC MODE is displayed.

Kill Editor

Used in Text or BASIC mode.
Syntax: ;MK

The Kill command deactivates all DFH Editor functions, including the DOS commands. The command intercept wedge is removed, and the CHRGET routine is restored to its power-on condition.

The message DFH EDITOR KILLED is displayed.

The top-of-memory setting which was established when the DFH Editor was activated is not altered by this command. See Deactivating the Editor, page 83, for additional top-of-memory notes.

Set Text Mode

Used in Text or BASIC mode.
Syntax: ;MT

This command sets the DFH Editor to Text mode. All DFH Editor functions are active and are set to work correctly with sequential files. The DOS commands for loading, saving, verifying, and appending are set for proper handling of sequential files.

The message TEXT MODE is displayed.

Insert Quote

Used only in Text mode.

```
Syntax: ;QT S,Range
              ,R1
              ,R1-
              ,-R2
              ,R1-R2
```

Where: S = Stop on error flag, optional (only perform error checks); R1 = start line number; R2 = end line number.

Without the stop flag (S), this command inserts a quote character at the start of all lines within the specified range. If the first character is already a quote, the line will not be changed. If a range is not specified, the entire file will be processed.

The message SYNTAX ERROR is displayed if the range is not specified properly. IMBEDDED QUOTE is displayed if a quote character is found other than as the first or last character in the line. DATA >74 CHRS is displayed if there are more than 74 data characters (including the quote) in the line.

When the stop flag (S) is included, quotes are not inserted, but each line in the specified range is checked for errors. If an error is found, the line is displayed along with the appropriate error message and the checks are halted. This process assumes that you would be wanting to correct the detected errors.

Additional notes:

1. The Insert Quote command should be used before editing a file which might contain numbers as the first data characters in any record. The leading number would be interpreted as part of the line number during editing and would cause incorrect results.
2. The Insert Quote command should be used before editing a file which might contain shifted characters. Without leading quotes, most shifted characters will be interpreted as BASIC tokens. When these characters are LISTed, they will appear as BASIC commands. Editing such a line would reinstall the line with the commands as character strings rather than the token equivalent.
3. If you have inserted quotes to enable editing a file but want to save the file without the quotes, they can be deleted with the ;FC (Find and Change) command: ;FC/"//.

Renumber

Used only in Text mode.

```
Syntax: ;RN NI,NS,Old Range
                  ,R1
                  ,R1-
                  ,-R2
                  ,R1-R2
```

Where: NI = Line number increment value; NS = New start line number; R1 = Old start line number; R2 = Old end line number.

This command renumbers lines in the specified range of old line numbers, assigning a new start line number and using the specified increment value. The default values are: NI = 10, NS = 1000, R1 = 0, R2 = 63999. Illustrations of defaulting combinations are shown in the examples which follow.

Caution: To allow you maximum flexibility, no error checking is performed before the new line numbers are assigned. If you are using the "old range" specification, you will usually need to insure that the line numbers remain in proper sequence. If you discover that the line numbers are not all in ascending order, renumbering the entire file will correct the problem.

Examples of renumbering:

;RN Renumbers the entire file. New line numbers start at 1000 and increment by 10.

;RN 5 Renumbers the entire file. New line numbers start at 1000 and increment by 5.

;RN 15,2000 Renumbers the entire file. New line numbers start at 2000 and increment by 15.

;RN 2,1400,1000-3000 Renumbers only existing lines in the range 1000–3000. New line numbers will start at 1400 and increment by 2.

UnNew

Used in Text and BASIC modes.
Syntax: ;UN

UnNew restores the last data file or BASIC program contained in memory if a NEW command has been executed. The message ERROR will be displayed if the file or program in memory cannot be reconstructed.

Chapter 5

Disk Use

Disk Support Commands

Many programmers use one of the available shorthand command sets for the Disk Operating System (DOS). Very often the program used is "DOS 4.0" or "UNIVERSAL WEDGE" for PET computers or "DOS 5.1" for Commodore 64 computers.

Not a New DOS!

In creating the DFH Editor, there was no desire to create a new DOS shorthand command set. As far as they went, the existing commands were just fine. Additional capabilities were needed much more than changes.

Consequently, the DOS shorthand command set for the DFH Editor will look familiar to anyone who is presently using one of the Commodore DOS programs. In fact, you could probably use it for a long time before discovering any of the extended capabilities.

Some commands perform differently depending on the mode of the DFH Editor. For example, a sequential file is loaded differently than a BASIC file, but there was no reason to invent a new LOAD command. This DOS knows that it can only load BASIC files in BASIC mode and sequential files in Text mode.

Some commands have an extended parameter set. For example, in BASIC mode, the LOAD command can now be directed to the computer's start of BASIC, to the LOAD address contained in the program file, or to any address specified by the operator. As the LOAD is executed, the start and end addresses are displayed so that you always know exactly where the program is located in memory.

Several new commands such as SAVE, Verify, and Append were added to the shorthand command set. In addition to the shorthand convenience, extended parameter sets make these commands very powerful. For example, the SAVE command can perform a normal SAVE, or it can be directed to save any address range of memory. For single-drive Commodore 64 systems, this is handy because machine language programs can now be moved from disk to disk as easily as BASIC programs.

Various small changes were made to provide more convenience for the operator. For example, why should you have to write a small program to see why the disk error light is on? When this utility is used, any command which provokes a disk error will report the error condition in plain English.

All in all, you should find this DOS powerful and easy to use.

DOS Activation

The DOS functions are activated (and deactivated) along with the file-editing functions of the DFH Editor. For most operations you can leave the DFH Editor, and consequently the DOS, active at all times. This is true even when you are writing or running programs that have nothing to do with data file handling. In fact, if you don't already have a good BASIC programming support utility, you will find that some DFH Editor commands can be very useful while you're writing or revising BASIC programs. Just remember to put the DFH Editor into BASIC mode before trying to work on BASIC programs.

On those occasions where there is a conflict between the DFH Editor and another program, it will usually be because both programs want to be loaded into the same memory addresses. In the Commodore 64 the memory from $9000 through $9FFF (decimal 36864 to 40959) should be reserved for the DFH Editor. In PET computers, the memory from $7000 through $7FFF (decimal 28672 to 32767) should be reserved.

DOS Shorthand Commands

The Commodore Disk Operating Systems are fully supported using a shorthand syntax similar to DOS 4.0 (for PET) or DOS 5.1 (for Commodore 64). These shorthand commands can be used only from the keyboard. Inside a BASIC program you must still use the normal disk command syntax as described in your computer user's manuals.

The Disk Error Channel is read after each disk command is executed. Any message other than 00 OK 00 00 will be displayed. A general requirement for all commands is that the first character of the DOS command must be the first nonspace character on the screen line when the RETURN key is pressed.

The primary DOS command codes and their general meanings are shown in the following table. In most cases, these codes are followed by other characters to completely define the command.

>	Read error channel
>#	Set default device number
>$	Display directory
>C	Copy disk file
>D	Duplicate disk
>I	Initialize disk
>N	New (format) disk
>R	Rename a file
>S	Scratch a file
>V	Validate a disk
/	Load a file
↑	Load and run
&	Append to memory
←	Save a file
]	Verify a file

Read Error Channel

Syntax: >

This command reads the disk error channel and displays the error message on the screen. This command has no options, and no additional characters are allowed. If the disk error light is on, this command will display the error message and clear the error status.

Set Default Device Number

Syntax: ># device#

Where: device# = Disk device number.

Use this command to set the disk device number that will be used as a default by the DOS commands.

When the DFH Editor is activated, the default device number is set to 8.

The current device number is displayed, along with the current operating mode, when this command is executed. The current device number is also displayed when the DFH Editor menu display is commanded.

Display Disk Directory

Syntax: >$ dr:qualifiers

Where: dr = drive number; qualifiers = partial directory specifications.

Use this command to display the disk directories without disturbing memory contents. The display can be paused or resumed by momentarily pressing the space bar or halted by pressing the RUN/STOP key.

The drive number is optional. If it is not specified, the directories for both drives will be displayed on dual drive systems.

When they are used, the qualifiers determine which directory entries will be listed. They allow selective examination of a directory as shown in the last five of the following examples:

>$ Display the complete disk directory (64), or both directories (PET).

>$1 Display the complete disk directory for the disk in drive 1.

>$1: Display only the disk title and number of blocks free on drive 1.

>$0:AB* Display drive 0 directory titles beginning with "AB".

>$0:?A?B* Display drive 0 directory titles with second character = "A" and fourth character = "B".

>$1:*=S Display all the sequential file titles on drive 1.

>$0:AB*=P Display all the program file titles on drive 0 beginning with "AB".

Copy Disk Files

Syntax: >C dr:new=dr:old1,dr:old2....
Where: dr = drive number; new = new filename; old = old filename(s).

Copy Disk Files is usually used to copy files from one disk drive to another. For this purpose it can be used only with a dual disk drive.

It can be used on a single disk drive to copy a file back onto the same disk with a new filename assigned. However, the Rename command is often better suited to that task.

When multiple "old" filenames are specified, the "new" file will contain the concatenated (added end to end) combination of all the old files in the order they were specified.

A special form of this command >**C dr=dr** will copy all files on the disk in one drive onto the disk in the other drive. This form of the Copy command works on all dual drives except for the Model 2040.

Disk drive numbers are always required and wild card characters (? or *) cannot be used in the filenames.

The following are examples of the major forms of the Copy command:

>C1:FILEA=0:FILEA Copy a file on drive 0 named FILEA to drive 1.

>C0:FILEA=1:FILEB Copy a file on drive 1 named FILEB to drive 0 where it will be named FILEA.

>C0:FILEA=0:FILEB Copy a file on drive 0 named FILEB to a new location on drive 0 where it will be named FILEA. When the drive numbers are the same, the filenames must be different.

>C1=0 Copy all the files on the disk in drive 0 onto the disk in drive 1. This form of the Copy command is not valid for Model 2040 disk drives.

>C0:FILEA=0:FILEB,0:FILEC Copy FILEB and FILEC into a new file named FILEA. The concatenation proceeds in the order the source files are specified.

Up to four files can be concatenated by a single command so long as the total number of characters in the command is less than 40. This form of the Copy command can be used to perform the functions of the CONCAT command found in BASIC 4.0 systems.

For total disk copy operations on dual drive systems, the Copy command is much preferred over the Duplicate command for the following reasons:

1. The Copy command allows the disk identification characters to remain different on the two disks, while the Duplicate command does not. Refer to the Error Sources and Handling section for detailed information on this very important subject.
2. The Copy command moves only valid file data, so it is usually quicker than the Duplicate command. Also, the data is stored in optimum track and sector locations on the destination disk, which can provide somewhat faster access.

Duplicate Disk

Syntax: >D newdr=olddr
Where: newdr = destination drive number; olddr = source drive number.

The Duplicate Disk command is only valid for systems with a dual disk drive. It performs the same function as the BACKUP command in BASIC 4.0 systems. As an example, the command:

>D1=0

duplicates the contents of the disk in drive 0 onto the disk in drive 1. The disk title, disk ID, Block Allocation Map, directory, all files, and all unused blocks are duplicated. No disk cleanup is attempted.

This command will not work unless all tracks and sectors can be read. Thus, it cannot be used with a disk that has a defect which has been manually blocked off (marked as used) in the Block Allocation Map.

Caution: Severe problems can develop from having more than one disk with the same ID in your library. Therefore, the Copy command or a separate disk-copying program should be used in place of the Duplicate command whenever possible. This subject is discussed in detail in the Error Sources and Handling section.

Initialize Disk

Syntax: >I dr
Where: dr = drive number

This command causes the drive controller to load the Block

Allocation Map from the disk into the controller memory. The general purpose is to let the drive controller know that a different disk has been installed in the drive.

The drive number is not required for single-drive disk units, but if it is used it must be specified as drive 0.

If the drive number is omitted on a dual drive system, both drives will be initialized.

An initializing command is required for Model 2040 drives when disks are changed. Its use is optional for most other drives unless disks with the same ID code are being changed.

Duplicate ID codes within your disk library are very dangerous except with later model 8050 and model 8052 disk drives. Sooner or later you will destroy data on a disk because of a failure to initialize allowed by the duplicate ID codes. This subject is discussed in detail in the section on Error Sources and Handling.

New (Format) Disk

Syntax: >N dr:diskname,ident

Where: dr = drive number; diskname = title of the disk; ident = disk ID code.

Use this command only on new disks or on disks that contain files you don't want any more. The New Disk command prepares (formats) a disk for first-time use by writing all necessary track, sector, and directory information. It is equivalent to the HEADER command in BASIC 4.0 systems.

If the disk has never been used, the two-character identification code must be included in the command. In this case the formatting is a lengthy operation which involves writing in every track and sector on the disk.

If the disk has already been used and if you do not need to change the ID code, a shorter form may be used. When the ID is not specified, a new disk title is created, the Block Allocation Map is cleared, and all files are marked as *scratched.* This operation takes only a few seconds and is commonly referred to as a Short Form New.

The results of a Short Form New appear very similar to a Complete New, since the directory will show a completely empty disk. Actually, all previous file contents are still on the disk, but are not accessible except by specialized disk file-recovery programs.

Rename a File

Syntax: >R dr:newname=dr:oldname
Where: dr = drive number; newname = new filename; oldname = existing filename.

Change the name of a disk file with this command. The file is not moved. Only the filename in the disk directory is changed.

The second drive number is not required for most systems. If it is used, both drive numbers must be the same.

Scratch a File

Syntax: >S dr:qualifiers
Where: dr = drive number; qualifiers = filename selection information.

Be careful using this command. It will delete one or more files on a disk and return a confirming message. For example, the message "01, FILES SCRATCHED, 03,00" would indicate three files were scratched.

If a full filename is used as a qualifier, only the named file will be scratched. Groups of files can be scratched by using wild card characters (? or *) in the qualifier portion of the command.

The following are examples of commonly used forms of the Scratch command:

>S0:TABLE Would scratch the file named TABLE on drive 0.
>S0:AB* Would scratch all files on drive 0 that have names beginning with AB.
>S0:??A?? Would scratch all files on drive 0 with five-character names where the third character is A.
>S1:* Would scratch all the files on drive 1. This can be done much faster using the short form of the Disk New command.

Caution: You should never use the Scratch command as a substitute for the Validate command to get rid of an open file. The sector linkages are not set correctly in an open file, and valid sectors of another file can be left open for reuse during future write operations. Additional information on this subject is included in the section on Error Sources and Handling.

Validate a Disk

Syntax: >V dr
Where: dr = drive number.

The primary use of this command is to remove directory entries for files that have been left open. An open file is identified in a directory listing by an asterisk (*) just ahead of the file type code, such as *PRG or *SEQ.

Validate is the same as the COLLECT command in BASIC 4.0 systems. It constructs a new Block Allocation Map for the disk by tracing the block linkages for all properly closed files. All open files are marked scratched.

Caution:

1. You should never use the Scratch command as a substitute for the Validate command, to get rid of an open file. The dangers are explained in the section on Error Sources and Handling.
2. Never use the Validate command on a protected commercial disk, on a disk containing REL files, or on a disk which has user-allocated sectors. The BAM for the disk will probably not be reconstructed properly in those cases, and will eventually result in loss of data.

Load a File

Syntax: / dr:filename,qualifier

Where: dr = drive number; filename = name of file; qualifiers = loading directions (BASIC mode only).

This command is used to load program (PRG) and sequential (SEQ) files from disk into memory. The operation of this command depends on the mode of the DFH Editor. Program files can be loaded in BASIC mode and sequential files can be loaded in Text mode. User (USR) files and relative (REL) files cannot be loaded with this command.

A command to load a file which does not match the current mode will produce the error message FILE TYPE MISMATCH, followed by a reminder of the current mode.

The drive number is not needed for single drive systems. If it is used, it must be 0. With dual drives the drive number is optional. If it is omitted, both drives will be searched for the named file. A good rule to remember is that drive numbers are optional for loading, but they are always required for saving.

Both types of wild card characters (? and *) can be used in the filename. For example, specifying the filename as ?A* would cause loading of the first file found on the disk which has a filename with an A as the second character.

Because the LOAD command operates differently depending on the mode of the DFH Editor, the two modes will be discussed separately.

Loading Sequential Files

Syntax: / dr:filename

Since the qualifier parameter is not valid in the Text mode, it is not used here. A typical LOAD command might be:

```
/0:TEST1
```

This command would cause the sequential file named TEST1 to be loaded from drive 0 into the computer memory.

The loading would start at the current start-of-BASIC address. For 64 systems, this would normally be $0801 (decimal 2049), while for PETs it would be $0401 (decimal 1025).

When sequential files are loaded, line numbers are added to each record as it is received from the disk. The line numbers start at 1000 and increment by 10.

The value of the status variable (ST) will be displayed at the end of the LOAD operation, along with a reminder of the current operating mode, BASIC or Text. ST=40 (hex, decimal 64) is normal for a good LOAD operation.

Loading Program Files

The qualifier after the filename is an optional parameter for the LOAD command when the DFH Editor is in BASIC mode. This allows three forms of the LOAD command:

/ dr:filename for a "relative" load.
/ dr:filename,1 for an "absolute" load.
/ dr:filename,$xxxx for a "directed" load.

Let's examine each of these three forms.

For a relative LOAD, the program file will be loaded at the current start of BASIC address. For Commodore 64 systems this address is normally $0801, and for PETs it is $0401. The address of the end of the program is determined by the length of the program file.

A relative LOAD will normally be used to load BASIC programs. With this feature, PET computers can now load BASIC programs prepared on 64 systems. Special relocation procedures are not required.

For an absolute LOAD, the program file will be loaded at the address specified by the LOAD point bytes contained in the file. The LOAD point bytes are the first two bytes in all program files. The LOAD point address bytes are automatically set when the file is saved. They always indicate the address where the program was located when the SAVE was performed.

An absolute LOAD will normally be used to load machine language programs.

The absolute LOAD operates exactly like the absolute LOAD (using the trailing ,1) of the Commodore 64 computer. This is also identical to the normal LOAD operation in a PET computer.

The directed LOAD is a new form which allows you to direct the LOAD to begin at any address (designated in hexadecimal). When used with its counterpart, the directed SAVE, it can be used to move the contents of any section of RAM or ROM to any memory address you want.

As an example for the Commodore 64, you could create and *direct save* a screen image, then *direct load* it to an alternate screen location and direct save it from there. You now have an alternate screen image that can be loaded by a BASIC program.

Another use for the directed LOAD is for program files that would normally load into zero page (addresses $0000 through $00FF) or other areas where the operating condition of the computer is altered. These files can be direct loaded into a less delicate part of memory where they can be examined, and perhaps changed, by another utility program such as a machine language monitor.

For all forms of program loading, the start and end address of the actual LOAD will be displayed in hexadecimal notation.

Appropriate error messages will be displayed for any disk errors that are encountered during the loading operation.

PET users should note that in the preceding description we have altered the normal DOS LOAD command to the form used by 64 computers. As longtime PET users, we are very unhappy that Commodore did not choose to make the relocating LOAD feature compatible in both directions by having the Commodore 64's ,1 indicate relocation rather than absolute. However, we are stuck with it and have simply tried to make the best out of a bad situation by accepting the new standard.

If you like to use the DOS LOAD and RUN command, you may soon encounter a special problem that can be easily corrected. Many machine language programs that are designed to load and run like a BASIC program are assembled with a LOAD point of $0400 rather than $0401. The LOAD and RUN command in this DOS always assumes that a BASIC program is being loaded, and for PET computers, defaults to a LOAD at $0401. This leaves the program offset by one byte, and it will not run.

The solution is very simple. Load the program using the ,1 for an absolute LOAD and then resave it using the relative form of the SAVE command. The relative SAVE will establish a new LOAD point address of $0401 which allows you to use the LOAD and RUN command.

Load and Run

Syntax: ↑dr:filename

Where: dr = disk drive number; filename = name of file to be loaded.

Load and run BASIC program files with this command; not valid in Text mode.

The drive number is not needed for single-drive systems. If it is used, it must be specified as 0. For dual-drive systems, the drive number is optional. If it is omitted, both drives will be searched for the named file.

This command always performs a relative LOAD. The program is loaded beginning at the current start of BASIC. For 64 computers this address is normally $0801, and for PETs it is $0401.

The start and end addresses of the load will be displayed in hexadecimal notation. However, they may be hard to see if the program you are running begins with a CLEAR SCREEN command or other commands which would remove the LOAD message from the screen.

Both types of wild card characters (? and *) can be used in the filename. For example, specifying the filename as ?A* would cause loading of the first file found on the disk which has a filename with an A as the second character.

The message FILE TYPE MISMATCH, followed by a reminder of the current mode, will be displayed if the requested file is not a program (PRG) file.

Appropriate error messages will be displayed for any disk errors that are encountered during the loading operation.

Append to Memory

Syntax: & dr:filename

Where: dr = drive number; filename = name of file to append.

Use Append to Memory to load program (PRG) or sequential (SEQ) files from disk and append them to a file already in the computer memory. The operation of this command depends on the mode of the DFH Editor. Program files can be appended in BASIC mode and sequential files can be appended in Text mode. User (USR) files and relative (REL) files cannot be appended.

The message FILE TYPE MISMATCH, followed by a reminder of the current mode, will be displayed if the requested file does not match the current mode of the DFH Editor.

The drive number is not required for single-drive systems. If it is used, it must be 0. With dual drives the drive number is optional. If it is omitted, both drives will be searched for the named file.

Both types of wild card characters (? and *) can be used in the filename. For example, specifying the filename as ?A* would cause loading of the first file found on the disk which has a filename with an A as the second character.

In Text mode the complete file in memory will be renumbered following the append operation. This is necessary because line numbers do not exist on the disk for sequential files, but are assigned when the file is loaded or appended.

In BASIC mode the line numbers of the appended file are not altered. It is the user's responsibility to be sure that the line numbers of the appended file are all greater than the largest line number of the original file in memory. Otherwise, it may not be possible to edit the resulting file. To prevent this problem, renumber the file which is to be appended, before you try to append it.

The value of the status variable (ST) will be displayed at the end of the Append operation, along with a reminder of the current operating mode. ST = 40 (decimal 64) is normal for a good LOAD operation.

This command is not intended to be used to append a machine language file to a BASIC program. It will perform the

Append, but you will probably not be able to list the result because the last two bytes of the BASIC program will have been altered by the Append operation.

Appropriate error messages will be displayed for any disk errors that might be encountered during the appending operation.

Save a File

Syntax: ←@dr:filename,range
Where: @ = Replacement mode indicator; dr = Disk drive number; filename = Name of file to save; range = Hex address range (BASIC mode), or Line number range (Text mode).

This command is used to save sequential files, BASIC programs, or other memory images (such as machine language programs) to disk.

SAVEs performed with the DFH Editor in Text mode will produce sequential (SEQ) files, while SAVEs in BASIC mode will produce program (PRG) files. This is completely independent of the actual nature of the memory contents. For example, it would be possible (but not useful) to load a BASIC program while in BASIC mode, then switch modes and save it as a sequential file. Watch your mode changes!

The @ is optional. When used, it indicates that the saved file should replace any existing file which has the same name. If the @ is not used and a file with the specified name already exists, a FILE EXISTS message will be displayed and the SAVE will not be performed.

The drive number is required. A SYNTAX ERROR message will be displayed if the drive number is not included in the command.

The filename must be fully specified and must not contain any wild card (? or *) characters, commas, or colons.

The current DFH Editor mode is displayed when the SAVE has been completed.

The actions of the SAVE command are dependent on the current mode of the DFH Editor as discussed in the following paragraphs.

Saving Sequential Files

SAVEs performed with the DFH Editor in Text mode will produce sequential (SEQ) files. The line numbers are discarded as the records are saved.

In Text mode, the range parameter can be used to specify a range of numbered lines to be saved. For this mode the syntax of the SAVE command can be shown as:

```
←@dr:filename,range
              R1
              R1-
              -R2
              R1-R2
```

Where: R1 = Start line number; R2 = End line number.

The range parameter is optional. If it is omitted, the entire file in memory will be saved. The file occupies the section of memory defined by the start-of-BASIC and end-of-program pointers.

The range parameters are used in the same manner as for a LIST command. For example:

350–500 = lines from 350 through 500.
650– = all from 650 through end of file.
–200 = all lines from start of file through 200.

An example of the SAVE command using the range parameter in Text mode is:

```
←0:TESTFILE,1540-2200
```

This command would save lines 1540 through 2200 as a sequential file named TESTFILE. Notice that, in this example, the optional @ has been left out so the SAVE would not be executed if a file named TESTFILE already existed on drive 0.

Saving Program Files

SAVEs performed with the DFH Editor in BASIC mode will produce program (PRG) files. In BASIC mode the range parameter can be used to specify an address range of memory to be saved. When used in this way, it is called a *directed* SAVE. In BASIC mode the syntax of the SAVE command can be shown as:

```
←@dr:filename,$xxxx yyyy
```

Where: xxxx = Start address (Hex); yyyy = End address plus one (Hex).

The range parameter ($xxxx yyyy) is optional. If it is not used, the complete BASIC program in memory will be saved.

The directed SAVE is intended for use in saving machine language programs. Except for special purposes, such as program relocation, BASIC programs should not be saved using the range parameter.

An example of the SAVE command using the range parameter in BASIC mode to save a machine language program is

```
←@1:TESTPROG,$7000 72AE
```

Note that the leading dollar sign ($) is omitted for the ending address. This command would save the contents of the computer memory from hexadecimal address $7000 through $72AD as a program file named TESTPROG. In this example the optional @ has been included so the SAVE will automatically replace any existing file named TESTPROG on drive 1.

Verify a File

Syntax:] dr:filename,qualifier

Except for the primary command character], the syntax for this command is exactly the same as the syntax for the LOAD command.

Use this command to verify (compare) the contents of a file on disk against the contents of the computer's memory. Its operation is exactly the same as the LOAD command except that the memory contents are not changed. Please refer to the LOAD command for a complete description of the syntax.

The message ?VERIFY ERROR will be displayed if the file on disk is not exactly the same as the file in memory.

Chapter 6

DFH Applications Examples

Why Samples?

You now should have a good idea of the power of the DFH system. But the discussion so far has been about the general use and functions of DFH. Many possible uses perhaps came to mind as you read and experimented with the system. Indeed, you may have already entered some data and begun to use DFH. We suspect that some readers will be making extensive use of the system long before they read this chapter.

The purpose of this chapter is to illustrate, through examples, the features of the DFH programs and how they can be applied to solve your data-handling problems. The examples were chosen to show the use of a wide variety of DFH program features. The main emphasis is placed on understanding the reasons each feature is used and how alternate methods could be used to obtain different results.

The DFH programs were not designed to solve some narrowly defined problem. They are intended to assist you in handling data files for almost any purpose. Thus, we cannot hope to guess exactly what you want to do, or direct you step by step in an exact procedure to obtain the results you want.

To get the most out of this chapter, you should have some understanding of how to use the various DFH functions. This is explained in the chapter on operating procedures. In the applications examples, the DFH functions are usually mentioned in a general way without mentioning the step-by-step operations needed to perform them.

A Magazine Cardfile System

This application example was chosen to demonstrate how a number of specialized files and printouts can be obtained from a single data base. Many of the capabilities of the DFH programs will be used in combination with each other.

The objective for this example is to create a cardfile system for the articles contained in several monthly magazines. The primary requirement is that the system must eliminate most of the time normally spent in searching for a particular article or subject.

Some ways the system will be used are:

- To find the location of articles dealing with a particular primary subject.
- To review the subjects of individual magazines in chronological order.
- To find the location of articles dealing with a particular secondary subject. This could be a more time-consuming search, but should be quite accurate.

There are some mistakes which are often made in a situation such as this. One mistake is trying to create the ultimate system. Simplicity is much more important than an elaborate result. If a particular feature is not easy to implement and use, or if it does not provide a distinct benefit, it should not be built into the system.

Another mistake is losing sight of the fact that total time is important, not the time for any one task. There is no point in saving five hours a month of lookup time if it takes five hours each month to maintain the data base for the system.

The third mistake is designing an on-line system when it is not needed. On-line systems are fine for businesses where the computer is left running the same program all day. In the home you may want to use your cardfile system while the computer is tied up running some other program.

For example, you might be using a word processor program to write a report and need to locate a reference article for information to go in the report. Chances are, you won't shut down the word processor to activate the cardfile system. That situation is self-defeating. You have a system that you can't use when you need it.

The Necessary Elements

The list of really necessary data for the system was narrowed down to:

1. Magazine identification: This was reduced to a code number to save typing time and file space. It really isn't too difficult to keep a cross-reference sheet handy showing that 01 means *National Geographic,* and so on, especially when there may be only a few items in the list.
2. Date: A four-digit system was chosen with year first, then month. This allows the dates to sort correctly without any special processing. They may be a little hard to read at first, but remember, it's total time we are trying to save.
3. Page number: The only special consideration here is to pick a standard number of digits and pad with leading blanks or zeros. Again, this is to avoid the need for any special sorting requirements.
4. Subjects: Here keywords are used rather than descriptions. The primary keyword is listed first, and followed by as many secondary keywords as needed.

Titles of the articles are not included because they can be long, and we didn't like the idea of all that extra typing every month. Also, you are not likely to remember the exact title anyway.

A few sample records, representing three different magazines, are shown below:

```
1010 "01!8405!557!volcanoes,archeology!
1020 "01!8405!626!krill,ocean,marine life!
1170 "02!8404!048!stocks,investments!
1180 "02!8404!140!disk,1541,scratching!
1210 "03!8405!038!investment,sociology!
1220 "03!8405!122!herbs,natural health!
```

At this point you might question whether the four data fields are in the proper order. Actually, there is no correct order. The best way to set up the data depends on how you intend to use it. However, the DFH programs allow you to restructure the data anytime you wish, so you should not worry too much about how you set it up in the beginning. We will explore an alternate order a little later on.

Maintaining the Data File

Since the data file contains only the minimum essential information, it is quite easy to maintain. The editing feature of the DFH programs is used to add new information to the file once a month, or at any other time you wish.

Taking advantage of the repeating data entry capability, the magazine ID and the date are entered with only a carriage return. The page number is no problem, and only the choice of keywords requires any thought.

Descriptive keywords can be obtained directly from the titles of some articles. For others, the title will not be descriptive, and you will need to scan the article to see what it is about.

Often an article will cover more than one subject. If you can't decide on a single primary keyword because several seem equally important, you may want to create more than one data record for the article. Remember that creating this type of data file is an individualized operation. You must select keywords in a manner that means something to you, never mind the rest of the world.

Although we decided to use a primary and secondary keyword system for this example, you might just as logically decide to use only one keyword per record and create multiple records when more than one subject is covered. We felt that this took up too much file space for the added benefit it provided, but you should choose a method that is best for you.

In any case, using a keyword system is efficient because you don't have to read the entire article, and the keywords you choose will usually mean more than the title of the article. Most important of all, the time you spend keeping the data current can be held to a minimum.

Locating an Article

Our most common use of the cardfile system is to locate an article or a group of articles based on the primary keyword. We can do this using a printed output list that is updated once a month.

When the magazines for the month have all arrived, we have a short file-updating session. The DFH add and edit features are used to enter all the new data records into the data file. The file is sorted on field 4 to put it in order according to the primary keywords, and then it is printed.

The printing format was set up and saved when we first started using the system. At first we would change the date in the print format file each month. Later we realized that simply discarding the previous copy would insure that we were using the latest version. Always look for ways to avoid work.

The printed output from this operation looks something like the following sample:

```
Magazine Subject List        Page 1
Mag   Date    Page    Subjects
02    8307    160     disk,backup
02    8404    140     disk,1541,scratching
03    8405    122     herbs,natural health
03    8405    038     investment,sociology
01    8405    626     krill,ocean,marine life
02    8311    271     programming,function key
02    8301    192     programming,input
02    8307    224     programming,usr,function
02    8404    048     stocks,investment
01    8405    557     volcanoes,archeology
```

Using this list, you can quickly locate all the articles dealing with any primary subject without regard to when they were published or which magazine they were in.

This one list satisfies most of our needs for locating information on any subject. The remaining needs take a variety of forms, most of which can be satisfied by some small additional processing.

Reference by Magazine

For some types of research it is useful to have a separate subject list for each magazine with the entries sorted by date. Working from the master data file, these lists can be prepared quickly and easily.

The first step is to sort the master file on field 0. This is a special sorting designation which causes sorting of entire records without regard to field delimiters. This sorting step groups the records for each magazine together. Within each group the records are sorted by date, and within each date they are sorted by page number. This order of sorting is a direct result of the ordering of the data within the data fields.

Next, the DFH split function is used to examine the contents of field 1 and create (split off) a separate file for each of the magazine code numbers. These files should look similar to the following examples:

File 01:

```
  1000 "!←@0:magfile.001
  1010 "01!8405!557!volcanoes,archeology!
  1020 "01!8405!626!krill,ocean,marine life!
```

File 02:

```
  1000 "!←@0:magfile.002
  1010 "02!8404!048!stocks,investment!
  1020 "02!8404!140!disk,1541,scratching!
```

File 03:

```
  1000 "!←@0:magfile.003
  1010 "03!8405!038!investment,sociology!
  1020 "03!8405!122!herbs,natural health!
```

All that remains is to print these files. Each one can be printed with the full magazine title in the heading. The date, page number, and keyword fields need to be printed, but there is no point in printing the magazine ID code.

The printouts look like the following sample:

```
Subject Index for COMPUTE!   05-25-84 Page 1
Date   Page   Subject
8301   192    programming,input
8307   160    disk,backup
8307   224    programming,usr,function
8311   271    programming,function key
8404   048    stocks,investment
8404   140    disk,1541,scratching
```

You may have noticed that we did not re-sort the master data file back into its normal order during this process. That was not an oversight. The file will be re-sorted after the next month's additions are made, and nothing is gained by sorting it twice. Again, keep looking for ways to avoid work.

Reference by Date

Once in a while you may want to look at a cardfile listing which is organized by date without regard to the source of the articles. This can be useful for spotting trends. For example, it might reveal that more and more articles on a particular subject are being published by more than one magazine.

The data file was not originally set up to support this particular operation, but the DFH restructuring function will allow us to quickly change the order of the data fields. What we need to do is copy the data in the original file which is organized as:

Magazine Date Page Subjects

into a new data file with the data organized as:

Date Subjects Magazine Page

Using the restructuring program, we would place data from old field 2 in new field 1, data from old field 4 in new field 2, etc.

The new data file is then sorted on field 0. This puts the records in order by date. Within identical date groups, the order will be according to subject keywords. When both the date and primary keyword are identical, the ordering will be by secondary keywords. A sample section of the new file might appear as:

```
1010 "8301!programming,input!02!192!
1020 "8307!disk,backup!02!160!
1030 "8307!programming,usr,function!02!224!
1040 "8311!programming,function key!02!271!
1050 "8404!disk,1541,scratching!02!140!
1060 "8404!stocks,investment!02!048!
1070 "8405!herbs,natural health!03!122!
1080 "8405!investment,sociology!03!038!
1090 "8405!krill,ocean,marine life!01!626!
1100 "8405!volcanoes,archeology!01!557!
```

This new ordering of the data fields allows the records to be sorted into a sequence which satisfies our needs. Note that the columns of printed output need not follow this new order. The DFH printing function can print the data fields in any order you like.

We have found that the printed output seems easier to use when the column order is the same for all printed lists. When we first produced a satisfactory printed output for this file arrangement, we saved the print format file. Now it can be recalled and used whenever it is needed.

Another timesaver—for a special-purpose file like the one we have just described, there is no reason to keep the file after you have printed a list from it. At that time it has served its purpose and should be scratched.

This illustrates a trap you should always avoid: having two files to update when one will do. Every time you sit down at the keyboard to enter or edit data, you run the risk of making errors. The more typing you do, the more errors you make, so avoid typing whenever possible.

A Single Subject List

On some occasions, you may want to do an exhaustive search for all articles about a particular subject without regard for whether it was the primary subject or one of the secondary subjects. The EXTRACT function can accomplish this very easily.

The EXTRACT function can operate in two ways. You can look for a word occurring in a specified place within a field, or for a word occurring anywhere within a field. The latter method is what we should use here, because we don't know whether the keyword we specify will be primary or secondary.

All the records that contain the specified keyword in field 4 (the subjects field) will be saved in a new file.

This new file can be sorted in any way that suits your needs, or it may not need to be sorted. If the master file was already sorted by keyword (field 4) when the EXTRACT was performed, the extracted records will still be in that order. If that is the order you want, all you need to do is print the file. You can probably use one of your previously prepared printing formats.

Another possibility is to do another EXTRACT operation on the new file using a different keyword. The resulting file would have only those records which contain both keywords.

As you can see, there is an almost endless variety of ways you can process even a simple data base like the one in this example. That is one of the real advantages of the DFH programs. They do not presume to know what you want to do. They simply provide the tools for you to use in obtaining any result you desire.

A Genealogy File

Some of your data file handling and processing problems will be difficult, but even seemingly impossible jobs can often be handled by careful planning.

The Genealogy File is used as an example because it requires handling large amounts of complex and unstructured data. The problems that must be solved will test some limits of the DFH programs.

Several new data-handling concepts will be introduced. These are important because the underlying ideas can be applied to many other situations. One of these concepts is the creation of data fields that are used to control the overall data-processing tasks. In many cases the data contained in such a field will never be included in the final printed output.

Two data control fields will be used in this example. An identification (ID) field is used to control data sorting, and an operations (OP) code field is used to direct the operations of a separate printing program.

It is worth noting that selecting the data organization and handling procedures to produce the genealogy listing shown in this example was not a simple task. Various options were explored and abandoned. The question "Can this be done?" was asked more than once. When you start working with your own complex data sets, don't be surprised if you have the same doubts, but don't give up too soon. The results can be very gratifying.

The Desired Results

The desired final output from the Genealogy File is a printed listing as shown in the following sample. (This may or may not be the best way to present a family tree; but it works, and it was the desired form.)

There is a separate section containing the data set for each person. Each section starts with the ID number and name of the person, and contains all the information about that person.

The numbers down the right side are IDs for the immediate relatives (parents, spouse, and children) mentioned within the data set. The data sets are in order by ID so that the information for any person can be located quickly and easily.

```
1120 HENRY STANDAGE (M*)
                    -BECAME U.S. CITIZEN 13 NOV 1855
                    -MEMBER OF THE MORMON BATTALION
     ADDRESS:       -CHILDHOOD IN PETWORTH ENGLAND
                    -MOVED TO AMERICA IN 1835
                    -MOVED TO UTAH IN 1847
     BORN:          26 FEB 1818 LONDON ENGLAND
     OCCUPATION:    STOREKEEPER AND FARMER
     FATHER:        WILLIAM STANDAGE                        #1050
     MOTHER:        ELIZABETH SARAH (HOWARD) STANDAGE       #1060
     DIED:          08 MAY 1899 MESA ARIZONA
     BURIED:        ? ? ? MESA ARIZONA
     MARRIED:       SOPHRONIA ARMENIA SCOTT                 #1130
                    16 APR 1845
                    -MAYBE 13 APR 1845
     CHILDREN:      -NONE-
     ADOPTED:       RANSON ROSS REEVES
     MARRIED:       HENRIETTA ROGERS                        #1140
                    16 APR 1851 SALT LAKE CITY UTAH
     CHILDREN:      WILLIAM NOAH STANDAGE (M)               #1160
                    EDA ELIZABETH STANDAGE (F)              #1180
                    HANNAH MARINA STANDAGE (F)              #1200
                    SARAH CAROLINE STANDAGE (F)             #1220

1130 SOPHRONIA ARMENIA (SCOTT) STANDAGE (F-)
     BORN:          ? ? 1821
     DIED:          01 JUL 1896 MESA ARIZONA
                    -BROKEN NECK FROM FALL
     MARRIED:       HENRY STANDAGE                          #1120

1140 HENRIETTA (ROGERS) STANDAGE (F-)
     BORN:          30 MAY 1832 SHELBYVILLE OHIO
     FATHER:        NOAH ROGERS
     MOTHER:        EDA (HOLLISTER) ROGERS
     DIED:          11 OCT 1898 MESA ARIZONA
     BURIED:        12 OCT 1898 MESA ARIZONA
     MARRIED:       HENRY STANDAGE                          #1120

1150 WILLIAM STANDAGE (M*)
     ADDRESS:       -POSSIBLY EMIGRATED TO AMERICA IN 1834
                    BUT HIS BROTHER HENRY BELIEVED HIM TO
                    BE IN GLASGOW SCOTLAND IN 1860
....
....
```

Defining the Problems

By examining the desired output, several problems can be identified. Let's look at these problems and the general techniques that are used to solve them.

First, it is obvious that the amount of information in each data set can be quite large and the maximum amount cannot be predicted. Also, there is nothing in the genealogy data itself that can be used to identify the records belonging to a particular data set. Since sorting will be performed, some data must be added to each record to hold each data set together. In this example the identification (ID) number is added for that purpose.

When you are forced to add data to a file, you should try to get the maximum possible benefit from that data. The ID number provides a second benefit by becoming the primary means of finding the data for each person in the file.

The second problem is that most of the information in each data set is optional. This forces the use of identifying headings (BORN:, FATHER:, MOTHER:, etc.) to produce a useful printed output. But these headings would take up a lot of space in the data file, and rather than solving any problems, they would become just another complicating factor.

This brings up the general principle that a data file should not contain any unnecessary data. It is not obvious how to apply this principle until we look at the third problem.

The third problem is that the printed output we want is too complex to be conveniently handled by the DHF printing routines. As it turns out, a separate printing program will solve the complex print format problem and will allow us to include a very small amount of information in the file which will solve the headings problem and provide complete data sorting control. For this example the added data will be called the operations (OP) code.

When there is this much to be gained, it is worthwhile to consider writing your own auxiliary processing programs to work with the DFH system files. We have chosen that option for this applications example to illustrate the general procedures involved.

The File Format

Let's take a look at the data records which produce the first few lines of the sample output:

```
2050 "1120!100!HENRY STANDAGE (M*)!!
2060 "1120!101!-BECAME U.S. CITIZEN 13 NOV 1855!!
2070 "1120!102!-MEMBER OF THE MORMON BATTALION!!
2080 "1120!110!-CHILDHOOD IN PETWORTH ENGLAND!!
2090 "1120!111!-MOVED TO AMERICA IN 1835!!
2100 "1120!112!-MOVED TO UTAH IN 1847!!
2110 "1120!120!1818-02-26!LONDON ENGLAND!
```

```
2120 "1120!130!STOREKEEPER AND FARMER!!
2130 "1120!150!WILLIAM STANDAGE!1050!
```

In the first field of each record is the number 1120. This is the identification (ID) number for HENRY STANDAGE. This is followed by a three-digit operations (OP) code number. Together these numbers provide the basis for all sorting that will be required.

The OP code is also used to tell the auxiliary printing program how to handle the information in the remaining two fields. Notice that whenever the OP code ends in 0, a heading of some kind is printed. For example, the 100 code causes the printing of a blank line followed by the ID number and the subject's name, while the 110 code triggers the ADDRESS: heading.

Now compare the record containing the 120 OP code with the BORN: line in the sample output. See how the date has been altered prior to printing? Dates are easy to sort if they are represented by numbers in year-month-day order. Since people don't read that form very well, we simply have the print program do a conversion to keep everyone happy.

Two points are being made here. First, by writing an auxiliary program, you can process more complex data than would be possible with the DFH programs alone. Second, you can simplify your processing tasks with techniques like our OP code system.

Operations Codes

The following list shows all of the possible operations codes that could be used in field 2. These codes dictate what type of information should be placed in field 3 and field 4. Special headers that will be printed in response to some codes are shown in parentheses in field 3. Field 1 is not shown because it always contains the ID number.

FIELD 2	**————————FIELD 3**	**————————**	**FIELD 4**
100	**NAME**		
101–109	**misc. personal info.**		
110	**(Address:)**	**TEXT**	
111–119	**misc. residence info.**		
120	**(Born:)**	**DATE**	**PLACE**
121–129	**misc. birth info.**		

130	**(Occupation:)**	**TEXT**	
131–139	**misc.**		
140–149	**RESERVED**		
150	**(Father:)**	**NAME**	**ID number**
151–159	**misc.**		
160	**(Mother:)**	**NAME**	**ID number**
161–169	**misc.**		
170	**(Died:)**	**DATE**	**PLACE**
171–179	**misc. cause, etc.**		
180	**(Buried:)**	**DATE**	**PLACE**
181–189	**misc.**		
190–199	**RESERVED**		
200	**(Married #1:)**	**NAME**	**ID number**
201–209	**NOT USED**		
210		**DATE**	**PLACE**
211–219	**misc.**		
220–229	**RESERVED**		
230	**(Children:)**	**NAME**	**ID number**
231–239	**NAMES**	**ID number**	
240	**NOT USED**		
241–249	**children cont.**	**NAMES**	**ID number**
250	**(Step:)**	**NAME**	**ID number**
251–259	**children**	**NAMES**	**ID number**
260	**(Adopted:)**	**NAME**	**ID number**
261–269	**children**	**NAMES**	**ID number**
270–299	**RESERVED**		
300–899	**Used like 200–299 for additional marriages.**		
900–999	**RESERVED**		

The Printing Program

The auxiliary printing program that was used for genealogy files in this example is shown in the following listing. When this program is used with data files like the ones just described, it will produce printed output in the form shown in this example.

Since this program is intended as an example rather than a finished product, it does not contain any error-checking features. You should feel free to modify it (and also the data format) to satisfy your own needs and desires.

This program is not intended for general use with the other DFH programs. The DFH PRINT program is for general use. This example is meant as an illustration of the type of specialized printing that is possible.

Sample Auxiliary Printing Program

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1000 REM SAVE "@0:EX.G.PRINT",8                :rem 243
1010 :                                          :rem 252
1020 REM"- AN EXAMPLE PROGRAM TO FORMAT AND PRINT
     {SPACE}GENEALOGY -"                         :rem 95
1030 REM"- FILE DATA PREPARED BY THE 'DFH' PROGRAM
     S -"                                       :rem 218
1040 :                                          :rem 255
1050 REM"-- TOP OF MEMORY = $7900 TO PROTECT SUBRO
     UTINES --"                                  :rem 91
1060 IF PEEK (65534)=72 THEN 1080: REM"-- C64 COMP
     UTER --"                                   :rem 163
1070 POKE 52,0: POKE 53,121: GOTO 1110: REM"-- PET
      --"                                       :rem 128
1080 : POKE 55,0: POKE 56,121: REM"-- C64 --"
                                                 :rem 79
1090 :                                            :rem 4
1100 REM"-- TEST/INSTALL M.L. SUBROUTINES --"
                                                :rem 124
1110 : IF PEEK(30977)=21 AND PEEK (30980)=30 THEN
     {SPACE}1160                                :rem 144
1120 PRINT "{RVS} LOADING DFH SUBS$79 {OFF}"
                                                :rem 218
1130 CLR : LOAD "DFH SUBS$79",8,1               :rem 168
1140 :                                            :rem 0
1150 REM"--PROGRAM TITLE & INITIALIZATION--"
                                                :rem 139
1160 : CR$=CHR$(13): FT%=0: FA$="DA"            :rem 255
1170 RB$="{RVS}{39 SPACES}{OFF}"+CR$            :rem 138
1180 PRINT "{CLR}";RB$;"{RVS}{2 SPACES}G.PRINT
     {20 SPACES}05-19-84{2 SPACES}{OFF}";CR$;RB$;
                                                :rem 253
1190 PRINT "{RVS}{4 SPACES}A PROGRAM TO PRINT FAMI
     LY TREE{5 SPACES}{OFF}"                    :rem 187
1200 PRINT "{RVS}{4 SPACES}(GENEALOGY) DATA FILES.
     {12 SPACES}{OFF}";CR$;RB$;                   :rem 1
1210 PRINT "{RVS}{4 SPACES}---- REQUIRES DFH SUBS$
     79 ----{5 SPACES}{OFF}";CR$;RB$;"{DOWN}"
                                                 :rem 43
1220 :                                          :rem 255
1230 DIM DA$(10),MO$(12),HD$(30)                 :rem 43
1240 FOR JJ=0 TO 12: READ MO$(JJ): NEXT JJ:rem 189
```

```
125Ø DATA " ? ","JAN","FEB","MAR","APR","MAY","JUN
     "                                        :rem 66
126Ø DATA "JUL","AUG","SEP","OCT","NOV","DEC"
                                              :rem 172
127Ø S1$="{5 SPACES}"                         :rem 243
128Ø HD$(11)=S1$+"ADDRESS:{4 SPACES}": HD$(12)=S1$
     +"BORN:{7 SPACES}"                       :rem 31
129Ø HD$(13)=S1$+"OCCUPATION: ": HD$(15)=S1$+"FATH
     ER:{5 SPACES}"                           :rem 157
13ØØ HD$(16)=S1$+"MOTHER:{5 SPACES}": HD$(17)=S1$+
     "DIED:{7 SPACES}"                        :rem 2Ø8
131Ø HD$(18)=S1$+"BURIED:{5 SPACES}": HD$(2Ø)=S1$+
     "MARRIED:{4 SPACES}"                     :rem 167
132Ø HD$(21)=S1$+"{12 SPACES}": HD$(23)=S1$+"CHILD
     REN:{3 SPACES}"                          :rem 245
133Ø HD$(25)=S1$+"STEP{8 SPACES}": HD$(26)=S1$+"AD
     OPTED:{4 SPACES}"                        :rem 241
134Ø PH$="GENEALOGY DATA FOR 'STANDAGE' FAMILIES"+
     CR$                                      :rem 1ØØ
135Ø PH$=PH$+"⁅37 T⁆"                         :rem 124
136Ø :                                        :rem 4
137Ø REM"--- START OF MAIN PROGRAM ---"       :rem 97
138Ø :                                        :rem 6
139Ø OPEN 4,4: GOSUB 172Ø: REM"--- PAGE HEADING --
     -"                                       :rem 148
14ØØ : INPUT "DATA FILE NAME{6 SPACES}";IL$: OPEN
     {SPACE}8,8,8,"Ø:"+IL$+",S,R"             :rem 184
141Ø INPUT# 8,FD$: FD$=LEFT$(FD$,1): TT=ST: GOTO 1
     44Ø                                      :rem 93
142Ø :                                        :rem 1
143Ø : PRINT# 4,CR$;: NL=NL+1: IF NL>58 THEN GOSUB
      171Ø                                    :rem 147
144Ø : IF TT<>Ø THEN 176Ø                     :rem 225
145Ø INPUT# 8,DA$(Ø): TT=ST: SYS 3Ø979: CD=VAL(DA$
     (2))                                     :rem 116
146Ø CP%=(CD+.5)/1Ø: IF CP%*1Ø<>CD THEN PRINT# 4,H
     D$(21);: GOTO 16ØØ                       :rem 219
147Ø :                                        :rem 6
148Ø REM"--- PRINT PRIMARY CODE LINES ---"    :rem 74
149Ø :                                        :rem 8
15ØØ IF CP%<>1Ø THEN 153Ø                     :rem 224
151Ø PRINT# 4,CR$;CR$;: NL=NL+2: IF NL=>58 THEN GO
     SUB 171Ø                                 :rem 138
152Ø PRINT# 4,DA$(1);" ";DA$(3);: GOTO 143Ø:rem 96
153Ø : IF CP%>29 THEN CP%=CP%-1Ø: GOTO 153Ø
                                              :rem 153
154Ø :                                        :rem 4
155Ø PRINT# 4,HD$(CP%);                       :rem 2Ø7
156Ø IF CP%=12 OR CP%=17 OR CP%=18 OR CP%=21 THEN
     {SPACE}166Ø                              :rem 165
```

```
157Ø :                                           :rem 7
158Ø REM"--- ALLIGN & PRINT ID#'S ---"         :rem 152
159Ø :                                           :rem 9
16ØØ : PRINT# 4,DA$(3);: SP=18+LEN(DA$(3)): IF SP>
     6Ø THEN SP=6Ø                               :rem 9
161Ø IF DA$(4)<>"" THEN PRINT# 4,SPC(62-SP);"#";DA
     $(4);                                      :rem 55
162Ø GOTO 143Ø                                 :rem 2Ø2
163Ø :                                           :rem 4
164Ø REM"--- PRINT DATE ENTRIES ---"           :rem 198
165Ø :                                           :rem 6
166Ø : PRINT# 4,RIGHT$(DA$(3),2);" ";MO$(VAL(MID$(
     DA$(3),6,2)));                             :rem 11
167Ø PRINT# 4," ";LEFT$(DA$(3),4);"{2 SPACES}";DA$
     (4);: GOTO 143Ø                           :rem 232
168Ø :                                           :rem 9
169Ø REM"--- PRINT PAGE HEADING ---"           :rem 16Ø
17ØØ :                                           :rem 2
171Ø : FOR JJ=NL+1 TO 66: PRINT# 4,CR$;: NEXT JJ
                                                :rem 11
172Ø : PRINT# 4,PH$;CR$;CR$;: NL=3: RETURN :rem 81
173Ø :                                           :rem 5
174Ø REM"--- TEST FOR MORE FILES ---"          :rem 2Ø7
175Ø :                                           :rem 7
176Ø : CLOSE 8: PRINT "END OF FILE"            :rem 77
177Ø INPUT "MORE FILES TO PRINT{3 SPACES}Y{3 LEFT}
     ";KB$: IF LEFT$(KB$,1)="Y" THEN 14ØØ :rem 218
178Ø PRINT# 4: CLOSE 4: END                    :rem 169
```

Chapter 7

File Conversion

Converting Non-DFH Files

There may be times when you want to use the DFH programs on data files that were not produced by the DFH programs. The structure of those files will probably not be the same as a DFH file. However, there is a good chance that they can be converted to the DFH structure. The amount of effort required will obviously depend on the degree of similarity between the two structures.

The DFH programs are quite flexible about what structures they can read. If the minimum requirements for DFH files are met, the existing files can be read and a standardized output file will be produced.

Minimum Requirements

The minimum requirements for a data file to be read and processed by the DFH programs are:

1. The first character in the file other than a quote (the first readable character) will be used as the delimiter for the file. All other characters in the first record are assumed to be file identification information, not data, and will be ignored.
2. The delimiter must not be a numeric character or a quote.
3. The first data record must contain at least as many fields as any other record in the file. Most of the DFH programs count the delimiters in the first data record to determine how many fields are being used.
4. A quote must be the first character of all data records which contain shifted characters, control characters, commas, or colons. All DFH processing programs use the INPUT# command, so the quote is necessary to handle special characters.
5. Embedded quotes are not allowed in any record.
6. No record should contain more than 74 characters. This permits the addition of line numbers while remaining within the 80-character limit for onscreen editing and input operations.

If these minimum requirements are not already met by the existing file, some conversion will be required. Some files can be converted quite easily using only the DFH Editor commands, while others may require a BASIC program to assist in completing the conversion.

In either case, you should perform a preliminary conversion using the DFH Editor. This will insure uniformity within the file and will allow you to write much simpler BASIC conversion programs if they are needed.

Preliminary Conversion

Because the DFH Editor is a general data file editing utility, it is well-suited to making the types of changes implied by the minimum requirement rules.

The following procedure can be used as a preliminary conversion process on almost any sequential data file. In the beginning, you should follow it step by step in the order it is presented. As you gain more experience and understanding of the processes involved and the reasons for them, you may want to alter the procedure to be more efficient for individual situations.

It is very important that you not attempt any onscreen manual editing until the preliminary conversion process has been completed. There are some situations where premature editing, including a simple RETURN keypress on a data line, could severely alter the data contained in the file. Once you understand the ways you can lose data, you can violate this rule. Until then, play it safe and don't edit too early.

Step 1: Load and Examine:

Activate the DFH Editor, load the file you want to convert, and list the first few lines.

The most important thing this does is to insure that the DFH Editor is able to handle the file. The DFH Editor cannot handle files which have records containing more than 250 data characters, are too large to fit into the computer memory, or contain any record terminated by more than one carriage return character.

Due to a peculiar problem in the model 2040 disk units, the DFH Editor had to be programmed to interpret two consecutive carriage return characters as being the same as an end of file.

If any of these problems exist, they must be corrected by a preprocessing program before continuing the conversion process.

Another reason for listing a few lines is that even a quick examination can give you some idea of the type of data contained in the file and the general method of data organization.

Step 2: Find Two Unused Characters:

Use the ;FI (Find) command to find two characters which are not used in the file. They will be used during the conversion and then removed. For easy reference we will call these unused characters U1 and U2.

If you are checking for the character *, the command would be

```
;FI/*/
```

If no data records are displayed in response to this command, then * is not used in the file. This process can be aided by simply looking at some of the data. For example, if it appears there are no shifted characters, you might try to find two shifted characters that are not used in the file.

The two unused characters you select will only be used on a temporary basis during the preliminary conversion. Since they will be removed later, there should be no concern about their contaminating the file data.

Step 3: Replace Embedded Quotes:

Use the ;FC (Find and Change) command to find all quotes and change them to U1. If the U1 character was an *, this command would be

```
;FC/"/*/
```

This is only the first step in eliminating all embedded quotes. It will not be completed until much later in the process. Don't worry that we also seem to be eliminating leading and trailing quotes. Those cases will be corrected later.

Step 4: Insert Leading Quotes:

Use the ;QT (Insert Leading Quotes) command to insert a quote as the first character of each record. A single execution of this command, when used without its range parameters, will perform the needed quote insertions for every record in the file.

While the ;QT command is executing, you might notice error messages indicating that there are more than 74 characters in some of the records. This problem will be handled later. For now you should simply ignore the long-line error messages.

Step 5: Remove Leading U1 Characters:

Use the ;FC (Find and Change) command to find all cases where the new leading quotes are immediately followed by the U1 character. The purpose of this step is to restore a single leading quote to all the records which originally contained leading quotes. (Recall that we changed all quotes to U1 in step 3.)

If the U1 character was *, this command would be

```
;FC/"*/"/
```

The only U1 characters remaining in the file after this step will be replacements for embedded or trailing quotes.

Step 6: Protect Trailing Spaces:

Use the ;AD (Add Final Character) command to add the U2 character to the end of all records in the file.

This step has two purposes. First, it provides a temporary character to protect any trailing spaces in the data records. (This temporary character will later be replaced with a delimiter character.) Second, it provides a means of locating and removing any U1 characters which were originally trailing quotes.

If the U2 character was %, this command would be

```
;AD %
```

Step 7: Remove Trailing Quotes:

Use the ;FC (Find and Change) command to replace all U1-U2 character combinations with U2 characters. At this time the U1-U2 combination can exist only where the file originally contained a trailing quote.

If the U1 character was * and the U2 character was %, this command would be

```
;FC/*%/%/
```

After this step, the only U1 characters remaining in the file will be the ones used to temporarily replace embedded quotes. All U2 characters will be at the end of the data records. In some cases, they will have replaced trailing quotes which are not needed in the DFH files.

Step 8: Install Delimiters:

This step must be done in different ways depending on whether the original data file structure used delimiter characters.

If delimiters were not used in the original file, you must pick one. Use the ;FI (Find) to help you find a character not already used in the file. Then, use the ;FC (Find and Change) command to replace the U2 characters with your chosen delimiter character.

If the file contained multifield records using delimiters to mark the field boundaries, you must execute two commands to complete this step. First, use the ;FC command to replace all delimiter-U2 combinations with the delimiter character. Then use the ;FC command again to replace all remaining U2 characters with the delimiter character.

You may find some files that use a group of characters in a particular sequence as the delimiter. Once they are identified, these multiple-character delimiters can be converted exactly like the single-character versions. Just remember that such groups must ultimately be reduced to single characters because the DFH programs will accept only single-character delimiters.

Step 9: Fix the Start of the File:

Install a first record with the delimiter as the first nonquote character. Remember that the DFH programs expect the file data to begin with the second record in the file.

If the first record is already being used for some type of file identification, you can simply insert the delimiter as the first character of that record.

If the first record contains file data, you must add a completely new first record containing the delimiter as its first character. Although the delimiter is the only required item, you might want to create a DFH standard first record at this time:

```
900 "!←@0:filename
```

As illustrated here, the ! is the file delimiter. The left arrow character is the DFH Editor's SAVE command. The advantage of installing a complete first record is that it can be used to execute file SAVEs without the worry of typing errors. As you become more comfortable with file conversions, you may wish to create a standard first line very early in the conversion process. Having such a first line makes it easy to save the file after every few conversion steps.

Sometimes you may find files that use more than one record for identification, setup, or other nondata information.

In those cases the extra records must be deleted because the DFH programs would treat them as data.

Step 10: Fix Embedded Quotes:

Use the ;FI (Find) command to locate all remaining U1 characters. These characters mark the spots where embedded quotes existed in the original file. They must be handled in some logical manner, but exactly what to do depends on the nature of the data in the file and how you intend to use it.

There are cases where embedded quotes can simply be removed with no loss of meaning. Sometimes, where the data is textual in nature, they might be replaced with apostrophes (') and still convey the same meaning. The only firm rule is to think the problem through before acting because you are changing the actual data in the file at this step, not just conditioning it for use with the DFH programs.

Step 11: Check for Long Records:

This final step is to check for, and possibly fix, records containing more than 74 data characters.

The ;QT (Insert Quotes) command provides a way to do this. When used in its simple form ;QT, it will now display only records containing more than 74 data characters. Remember, we have already insured that there are no embedded quotes which it could find as errors, and all the records already contain leading quotes, so no installation displays will be shown. That leaves only the possibility of long-line errors to be displayed.

If no long records are found, you are through with the preliminary conversion process, and perhaps through with the complete process, as we will see in a moment.

Editing Long Records

If only a few long records are found, you should take the time to examine them closely. It is often possible to reduce the number of characters in a line without altering the meaning of the data.

These alterations can be done with normal onscreen editing techniques, but this is not recommended because all displayed characters in excess of 80 (including line numbers and spaces) will be lost at the first RETURN and will require manual reentry after the record has been shortened.

A much better way to edit long records is by using the

;FC command with a range parameter specifying the single line to be altered. For example, a command to change CALIFORNIA to CA in line 2350 could be:

;FC/CALIFORNIA/CA/,2350

As the change is made, the changed record will be displayed and automatically checked to see if it is still too long.

If a change such as the one just described can be applied to the complete file, simply use the ;FC command without the range parameter. This can sometimes provide significant reductions in overall file size—usually a good objective.

The remaining possibility for long records is that there might be a large number of them which cannot easily be corrected by manual editing. The general cases of splitting long records and combining short records will be discussed in the next two sections, and will involve writing separate BASIC conversion programs.

Not all long record problems can be solved. Occasionally files with long records simply can't be split into shorter records. Fortunately, that should not happen very often.

Combining Records

If you have done the preliminary file conversion described in the preceding paragraphs, you may not need to do any more. If there are no long records remaining in the file and if the data is organized the way you want it, the conversion is complete.

The previous discussion assumed that the fundamental structure of the old file placed all items of each set of data in a single record. Obviously, that will not always be the case.

In the general discussion on file structures, we noted that one of the most common structures was the simple *one data item = one file* record method. The problem of how to group the data into sets was sidestepped at that time by implying that the program used to process the data would also handle the grouping, or that the groups (records) would have been created correctly if the data was to be handled by the DFH programs.

Now we are looking at a different situation. We are faced with data records that already exist and need to be grouped into new, larger, multifield records so that the data can be handled by the DFH programs.

One of the best ways to accomplish this is to write a BASIC program that will group each data set into a multifield record. Each field would then contain an individual data item (record) from the original file. The most important thing you must know is how many data items it takes to make a set. If this number is the same for all the sets in the file, a simple program like the one shown below will do the job just fine.

Converting Files 1

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:EX.F.CONV-1",8               :rem 249
1Ø1Ø :                                          :rem 252
1Ø2Ø REM"- AN EXAMPLE PROGRAM TO COMBINE GROUPS OF
      SHORT RECORDS -"                          :rem 175
1Ø3Ø REM"- INTO LONGER, MULTI-FIELD RECORDS IN A N
     EW FILE -"                                 :rem 131
1Ø4Ø :                                          :rem 255
1Ø5Ø REM"-INITIALIZE & GET FILENAMES-"          :rem 212
1Ø6Ø :                                            :rem 1
1Ø7Ø QT$=CHR$(34): REM"-QUOTE CHARACTER-"        :rem 94
1Ø8Ø CR$=CHR$(13): REM"-CARRIAGE RETURN-"        :rem 79
1Ø9Ø INPUT "OLD FILENAME{2 SPACES}";OF$         :rem 178
11ØØ INPUT "NEW FILENAME{2 SPACES}";NF$         :rem 18Ø
111Ø INPUT "FIELDS/RECORD ";FR                   :rem 15
112Ø :                                          :rem 254
113Ø REM"-OPEN DATA FILES-"                       :rem 6
114Ø :                                            :rem Ø
115Ø OPEN 8,8,8,"Ø:"+OF$+",S,R"                  :rem 35
116Ø OPEN 9,8,9,"Ø:"+NF$+",S,W"                  :rem 42
117Ø :                                            :rem 3
118Ø REM"-CREATE NEW FIRST LINE-"               :rem 154
119Ø :                                            :rem 5
12ØØ INPUT# 8,DE$: DE$=LEFT$(DE$,1)              :rem 85
121Ø PRINT# 9,QT$;DE$;"←@Ø:";NF$;CR$;            :rem 52
122Ø :                                          :rem 255
123Ø REM"-CONVERSION ROUTINE-"                  :rem 116
124Ø :                                            :rem 1
125Ø : RC=Ø: RO$=""                             :rem 132
126Ø : INPUT# 8,RI$: TT=ST: RO$=RO$+RI$         :rem 138
127Ø RC=RC+1: IF RC<FR THEN 126Ø                :rem 183
128Ø PRINT# 9,QT$;RO$;CR$;                      :rem 216
129Ø IF TT=Ø THEN 125Ø                          :rem 1Ø3
13ØØ CLOSE 8: CLOSE 9: END                      :rem 1Ø8
```

Let's take a look at the results produced by this sample program.

Assume that the original file contained records which we have determined should be grouped three per set. Also, we have already performed the preliminary conversion steps during which we selected ! as a delimiter and assigned a temporary filename of NAMES1. A partial listing of the file might appear as:

```
1000 "!←@0:NAMES1
1010 "MARY!
1020 "712 OAK ST.!
1030 "133-1478!
1040 "JOE!
1050 "884 ELM AVE.!
1060 "132-0808!
```

Running the sample program specifying NAMES2 as the new filename and requesting three fields per record would produce this file:

```
1000 "!←@0:NAMES2
1010 "MARY!712 OAK ST.!133-1478!
1020 "JOE!884 ELM AVE!132-0808!
```

After combining records as shown in this example, there is one additional step. Combining the short records may have produced new records that are too long, so you must again check for long records as you did in step 11 of the preliminary conversion process.

Simply use the DFH Editor to load the file and then execute the ;QT command. If no long-line errors are displayed, the file is ready to be used by any of the DFH processing programs.

If only a few long records are found, you should seriously consider whether they can be corrected on an individual basis. The procedures for this are also discussed in step 11 of the preliminary conversion process.

On the other hand, if a high percentage of the records are too long, you may need to consider some alternate method of grouping the data sets. The Genealogy File example in the applications section discusses this situation in some detail.

Splitting Long Records

If you complete the preliminary file conversion and find that most of the records are too long, you must decide on some method to split them. Exactly what you decide to do will, of course, depend on the nature of the data. Here is how you might handle a typical example.

We have casually been using name and address type data for some previous examples. Now let's use that same type of data for a more complex (perhaps more realistic) example. Assume that the existing file contains long records with six fields as follows:

Field 1: Name (Last, First, Middle)
Field 2: Address
Field 3: City, State and Zip Code
Field 4: Telephone Number(s)
Field 5: Birthdate
Field 6: Employer

The basic plan we will use in this case is to create separate records for the existing fields. This is almost the reverse of what we did in the example on combining records, but we will also introduce a new concept, the phantom field.

There is nothing magical about a phantom field. Even its name is nothing special and you may wish to call it something else depending on how it is used. (It is called an operations code field in the Genealogy File example in the applications chapter because of the function it performed there.) The term *phantom* often seems correct because it almost never appears in a printed output.

A phantom field is created to help control some data processing step. In this example it will be used to help control the data sorting process. It is assumed that there will be a need to sort a name and address file from time to time. Let's see why the extra help may be needed.

If we simply separated the fields of the example file into individual records and then sorted the file, the result would be a terrible mess. A partial solution would be to have two fields in each record of the new file, with the first field always containing the person's name. Now each set of data would stay together during a sort, but not in any particular order within the set.

Let's carry this idea further and use three fields per record. Again put the name in the first field, but move the file data to field 3. The second field will be a phantom field containing a sequence number. This number would indicate what field the data came from in the original file and would keep the records of each set in proper order during sorting.

You may have reasoned at this point that the name (field 1 in the original file) would not need to have its own separate record in the new file. After all, it's going to be included in all the other records. The problem is that if the name exists only in field 1, we must print field 1 in order to see it. Thus, we would have the name printed five times, once for each record.

That would look messy, so let's just treat the first field as another phantom field. Both the name (in field 1) and the sequence number (in field 2) will be used only for sorting control, and only the data (in field 3) will be printed. The first two fields could be combined, but file maintenance editing and new data entry using the DFH programs will be easier if they are left separated.

Now we can take a look at a BASIC program which performs the conversion we have just described:

Converting Files 2

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:EX.F.CONV-2",8              :rem 25Ø
1Ø1Ø :                                        :rem 252
1Ø2Ø REM"- AN EXAMPLE PROGRAM TO SPLIT LONG MULTI-
     FIELD RECORDS -"                         :rem 133
1Ø3Ø REM"- INTO SHORTER RECORDS IN A NEW FILE -"
                                              :rem 155
1Ø4Ø :                                        :rem 255
1Ø5Ø REM"-GET FILENAMES & OPEN FILES-"        :rem 135
1Ø6Ø :                                          :rem 1
1Ø7Ø INPUT "OLD FILENAME{2 SPACES}";OF$       :rem 176
1Ø8Ø INPUT "NEW FILENAME{2 SPACES}";NF$       :rem 187
1Ø9Ø OPEN 8,8,8,"Ø:"+OF$+",S,R"                :rem 38
11ØØ OPEN 9,8,9,"Ø:"+NF$+",S,W"                :rem 36
111Ø :                                        :rem 253
112Ø REM"-INITIALIZE & GET DELIMITER-"        :rem 221
113Ø :                                        :rem 255
114Ø DIM SE$(2Ø),DA$(2Ø)                      :rem 151
115Ø CR$=CHR$(13): QT$=CHR$(34)               :rem 1Ø8
116Ø : GET #8,DE$: IF DE$=QT$ THEN 116Ø       :rem 137
117Ø : GET #8,GT$: IF GT$<>CR$ THEN 117Ø      :rem 22Ø
118Ø FOR JJ=1 TO 2Ø                           :rem 184
119Ø SE$(JJ)=MID$(STR$(JJ),2)+DE$              :rem 48
12ØØ IF JJ<1Ø THEN SE$(JJ)="Ø"+SE$(JJ)        :rem 2Ø8
121Ø NEXT JJ                                  :rem 151
122Ø :                                        :rem 255
123Ø REM"-CREATE NEW FIRST LINE-"             :rem 15Ø
124Ø :                                          :rem 1
125Ø PRINT# 9,QT$;DE$;"←@Ø:";NF$;CR$;          :rem 56
```

```
126Ø :                                         :rem 3
127Ø REM"-PUT FIELDS IN ARRAY-"               :rem 18
128Ø :                                         :rem 5
129Ø : NR=1: GET #8,GT$: REM"-DISCARD QUOTE-"
                                              :rem 184
13ØØ : DA$(NR)=""                             :rem 25
131Ø : GET #8,GT$: TT=ST                     :rem 235
132Ø IF GT$=CR$ THEN 139Ø                      :rem 6
133Ø DA$(NR)=DA$(NR)+GT$                      :rem 34
134Ø IF GT$<>DE$ THEN 131Ø                    :rem 49
135Ø NR=NR+1: GOTO 13ØØ                      :rem 217
136Ø :                                         :rem 4
137Ø REM"-SAVE RECORDS TO DISK-"              :rem 92
138Ø :                                         :rem 6
139Ø : FOR JJ=1 TO NR-1                      :rem 145
14ØØ PRINT# 9,QT$;DA$(1);SE$(JJ);DA$(JJ);CR$;
                                              :rem 221
141Ø NEXT JJ: IF TT=Ø THEN 129Ø              :rem 114
142Ø CLOSE 8: CLOSE 9: END                   :rem 111
```

Since this is only a sample program to illustrate a process, it does not contain any error checks or self-protection features. Whatever problem you are actually trying to solve will undoubtedly be a little different from the one we have described. This program is general enough that it can be easily modified to solve the exact problem you are facing.

Because long records are hard to illustrate on a printed page without getting very difficult to read, we'll use short records. You can imagine them to be long and irregular.

Assume that we have already performed the preliminary conversion steps during which we found the delimiter to be * and assigned a temporary filename of TEST1. A partial listing of the file might appear as:

```
1000 "*←@0:TEST1
1010 "NA1*AD1*CT1*PH1*BD1*EM1*
1020 "NA2*AD2*CT2*PH2*BD2*EM2*
```

Running the sample program specifying TEST2 as the new filename would produce:

```
1000 "*←@0:TEST2
1010 "NA1*01*NA1*
1020 "NA1*02*AD1*
1030 "NA1*03*CT1*
1040 "NA1*04*PH1*
1050 "NA1*05*BD1*
```

```
1060 "NA1*06*EM1*
1070 "NA2*01*NA2*
1080 "NA2*02*AD2*
1090 "NA2*03*CT2*
1100 "NA2*04*PH2*
1110 "NA2*05*BD2*
1120 "NA2*06*EM2*
```

Whenever a record-splitting operation, as shown in this example, is completed, another check for any remaining long records is a good idea. Although you might feel it unlikely that any long records could remain, the check is easy to do and might avoid the loss of data during later processing.

If just a few long records are found, you may be able to correct them on an individual basis using the DFH Editor. The ;FC command is suggested due to its ability to handle long lines.

On the other hand, if a high percentage of the records are still too long, you will probably need to devise a new method of splitting the records.

There are no fixed rules about file conversion. Each time you want to convert an existing file to a structure that can be handled by the DFH programs, the details of the problem will be different. It is up to you to arrive at a suitable solution. All we can do is provide a powerful set of tools and instructions for their use.

You may also encounter some files that employ a structure that is either impractical or impossible to convert to the DFH format. One example of a very difficult structure is a sequential file that contains only one (very long) record. There is almost no means of even examining this type of file except with a BASIC program which uses the GET# command to access the data. Even the DFH Editor is of no use when the individual record length exceeds 250 data characters.

Chapter 8

The ML Subroutines

The Subroutine Package

In order to get full use of the explanations in this chapter, you should be an experienced BASIC programmer. It is not necessary to be able to understand or even read this chapter to get full use of the DFH programs. The purpose of this chapter is to illustrate how the machine language subroutines can be used in your own programs. If you're not interested in understanding how to use these programs for other applications, you should skip this chapter.

The machine language subroutines described here are used by each of the BASIC programs in the DFH family. The subroutine file, DFH SUBS$79, is automatically installed and used whenever it is needed without any special action from the user.

However, there is another way the subroutines can be used. Each subroutine performs useful functions that can easily be used in your own BASIC programs.

There are four machine language subroutines included in the DFH SUBS$79 program file. When loaded they occupy memory from $7900 (decimal 30976) to just short of $8000 (decimal 32768).

A brief summary of these functions is provided in the following paragraphs for quick reference. The summaries will be followed by detailed explanations and examples of how each subroutine can be used.

Sort. This is a routine to sort string data contained in single-dimension string arrays. Sorting can be performed on complete records (strings) or on individual fields within the records. Sorting can be in ascending or descending alphanumeric order. The number of non-null items sorted is reported to your BASIC program.

Activate by:	**SYS 30976 ($7900)**
Uses Variables:	**FA$ = Array name**
	FD$ = Delimiter
	FO$ = Sorting order
	FS% = Sort field number
	FT% = Number of strings sorted

Partition. Use this routine to separate the fields of a multifield string into individual strings. The multifield string is placed into the #0 element of a designated array by your BASIC program. The routine then places the contents of each field in the #1 through #n elements of the array and reports the total number of fields partitioned.

Activate by: **SYS 30979 ($7903)**

Uses Variables: **FA$ = Array name**
FD$ = Delimiter
FT% = Number of strings sorted

Convert. This is a routine to convert strings into equivalent strings in *WordPro* character code. Your BASIC program places the characters in a designated string variable and Convert leaves the results in the same string variable. The conversion can be to either uppercase or lowercase *WordPro* characters under control of the BASIC program.

Activate by: **SYS 30982 ($7906)**

Uses Variables: **WS$ = WordPro String**
WC$ = WordPro Case

Spool. This is a routine to read and disregard any specified number of records from a sequential data file on disk. It is used with files that have been opened by a BASIC program. When it is used to spool to a particular record, it will report the total number of records and characters in the file to that point. When directed to spool through an entire file, it will report the total number of records and characters in the file.

Activate by: **SYS 30985 ($7909)**

Uses Addresses: **30993-30994 = Target record number ($7911-$7912)**
30991-30992 = number of records spooled ($790F-$7910)
30995-30996 = number of characters spooled ($7913-$7914)

Multifield Records

Throughout this book we talk a lot about multifield records in sequential data files. The Sort and Partition subroutines depend heavily on this organization, so perhaps a short description oriented to those functions might help. Skip ahead if you are comfortable with multifield records.

Multifield records are an efficient method of storing and handling items of data that are related to each other in some way. This is probably the most common of all data-handling situations. The data for ordinary items like checkbooks, expense records, name and address lists, and price lists are all of this type. Each record (or line) in a multifield file will contain all the related data for one entry with each item of data in its own field.

A special character, called a delimiter, is used to mark the transition points between adjacent data fields. This allows the data to be fully compacted for storage because the only nondata characters in the file are the delimiters and a leading quote character.

A typical file of part numbers and prices, when loaded and listed using the DFH Editor, might appear as follows:

```
1000 "!←@0:PRICELIST
1010 "22007!9.95!
1020 "22493!24.50!
1030 "31447!4.45!
1040 "40987!34.00!
```

In the first record, the first nonquote character is the delimiter character used for the entire data file. In this example it is the exclamation mark (!). The remainder of the first record contains file identification information which is set up in the form of an DFH Editor SAVE command.

The leading quotation mark in each record allows the data to contain characters which would normally cause problems during an input operation. Typical characters in this class are the comma, the colon, and most shifted characters.

The leading quote also allows the first field to contain leading spaces or numeric characters. Data starting with numeric characters would normally cause trouble during onscreen editing since the numeric characters would be interpreted as part of the line number. The quote is discarded during INPUT operations, so it will not interfere with any type of data processing.

A delimiter is used at the end of each data record. This allows the last field in the record to contain trailing spaces.

When you are setting up your own file structures, you should seriously consider this method. Even if you have no intentions of using the DFH processing programs, you may still find uses for the DFH Editor's capabilities.

Both the Sort and the Partition subroutines presume a multifield structure even though the Sort routine can be directed to operate on the entire record without regard to delimiters. Obviously, Partition can do nothing without multiple fields.

These routines don't care how the data file is structured with regard to the file identification record or leading quotes. They operate on data records that have already been installed in a string data array under control of a BASIC program.

The Sort Routine

Sort is a handy machine language sorting utility. The routine uses the Heapsort algorithm, and operates on single-dimension string arrays that have been defined in a BASIC program.

A maximum string length of 255 characters is imposed by BASIC, and the Sort routine will work with strings up to that limit. If the INPUT# command is to be used, the limit is 80 characters per string. However, if the strings are to be saved in sequential data files, a more practical maximum of 74 characters is strongly recommended. This limit allows line numbers of up to four characters to be added for onscreen data editing under control of the DFH Editor. If the data files are to be used with other DFH programs, the 74-character limit is required.

The Sort routine presumes that the strings may be subdivided into separate data fields and that sorting may be done on the complete strings or on any single field within the strings. The boundaries of these fields are marked by delimiter characters:

S$(5)="JOE SMITH*DENVER*133-1784*"

In this example the string array element, S$(5) contains three fields separated by the delimiter *. The name is in field 1, city in field 2, and phone number in field 3.

The maximum number of records (strings) in the string array is defined by a DIMension statement in the BASIC program. The Sort routine will always operate on the entire array, including all null strings. The sorting time is determined more by the size of the array than by the amount of data in it. A sort time of eight seconds is typical for a 1000-record array.

To illustrate how the Sort routine handles null strings, consider the following program segment which creates a 31-element array with string data only in elements 11 through 20.

```
100 DIM A$(30): FOR J=11 TO 20
110 A$(J)=STR$(J): NEXT J
```

When the Sort routine is used on this array, the 10 non-null strings (11 through 20) will be sorted and moved into elements 0 through 9. The null strings will occupy the remaining elements of the array. This treatment of nulls is the same regardless of whether the sorting is in ascending or descending order.

The Sort routine uses five dedicated BASIC variables to define its sorting process.

FA$	Array Name (must be specified)
FT%	Total Non-null Strings sorted
FD$	Delimiter (default: FD$="*")
FS%	Sort Field (default: FS%=0)
FO$	Sort Order (default: FO$="A")

These variables are dedicated for the Sort routine in the same sense that ST, TI, and TI$ are dedicated for Commodore BASIC—you can use them, but only in specified ways.

The general programming procedure for using the Sort routine in a BASIC program is:

1. Set FA$ to define the array to be sorted. If the string array to be sorted is BX$( ), use FA$="BX".
2. Set FT%=0 to reserve a place in memory for the number of non-null strings sorted.
3. Set FD$ to define the delimiter. If the delimiter is ! then use FD$="!". If FD$ is not set, the Sort routine will use * as a default delimiter.
4. Set FS% to select the field to be sorted. If FS% is not set, the default will be field 0, which causes sorting of the complete strings including the delimiter characters.
5. Set FO$ to select the sort order. If FO$ is not defined, the default condition will be FO$="A", which produces a sort in ascending order. FO$="D" produces a sort in descending order.
6. SYS 30976 to execute the Sort routine. (Same as $7900.)

During execution of the Sort routine, a special flashing cursor will be displayed on the screen to let you know the program is busy sorting.

When control is returned to the BASIC program, the first string will be located in element 0 of the array and the last non-null string will be in element FT%−1. Actually the strings are not moved; only the array element pointers are changed. The result is the same and it's much faster.

An individual field in a string can be null simply by having two delimiter characters next to each other. A field is also considered null if the field number specified for sorting is larger than the number of fields actually contained in the string.

Only the strings containing data in the sort field will be sorted. The remaining strings in the array will be treated as nulls, and due to the nature of the Heapsort algorithm, they will be scattered through the remainder of the string array elements. (Not to worry. The scattered records can be recovered by sorting on some other field or on the entire record, a field 0 sort.)

Since FT% reports the number of non-null elements that it has sorted, the deliberate inclusion of null fields can be used to allow selective extraction of a part of the array contents.

There are three error messages that can be produced by the Sort routine.

UNDEF'D FUNCTION ERROR	No array name was assigned to FA$.
FILE NOT FOUND	The array assigned to FA$ was not a string array.
DIM'D ARRAY ERROR	The array assigned to FA$ had more than one dimension.

A BASIC program illustrating the use of the Sort subroutine is included in the programming examples at the end of this chapter.

The Partition Routine

The Partition subroutine is a fast machine language routine that creates separate strings from the individual fields of a multifield string. Like Sort, it is intended to be used as a subroutine called from a BASIC program. Due to its generalized nature, and because equivalent routines in BASIC are very slow, it is useful in a wide variety of situations.

The Partition routine uses three of the same dedicated variables used by the Sort routine:

FA$	Array Name (must be specified)
FT%	Total fields partitioned
FD$	Delimiter (default: FD$="*")

The general programming procedure for using the Partition routine in a BASIC program is:

1. Dimension a string array of up to 254 elements to hold the partitioned strings. For example, DIM DA$(20) for up to 20 fields in a string.
2. Set FA$ to identify the array to be used. If the array is DA$( ), then use FA$="DA".
3. Set FT%=0 to reserve a place for the number of fields partitioned.
4. Set FD$ to define the delimiter. If the delimiter is ! then use FD$="!". If FD$ is not set, the Partition routine will use * as the default delimiter.
5. Put the string to be partitioned in the 0 element of the partitioning array, DA$(0) for this example.
6. SYS 30979 to execute the Partition routine. ($7903)

When control is returned to the BASIC program, FT% will contain the number of separate strings created. The first of these strings will be located in element 1 of the defined array, the second string in element 2, etc. If any null fields were contained within the original string, the corresponding element in the array will be null.

Array elements not needed by the Partition routine will retain their previous contents. Thus, if you partitioned a six-field string and then partitioned a four-field string, the array elements 5 and 6 would still contain data from the first string. For this reason it is important to use the value in FT% to determine where the last valid field was placed.

The Partition routine can produce the same three error messages as Sort. However, please notice that the DIM'D ARRAY ERROR message now has two possible meanings.

UNDEF'D FUNCTION ERROR	No array name was assigned to FA$.
FILE NOT FOUND	The array assigned to FA$ was not a string array.
DIM'D ARRAY ERROR	The array assigned to FA$ had more than one dimension, or it was dimensioned with more than 254 elements.

A BASIC program illustrating the use of the Partition subroutine is included in the programming examples at the end of this chapter.

Subroutine Protection

DFH SUBS$79 can be loaded like a BASIC program, either from the keyboard (immediate mode) or from within a BASIC program. If you are using a Commodore 64, don't forget the ,1 at the end of the LOAD command:

```
LOAD "DFH SUBS$79",8,1
```

Because of its location at $7900, the top of memory must be set at $7900 (or lower) to protect the program code from strings generated by BASIC programs. This is done from the 64 keyboard by

```
POKE 55,0: POKE 56,121: CLR
```

or from the PET keyboard by

```
POKE 52,0: POKE 53,121: CLR
```

A much simpler method, from an operator's point of view, is to load DFH SUBS$79 from within the same BASIC program that will use it. Like DFH BOOT, the BASIC program should check to see if DFH SUBS$79 is loaded and protected. If that has not been done, the program should lower the top of memory and load DFH SUBS$79.

After this LOAD the computer will automatically begin execution at the lowest-numbered line in the BASIC program. This time the tests will pass and the BASIC program continues execution. The program lines to perform this Test and LOAD operation should be at or very near the start of the program. An example for the Commodore 64:

```
100 IF PEEK (56)>120 THEN POKE 55,0: POKE 56,121: CLR
110 IF PEEK (30977)=19 AND PEEK (30980)=28 THEN 130
120 LOAD "DFH SUBS$79",8,1
130 REM"-- PROGRAM CONTINUES --"
```

For PET computers line 100 should be

```
100 IF PEEK (53)>120 THEN POKE 52,0: POKE 53,121: CLR
```

While this is obviously not foolproof, it will get the job done in most cases, and the extra LOAD is performed only when necessary.

More complete examples of this technique can be found in the DFH programs.

Subroutine Examples

Below is a series of BASIC program examples that illustrate the use of the four machine language subroutines of DFH SUBS$79. We have included these examples as an illustration of how you might use these subroutines in your own BASIC programs.

The first program, EX.CREATE, is used to create a sequential file named TESTFILE and store it on disk. The other four example programs will use this file for demonstration purposes.

TESTFILE will contain a file identification line followed by ten data records. Each data record contains three fields: an ID number, a name, and a birthdate. The * is used as a delimiter character.

The example programs contain REM lines to show you what each section of code is doing. Therefore, the written explanation for each program is quite brief.

EX.CREATE

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1000 REM SAVE "@0:EX.CREATE",8                :rem 165
1010 :                                        :rem 252
1020 REM"-- AN EXAMPLE PROGRAM TO CREATE A TEST DA
     TA --"                                    :rem 68
1030 REM"-- FILE FOR USE BY OTHER EXAMPLE PROGRAMS
      --"                                      :rem 40
1040 :                                        :rem 255
1050 REM"-- INITIALIZE --"                    :rem 148
1060 CR$=CHR$(13): REM"-- CARRIAGE RETURN --"
                                              :rem 167
1070 QT$=CHR$(34): REM"-- QUOTE CHARACTER --"
                                              :rem 184
1080 FO$="TESTFILE": REM"-- OUTPUT FILENAME --"
                                              :rem 171
1090 :                                          :rem 4
1100 REM"-- OPEN THE DISK STORAGE FILE --" :rem 17
1110 OPEN 9,8,9,"@0:"+FO$+",S,W"              :rem 102
1120 :                                        :rem 254
1130 REM"-- CREATE FIRST LINE WITH DELIMITER = * -
     -"                                        :rem 71
1140 PRINT# 9,QT$;"*←@0:";FO$;CR$;            :rem 121
1150 :                                          :rem 1
1160 REM"-- READ, PRINT, AND SAVE DATA RECORDS --"
                                              :rem 211
```

```
117Ø FOR JJ=1 TO 1Ø: READ DA$: PRINT DA$     :rem 37
118Ø PRINT# 9,QT$;DA$;CR$;                   :rem 187
119Ø NEXT JJ                                 :rem 158
12ØØ :                                       :rem 253
121Ø REM"-- CLOSE DISK FILE AND QUIT --"     :rem 119
122Ø CLOSE 9: END                            :rem 133
123Ø :                                         :rem Ø
124Ø REM"-- FILE DATA, ID#*NAME*BIRTHDAY       :rem 9
125Ø DATA "ØØ1*TOM*Ø5-26"                    :rem 245
126Ø DATA "1Ø1*LA VERDA*Ø9-22"                 :rem 6
127Ø DATA "ØØ2*JOHN B*Ø3-24"                 :rem 117
128Ø DATA "1Ø2*PAT*Ø4-28"                    :rem 24Ø
129Ø DATA "ØØ3*MYRON*Ø8-15"                  :rem 161
13ØØ DATA "1Ø3*DONNA*11-1Ø"                  :rem 1Ø6
131Ø DATA "ØØ4*HOWARD*Ø2-Ø8"                 :rem 199
132Ø DATA "1Ø4*DEMA*Ø3-21"                    :rem 23
133Ø DATA "ØØ5*JOHN D*Ø8-Ø4"                 :rem 122
134Ø DATA "1Ø5*PEGGY*Ø7-12"                  :rem 131
135Ø :                                         :rem 3
```

Each of the following example programs begins with a check to see which computer is being used. This information is used to set the top of memory at $7900 to protect the machine language subroutine file DFH SUBS$79. This is followed immediately by a check to see if the subroutines are already in place. If not, the subroutine file is loaded.

The following program demonstrates the use of the Sort subroutine. Data records are read from TESTFILE into the middle of an array, and the contents of the array are printed.

The first sorting pass demonstrates a sort on field 2 in ascending order and shows that the records are moved to the start of the array during sorting.

The next pass demonstrates a descending order sort on field 1. Notice that the only setup changes required are to FS% to control the sort field and to FO$ to control the sorting order.

The program pauses after each operation to allow examination of the results on the screen.

EX.SORT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:EX.SORT",8               :rem 57
1Ø1Ø :                                    :rem 252
1Ø2Ø REM"-- AN EXAMPLE PROGRAM USING THE 'SORT' --
     "                                     :rem 79
```

```
1Ø3Ø REM"-- SUBROUTINE CONTAINED IN 'DFH SUBS$79'
     {SPACE}--"                                 :rem 2Ø5
1Ø4Ø :                                          :rem 255
1Ø5Ø REM"-- TOP OF MEMORY = $79ØØ TO PROTECT SUBRO
     UTINES --"                                 :rem 91
1Ø6Ø IF PEEK (65534)=72 THEN 1Ø8Ø: REM"-- C64 COMP
     UTER --"                                   :rem 163
1Ø7Ø POKE 52,Ø: POKE 53,121: GOTO 111Ø: REM"-- PET
      --"                                       :rem 128
1Ø8Ø : POKE 55,Ø: POKE 56,121: REM"-- C64 --"
                                                 :rem 79
1Ø9Ø :                                            :rem 4
11ØØ REM"-- TEST/INSTALL M.L. SUBROUTINES --"
                                                :rem 124
111Ø : IF PEEK(3Ø977)=21 AND PEEK (3Ø98Ø)=3Ø THEN
     {SPACE}116Ø                                :rem 144
112Ø PRINT "{RVS} LOADING DFH SUBS$79 {OFF}"
                                                :rem 218
113Ø CLR : LOAD "DFH SUBS$79",8,1               :rem 168
114Ø :                                            :rem Ø
115Ø REM"-- INITIALIZE --"                      :rem 149
116Ø : DIM DA$(2Ø): REM"-- DATA ARRAY --" :rem 231
117Ø SS=3Ø976: REM"-- SORT ADDRESS $79ØØ --"
                                                 :rem 13
118Ø FT%=Ø: REM"-- RECORD COUNT --"              :rem 84
119Ø FA$="DA": REM"-- SORT ARRAY NAME --" :rem 122
12ØØ :                                          :rem 253
121Ø REM"-- OPEN FILE AND GET DELIMITER --":rem 68
122Ø OPEN 3,8,3,"Ø:TESTFILE,S,R"                 :rem 36
123Ø INPUT# 3,FD$: FD$=LEFT$(FD$,1)              :rem 86
124Ø :                                            :rem 1
125Ø REM"-- LOAD DATA TO ARRAY ELEMENTS #5 THRU #1
     4                                            :rem 4
126Ø JA=4                                       :rem 197
127Ø : JA=JA+1: INPUT# 3,DA$(JA): IF ST=Ø THEN 127
     Ø                                           :rem 9Ø
128Ø CLOSE 3                                    :rem 116
129Ø :                                            :rem 6
13ØØ REM"-- DISPLAY RAW DATA --"                :rem 186
131Ø PRINT "{DOWN}ARRAY#","RAW DATA FROM DISK
     {DOWN}"                                     :rem 45
132Ø FOR JJ=Ø TO 15: PRINT JJ,DA$(JJ): NEXT JJ
                                                :rem 217
133Ø GOSUB 148Ø                                  :rem 2Ø
134Ø :                                            :rem 2
135Ø REM"-- SORT ON FIELD #2 AND DISPLAY RESULTS -
     -"                                          :rem 94
136Ø PRINT "{DOWN}ARRAY#","ASCENDING SORT ON FIELD
      #2{DOWN}"                                 :rem 249
137Ø FO$="A": FS%=2: SYS SS                     :rem 14Ø
```

```
138Ø FOR JJ=Ø TO FT%-1: PRINT JJ,DA$(JJ) : NEXT JJ
                                          :rem 15Ø
139Ø GOSUB 148Ø                            :rem 26
140Ø :                                    :rem 255
141Ø REM"-- SORT ON FIELD #1 AND DISPLAY RESULTS -
     -"                                    :rem 9Ø
142Ø PRINT "{DOWN}ARRAY#","DESCENDING SORT ON FIEL
     D #1{DOWN}"                           :rem 61
143Ø FO$="D": FS%=1: SYS SS               :rem 139
144Ø FOR JJ=Ø TO FT%-1: PRINT JJ,DA$(JJ) : NEXT JJ
                                          :rem 147
145Ø END                                  :rem 161
146Ø :                                      :rem 5
147Ø REM"--SUBR-- WAIT FOR OPERATOR --"   :rem 198
148Ø : PRINT "{RVS} PRESS ANY KEY TO CONTINUE
     {OFF}{DOWN}"                         :rem 243
149Ø : GET KB$: IF KB$<>"" THEN 149Ø      :rem 2Ø6
15ØØ : GET KB$: IF KB$="" THEN 15ØØ       :rem 129
151Ø RETURN                               :rem 167
```

The next program shows how to use the Partition subroutine to separate multifield records into individual strings.

The original records are displayed, followed immediately by the partitioned results.

EX.PARTITION

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:EX.PARTITION",8         :rem 171
1Ø1Ø :                                    :rem 252
1Ø2Ø REM"-- AN EXAMPLE PROGRAM USING THE 'PARTITIO
     N' --"                               :rem 193
1Ø3Ø REM"-- SUBROUTINE CONTAINED IN 'DFH SUBS$79'
     {SPACE}--"                           :rem 2Ø5
1Ø4Ø :                                    :rem 255
1Ø5Ø REM"-- TOP OF MEMORY = $79ØØ TO PROTECT SUBRO
     UTINES --"                            :rem 91
1Ø6Ø IF PEEK (65534)=72 THEN 1Ø8Ø: REM"-- C64 COMP
     UTER --"                             :rem 163
1Ø7Ø POKE 52,Ø: POKE 53,121: GOTO 111Ø: REM"-- PET
      --"                                 :rem 128
1Ø8Ø : POKE 55,Ø: POKE 56,121: REM"-- C64 --"
                                           :rem 79
1Ø9Ø :                                      :rem 4
11ØØ REM"-- TEST/INSTALL M.L. SUBROUTINES --"
                                          :rem 124
111Ø : IF PEEK(3Ø977)=21 AND PEEK (3Ø98Ø)=3Ø THEN
     {SPACE}116Ø                          :rem 144
```

```
112Ø PRINT "{RVS} LOADING DFH SUBS$79 {OFF}"
                                          :rem 218
113Ø CLR : LOAD "DFH SUBS$79",8,1         :rem 168
114Ø :                                      :rem Ø
115Ø REM"-- INITIALIZE --"                :rem 149
116Ø : DIM DA$(2Ø): REM"-- PARTITIONING SPACE FOR
     {SPACE}2Ø FIELDS --"                  :rem 82
117Ø SP=3Ø979: REM"-- PARTITION ADDRESS = $79Ø3 --
     "                                    :rem 191
118Ø FT%=Ø: REM"-- PARTITIONED FIELD COUNT --"
                                           :rem 6Ø
119Ø FA$="DA": REM"-- PARTITION ARRAY NAME --"
                                          :rem 236
12ØØ :                                    :rem 253
121Ø REM"-- OPEN DATA FILE & GET DELIMITER --"
                                          :rem 177
122Ø OPEN 9,8,9,"Ø:TESTFILE,S,R"           :rem 48
123Ø INPUT# 9,FD$ : FD$=LEFT$(FD$,1)       :rem 92
124Ø :                                      :rem 1
125Ø REM"-- INPUT, PARTITION & DISPLAY DATA RECORD
     S --"                                :rem 13Ø
126Ø : INPUT# 9,DL$: TT=ST                :rem 149
127Ø PRINT DL$                             :rem 11
128Ø DA$(Ø)=DL$: SYS SP: REM"-- PARTITION --"
                                          :rem 146
129Ø PRINT "{2 SPACES}";: FOR JJ=1 TO FT%: PRINT D
     A$(JJ),: NEXT JJ: PRINT              :rem 178
13ØØ IF TT=Ø THEN 126Ø: REM"-- NEXT RECORD --"
                                          :rem 116
131Ø CLOSE 9: END                         :rem 133
```

The next is for *WordPro* users only. It demonstrates the use of the Convert subroutine by taking the contents of the previously prepared sequential file, TESTFILE, and creates a *WordPro* file containing those data records. The *WordPro* file is named WPROFILE.

Since the only way to check the results of this program is under control of *WordPro,* if you do not use *WordPro,* you should skip this example.

Pay special attention to the code that adds the left-arrow character (*WordPro*'s carriage return indicator) and fills the remainder of the line with spaces. Each line in a finished *WordPro* file should have an exact multiple of 40 characters. This system will work for both 40- and 80-column computers.

EX.CONVERT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:EX.CONVERT",8                  :rem 18
1Ø1Ø :                                           :rem 252
1Ø2Ø REM"-- AN EXAMPLE PROGRAM USING THE 'CONVERT'
      --"                                        :rem 4Ø
1Ø3Ø REM"-- SUBROUTINE CONTAINED IN 'DFH SUBS$79'
     {SPACE}--"                                  :rem 2Ø5
1Ø4Ø :                                           :rem 255
1Ø5Ø REM"-- TOP OF MEMORY = $79ØØ TO PROTECT SUBRO
     UTINES --"                                  :rem 91
1Ø6Ø IF PEEK (65534)=72 THEN 1Ø8Ø: REM"-- C64 COMP
     UTER --"                                    :rem 163
1Ø7Ø POKE 52,Ø: POKE 53,121: GOTO 111Ø: REM"-- PET
      --"                                        :rem 128
1Ø8Ø : POKE 55,Ø: POKE 56,121: REM"-- C64 --"
                                                 :rem 79
1Ø9Ø :                                           :rem 4
11ØØ REM"-- TEST/INSTALL M.L. SUBROUTINES --"
                                                 :rem 124
111Ø : IF PEEK(3Ø977)=21 AND PEEK (3Ø98Ø)=3Ø THEN
     {SPACE}116Ø                                 :rem 144
112Ø PRINT "{RVS} LOADING DFH SUBS$79 {OFF}"
                                                 :rem 218
113Ø CLR : LOAD "DFH SUBS$79",8,1                :rem 168
114Ø :                                           :rem Ø
115Ø REM"-- INITIALIZE --"                       :rem 149
116Ø SC=3Ø982: REM"-- CONVERT ADDRESS $79Ø6 --"
                                                 :rem 216
117Ø SP$="{4Ø SPACES}": REM"-- 4Ø SPACES --"
                                                 :rem 74
118Ø WC$="": REM"-- UPPER CASE --"               :rem 199
119Ø :                                           :rem 5
12ØØ REM"-- OPEN DATA FILE AND WORDPRO FILE --"
                                                 :rem 43
121Ø OPEN 8,8,8,"Ø:TESTFILE,S,R"                 :rem 45
122Ø INPUT# 8,WS$: REM"-- DISCARD FIRST LINE --"
                                                 :rem 1Ø6
123Ø OPEN 9,8,9,"@Ø:WPROFILE,P,W"                :rem 123
124Ø PRINT# 9,CHR$(Ø);CHR$(64);: REM"-- DUMMY 'LOA
     D ADDRESS'=$4ØØØ --"                        :rem 2Ø3
125Ø :                                           :rem 2
126Ø REM"-- INPUT, PRINT AND CONVERT DATA --"
                                                 :rem 252
127Ø : INPUT# 8,WS$: TT=ST                       :rem 175
128Ø PRINT WS$:                                  :rem 96
129Ø SYS SC: REM"-- CONVERT WS$ TO WORDPRO CHARACT
     ERS --"                                     :rem 22
13ØØ :                                           :rem 254
```

```
1310 REM"-- PAD WITH LEFT ARROW & SPACES TO MULTIP
     LE OF 40 CHARACTERS --"                 :rem 53
1320 LE=LEN(WS$)                            :rem 146
1330 AC=39-LE-40*(LE>39)-40*(LE>79)-40*(LE>119)
                                            :rem 186
1340 REM"-- PREVIOUS LINE NOT VALID FOR STRING LEN
     GTHS > 159 CHARACTERS --"               :rem 58
1350 WS$=WS$+CHR$(31): REM"-- ADD LEFT ARROW --"
                                             :rem 24
1360 WS$=WS$+LEFT$(SP$,AC): REM"-- ADD SPACES --"
                                            :rem 131
1370 PRINT# 9,WS$;: REM"-- SAVE LINE TO WORDPRO FI
     LE --"                                  :rem 70
1380 LC=LC+1: IF TT=0 THEN 1270: REM"-- GET NEXT R
     ECORD --"                               :rem 78
1390 :                                        :rem 7
1400 CLOSE 8: CLOSE 9                        :rem 92
1410 PRINT "{DOWN}";LC;"DATA RECORDS CONVERTED TO
     {SPACE}WORDPRO"                        :rem 151
1420 PRINT "FORMAT.{2 SPACES}WORDPRO FILE 'WPROFIL
     E' SAVED"                                :rem 5
1430 PRINT "ON DISK.                        :rem 109
```

The final example program shows how to use the Spool subroutine to read and disregard records in a file to gain access to a particular record in that file.

The program allows you to choose any one of the ten data records in TESTFILE for spooling and display. It also displays the total number of characters in the file prior to the selected record.

Pay special attention to the method used to compute the POKE values for the target record number, and the method of recovering the total character count.

EX.SPOOL

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1000 REM SAVE "@0:EX.SPOOL",8               :rem 126
1010 :                                      :rem 252
1020 REM"-- AN EXAMPLE PROGRAM USING THE 'SPOOL' -
     -"                                     :rem 148
1030 REM"-- SUBROUTINE CONTAINED IN 'DFH SUBS$79'
     {SPACE}--"                             :rem 205
1040 :                                      :rem 255
1050 REM"-- TOP OF MEMORY = $7900 TO PROTECT SUBRO
     UTINES --"                              :rem 91
```

```
1060 IF PEEK (65534)=72 THEN 1080: REM"-- C64 COMP
     UTER --"                                :rem 163
1070 POKE 52,0: POKE 53,121: GOTO 1110: REM"-- PET
      --"                                    :rem 128
1080 : POKE 55,0: POKE 56,121: REM"-- C64 --"
                                              :rem 79
1090 :                                         :rem 4
1100 REM"-- TEST/INSTALL M.L. SUBROUTINES --"
                                             :rem 124
1110 : IF PEEK(30977)=21 AND PEEK (30980)=30 THEN
     {SPACE}1160                             :rem 144
1120 PRINT "{RVS} LOADING DFH SUBS$79 {OFF}"
                                             :rem 218
1130 CLR : LOAD "DFH SUBS$79",8,1            :rem 168
1140 :                                         :rem 0
1150 REM"-- INITIALIZE AND POKE TARGET # TO $7911-
     12{2 SPACES}(L,H) --"                   :rem 187
1160 : SS=30985: REM"-- SPOOL ADDRESS =$7909
                                              :rem 85
1170 : INPUT "TARGET RECORD # ";RN            :rem 91
1180 IF RN<1 OR RN>10 THEN 1170               :rem 62
1190 RP%=RN/256: POKE SS+9,RP%               :rem 161
1200 POKE SS+8,RN-RP%/256                    :rem 135
1210 :                                       :rem 254
1220 REM"-- OPEN FILE, SPOOL, AND INPUT RECORD --"
                                             :rem 250
1230 OPEN 5,8,6,"TESTFILE,S,R"               :rem 192
1240 SYS SS: REM"-- SPOOL --"                 :rem 15
1250 INPUT# 5,DA$: CLOSE 5                   :rem 106
1260 :                                         :rem 3
1270 REM"-- PRINT RECORD AND CHARACTER COUNT --"
                                             :rem 219
1280 PRINT "{DOWN}";DA$;"{DOWN}"              :rem 33
1290 TC=PEEK (SS+10)+PEEK (SS+11)*256        :rem 227
1300 PRINT "THE FILE CONTAINS";TC;"CHARACTERS"
                                              :rem 38
1310 PRINT "PRIOR TO THIS RECORD.{DOWN}"     :rem 251
1320 INPUT "ANOTHER RECORD{4 SPACES}Y{3 LEFT}";KB$
                                             :rem 134
1330 IF LEFT$(KB$,1)="Y" THEN 1170           :rem 214
```

Chapter 9

Entering the Programs

A Beginner's Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential,* but without a program, it isn't going anywhere. Most of the programs published in this book are written in a computer language called BASIC. BASIC is easy to learn and is built into all Commodore 64s and PETs.

BASIC Programs

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as O for the numeral 0, a lowercase l for the numeral 1, or an uppercase B for the numeral 8. Also, you must enter all punctuation such as colons and commas just as they appear in the book. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

Braces and Special Characters

The exception to this typing rule is when you see the braces, such as {DOWN}. Anything within a set of braces is a special character or characters that cannot easily be listed on a printer. When you come across such a special statement, refer to "How to Type In Programs."

About DATA Statements

Some of the DFH programs contain numerous DATA statements. These lines provide information needed by the program. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard and STOP key may seem dead, and the screen may go blank. Don't panic—no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it.* If your computer crashes, you can load the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get to Know Your Machine

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program, so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? You can always retype the line, but you at least need to know how to backspace. Do you know how to enter reverse video, lowercase, and control characters? It's all explained in your computer's manuals.

In order to insure accurate entry of each program line, we have included a checksum program. Please read the article called "The Automatic Proofreader" before typing in any of the programs in this book.

A Quick Review

1. Type in the program a line at a time, in order. Press RETURN at the end of each line. Use backspace or the back arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

How to Type In Programs

To make it easy to know exactly what to type when entering one of these programs into your computer, we have established the following listing conventions.

Generally, Commodore 64 and PET program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor-down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the SHIFT key. This would appear on your screen as a heart symbol. If you find an underlined key enclosed in braces (for example, {10 N}), you should type the key as many times as indicated (in our example, you would enter ten shifted N's).

Here is a list of some of the characters you will see in braces and the proper keys to press:

When you see	**Press**
{CLR}	SHIFT and CLR/HOME
{HOME}	CLR/HOME
{UP}	SHIFT/↑CRSR↓
{DOWN}	↑CRSR↓
{LEFT}	SHIFT/←CRSR→
{RIGHT}	←CRSR→
{RVS}	CTRL/9 (64) or RVS (PET)
{OFF}	CTRL/0 (64) or SHIFT/RVS (PET)
{SPACE}	Space bar

About the *quote mode:* You know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {CLR}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT/2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

In order to insure accurate entry of each program line, we have included a checksum program. Please read the article called "The Automatic Proofreader" before typing in any of the programs in this book.

Use the Correct Filenames

You must save each of the DFH programs using the exact filename listed below. Save each of the BASIC programs as you usually would:

SAVE *"filename"*,8

When creating the machine language programs, follow the directions in the section "Machine Language Program Generator" later in this chapter. Be sure to enter the proper filename when prompted by the generator program.

BASIC Program Filenames
DFH BOOT
DFH SORT
DFH PRINT
DFH MERGE
DFH SWAP
DFH SPLIT

Machine Language Program Filenames
DFH SUBS$79
DFH ED.C64$90
DFH ED.PET$70

The Automatic Proofreader

Charles Brannon

The listings for each of the BASIC programs contain a trailing REM and a checksum number at the end of each program line. These items are not a part of the programs. They simply show the checksum number that will be produced by the Proofreader routine as each line is entered.

For the Commodore 64 this is the same program that has been published several times in *COMPUTE!* and in *COMPUTE!'s Gazette.* If you are using a Commodore 64 and are already familiar with operating the Proofreader program, you can proceed immediately to type in the BASIC programs.

The Proofreader will work on all Commodore 64s and on all PETs containing BASIC 3.0 and 4.0. It will not work on the very earliest PETs.

The Proofreader for the PET operates exactly the same as the one for the 64, but it is located just below the screen memory, at $7F00, rather than in the tape buffer.

It is best to begin using the Proofreader with no other programs loaded in memory. Therefore, before running the Proofreader, turn your computer off then on again, and load and run the Proofreader. The following section explains how to install and use the Proofreader programs.

Preparing the Proofreader

1. Using the listing below, type in the Proofreader for your computer. Be very careful when entering the DATA statements—don't type an l instead of a 1, an O instead of a 0, extra commas, etc.
2. Save the Proofreader on disk at least twice *before running it for the first time.* This is very important because the Proofreader erases part of itself when you first type RUN.
3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place—you'll need it again and again, every time you enter a program from this book.

4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. On the 64 if you press RUN/STOP–RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN. The PET Proofreader can only be disabled by cycling power.

Using the Proofreader

All listings in this book have a *checksum number* appended to the end of each line, for example, *:rem 123. Don't enter this statement when typing in a program.* It is just for your information. The rem makes the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum number in the printed listing.* If it doesn't, it means you typed the line differently than the way it is listed. Immediately recheck your typing. Remember, don't type the rem statement with the checksum number; it is published only so you can check it against the number which appears on your screen.

The Proofreader is not picky with spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces, since the Proofreader will catch practically everything else that can go wrong.

There's another thing to watch out for: If you enter a line using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way.

Commodore 64 Proofreader

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO1018:READ
    A:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MADE AN ERRO
    R":PRINT"IN DATA STATEMENTS.":END
```

```
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER ACTIVATE
    D.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

PET Proofreader

```
1080 PRINT "{CLR}PLEASE WAIT..."
1090 FOR I=32512 TO 32686
1100 READ A: C=C+A: POKE I,A: NEXT I
1110 IF C=23728 THEN 1140
1120 PRINT "{DOWN}THERE IS AN ERROR"
1130 PRINT "IN THE DATA STATEMENTS.": END
1140 : PRINT "{CLR}{2 DOWN}PROOFREADER ACTIVATED."
1150 POKE 52,0: POKE 53,127
1160 SYS 32512: NEW
1170 :
1180 DATA 169,076,133,112,169,013
1190 DATA 133,113,169,127,133,114
1200 DATA 096,230,119,208,002,230
1210 DATA 120,142,176,127,140,177
1220 DATA 127,104,168,104,170,072
1230 DATA 152,072,224,180,240,013
1240 DATA 224,195,240,009,174,176
1250 DATA 127,172,177,127,076,118
1260 DATA 000,165,119,141,066,127
1270 DATA 141,082,127,165,120,141
```

```
128Ø DATA Ø67,127,141,Ø83,127,173
129Ø DATA ØØ2,ØØ2,2Ø1,ØØØ,24Ø,224
13ØØ DATA 2Ø1,Ø32,24Ø,22Ø,169,ØØØ
131Ø DATA 141,175,127,173,ØØ2,ØØ2
132Ø DATA 2Ø1,ØØØ,24Ø,Ø22,2Ø1,Ø32
133Ø DATA 24Ø,ØØ7,Ø24,1Ø9,175,127
134Ø DATA 141,175,127,238,Ø82,127
135Ø DATA 2Ø8,233,238,Ø83,127,Ø76
136Ø DATA Ø81,127,165,216,141,178
137Ø DATA 127,2Ø6,178,127,169,Ø19
138Ø DATA Ø32,21Ø,255,169,Ø18,Ø32
139Ø DATA 21Ø,255,169,Ø58,Ø32,21Ø
14ØØ DATA 255,174,175,127,169,ØØØ
141Ø DATA 172,252,255,192,Ø22,24Ø
142Ø DATA ØØ6,Ø32,217,22Ø,Ø76,154
143Ø DATA 127,Ø32,131,2Ø7,169,Ø32
144Ø DATA Ø32,21Ø,255,Ø32,21Ø,255
145Ø DATA 173,178,127,133,216,169
146Ø DATA Ø13,Ø32,21Ø,255,Ø76,Ø4Ø
147Ø DATA 127
```

Machine Language Program Generator

The three machine language program files, DFH SUBS$79, DFH ED.C64$90, and DFH ED.PET$70, are each quite long when entered by hand. Without some help, they would also be terribly hard to debug to find typing errors.

The method used to enter the DFH machine language programs will make errors very unlikely. In general, you would need to make a combination of four exactly compensating errors to get by the error-detecting routine.

There are a number of convenience features you should enjoy. You will be entering BASIC programs, so you can save the partially completed program whenever you wish and complete the entry process in as many sessions as you like. There will be no temporary addresses to remember. You can make test runs as often as you like to check your typing accuracy, or make none at all until you have entered all the data. It's your choice, and the program will assist you in either method.

Your typing errors can be corrected one at a time as they are found or simply noted and corrected in a single editing session. The program does all error checks each time it is run. It will not create the machine language file until all errors are corrected. When the machine language file is finally created, it is guaranteed to be correct and no debugging will be needed.

General Procedure

The first step in the procedure is to type in the BASIC program, ML PROG GEN. This is a reusable program which is set up so that DATA statements can be added to it. Three different sets of DATA statements will be transformed into three machine language program files on disk.

When ML PROG GEN has been typed in and saved, we will temporarily attach a very short set of DATA statements that are designed to test various features of the program. When testing is completed, we will strip away the DATA statements and save the program. This version should be left intact for future use.

In three separate operations (one for each of the machine language programs) we will modify copies of ML PROG GEN by attaching DATA statements. Each copy will be given a new name and saved.

The DATA statements contain the load point address of the machine language program, all the program bytes, and checksums for both rows and columns. All the data is in hex-byte form, eight-bytes per line, so that it is visually compatible with an eight-column machine language monitor display. The hex form and the absence of commas also reduce your typing effort.

As a double-check we have included Proofreader rems for each of the DATA statements. This means that you can use the Proofreader while entering the data and check each line as it is entered. Then, when the ML PROG GEN is run, it will do a second check of the data, as explained in the next paragraph. This will assure that the final machine language program will be correct.

When these BASIC programs are run, they ask for a filename to use in saving the machine language program. The data in each row and column is cross-checked for accuracy, and the machine language file is created and saved. If an error is detected, the file creation is aborted and the error is displayed. (Be sure to respond to the prompt with the correct filename, as shown in the first line of the DATA listing.)

If you wish, you may make a note of the DATA line numbers containing the errors and continue with tests for any additional errors. The errors are corrected by editing the same as with any BASIC program. Remember to resave the program after making corrections.

When the program finally runs to completion with no errors, it will have saved a completely correct machine language program file on disk—ready to load and execute.

Programmers will find the SAVE technique in the ML PROG GEN interesting. If the SAVE is creating a new file and encounters an error, it simply scratches the file. However, if it was doing a replacement SAVE, the previous file is completely recovered during the abort following error detection. It's a handy technique which we have never seen published. Look at the code if you are curious—the key is a disk initialization before closing the open write file.

Preparing the Generator

You are now ready to enter, save, and test the code generator program. In Chapter 10 you will find the program called ML PROG GEN. Type this program into your computer and save it on your disk.

Notice that the first line of the program not only contains the program title, but is structured as a SAVE command contained within a remarks line. If you list this line and then type spaces over the beginning part, you are left with a SAVE command for that program. It will execute when you press RETURN. This type of first line is used on all our BASIC programs to avoid typing errors while saving the programs.

Now load ML PROG GEN and add the following three DATA lines to the end of the program. They contain two deliberate errors to illustrate the error-checking capability of ML PROG GEN. The DATA lines must be entered exactly as shown to allow a valid test.

```
4000 DATA 00 79 4D 15 79 4C 1E 79 CA
4001 DATA 4C 27 79 20 30 69 .. .. 8D
4002 DATA B4 60 3B CB 57 3B 96 15 QQ
```

Now run the program. When you are asked to enter a filename, type something simple like TEST. You will be keeping this file only long enough to make sure the program is working.

As the program runs, it should tell you that there are errors in lines 4000 and 4001 and in columns 3, 6, 7, and 8. If you do not get this result, you have an error in either the program or in the DATA statements. Do not proceed beyond this point until you get this exact result.

Now you can correct the data errors by changing two bytes in lines 4000 and 4001 so that they read:

```
4000 DATA 00 79 4C 15 79 4C 1E 79 CA
4001 DATA 4C 27 79 20 30 79 4C 72 8D
```

Now save the program and then run it again. No error messages should be produced, and the TEST file should be saved on disk. If you get any indicated errors, you must find the source of the problem and correct it before proceeding.

Be sure you have saved a good copy of ML PROG GEN and then type:

```
LOAD"TEST",8,1
```

(The ,1 is for 64 users, but won't hurt the rest of you.)

Now spot-check a couple of locations to see if the correct code loaded into the right addresses.

PRINT PEEK (30977) Should produce 21.
PRINT PEEK (30980) Should produce 30.

You should feel free to do a more exhaustive test if you wish, but successful results to this point should be adequate to insure that the program is okay.

Finally, load the ML PROG GEN program again, delete the three DATA lines, and use program line 1000 to save the program for future use. You can also delete the TEST file. It is no longer needed.

Creating the Subroutines

Now that the machine language program generator has been tested and saved, we are going to make three new programs from it. These will be the programs which actually create the machine language program files.

Load ML PROG GEN from disk. Then find the listing for DFH SUBS GEN in Chapter 10. It will have only a title line and a lot of DATA statements. Enter this program as a modification to ML PROG GEN, which you already have in memory. Be sure to use the line numbers as shown. We want the new title line at the start and the DATA lines at the end. The DATA line numbers are important because any detected errors will be referred to by the line numbers as shown in the listing.

Save this new program as often as you like during the typing-in process, but be sure to use the new name contained in the new first line. You don't want to destroy your copy of ML PROG GEN.

When you have entered and saved the program, run it as many times as needed to locate and correct any errors (as explained above). This time, when asked for a program filename, enter the filename:

→ **DFH SUBS$79**

You must use this exact name because the bootstrap program for the Data File Handler programs will load the subroutines using this name.

When the program runs to completion without any detected errors, you will have a fully functional subroutine set stored on disk and ready to use.

Creating the Editor Programs

Now you need to repeat the process you used to create the machine language subroutines. This time you will be creating the DFH Editor program. If you want the DFH Editor for both the Commodore 64 and the PET, it will be repeated twice.

Again, load the ML PROG GEN program and modify it by adding the appropriate editor code for your machine. The listing for DFH ED.6 is for the Commodore 64, and DFH ED.P GEN is for PET computers.

When you assign the machine language program filename, use DFH ED.C64$90 for Commodore 64, and DFH ED.PET$70 for PET computers. Once again, the filenames must be exactly as shown because they will be loaded by the DFH bootstrap program.

Moving from the 64 to the PET

If you have chosen to type in the DFH programs on a Commodore 64, you will find that they will not immediately run on a PET. This is because the LOAD point bytes were set to $0801 when the programs were saved on the Commodore 64. To load correctly in a PET, the LOAD point bytes need to be changed to $0401.

Several articles have been published showing methods for changing the LOAD point bytes, but they are not necessary in this case. The DFH programs have a built-in capability for this change.

Simply use the bootstrap program (DFH BOOT) with your 64 to load one of the other BASIC programs. By the time it is loaded, your 64 will have been reconfigured to look like a PET. At the first opportunity, exit from the program using the Q (quit) option rather than returning to the bootstrap main menu.

Now, list the first line of the program and use the built-in SAVE command to save the program.

Next, load DFH BOOT. Since your 64 is now configured like a PET, the program will be automatically relocated to $0401. Do not run the program; simply list the first line and use the built-in SAVE command to save DFH BOOT from its present location at $0401.

You now have both DFH BOOT, and whichever other program you selected, saved with new LOAD point bytes. You can now run DFH BOOT and repeat the first part of this procedure to resave the remaining four BASIC programs.

Chapter 10

Program Listings

Machine Language Programs

This section contains the listings used to create the three machine language subroutines. Please refer to Chapter 9 for complete instructions on how to enter these programs.

The following list contains the filenames of each program along with a very brief statement of its purpose or function.

ML PROG GEN. This is the control program for generating machine language program files and storing them on disk. The code is generated from DATA statements which are added to ML PROG GEN. The program contains extensive error checking and can only function with DATA statements which are created by ML DATA GEN. Please refer to Chapter 9 for specific instructions for using this program.

DFH SUBS GEN. This set of DATA statements is added to ML PROG GEN. When the combined program is run, it will create the machine language program DFH SUBS$79 and save it on disk for your future use. DFH SUBS$79 is the subroutine set used by all the BASIC programs of the DFH family.

DFH ED.P GEN. This set of DATA statements is added to ML PROG GEN. When the combined program is run, it will create the machine language program DFH ED.PET$70 and save it on disk for your future use. DFH ED.PET$70 is the PET version of the DFH Editor.

DFH ED.6 GEN. This set of DATA statements is added to ML PROG GEN. When the combined program is run, it will create the machine language program DFH ED.C64$90 and save it on disk for your future use. DFH ED.C64$90 is the Commodore 64 version of the DFH Editor.

ML PROG GEN

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:ML PROG GEN",8          :rem 2Ø9
1Ø1Ø :                                    :rem 252
1Ø2Ø REM" GENERAL PROGRAM LINES ARE 1ØØØ-2ØØØ"
                                            :rem 4
1Ø3Ø :                                    :rem 254
1Ø7Ø :                                      :rem 2
```

```
1Ø8Ø PRINT "{CLR}{RVS} ------- ML PROGRAM GENERATO
     R -------- {OFF}"                          :rem 2Ø4
1Ø9Ø PRINT "{DOWN}THIS PROGRAM CREATES AND SAVES T
     O DISK,"                                   :rem 82
11ØØ PRINT "A MACHINE LANGUAGE PROGRAM{2 SPACES}FR
     OM THE"                                    :rem 58
111Ø PRINT "'DATA' STATEMENTS IN THIS PROGRAM.
     {2 SPACES}THE"                             :rem 25Ø
112Ø PRINT "FIRST DATA LINE STARTS AT LINE '4ØØØ'"
                                                :rem 15
113Ø PRINT "AND SHOULD INCREMENT BY '1'."       :rem 37
114Ø PRINT "{DOWN}EACH DATA LINE CONTAINS NINE HEX
      BYTES."                                   :rem 3Ø
115Ø PRINT "THE FIRST EIGHT BYTES REPRESENT PROGRA
     M"                                         :rem 2Ø1
116Ø PRINT "BYTES.{2 SPACES}THE NINTH BYTE IS A CH
     ECKSUM"                                    :rem 2Ø
117Ø PRINT "VALUE WHICH CAUSES THE TOTAL LINE SUM"
                                                :rem 2Ø8
118Ø PRINT "TO EQUAL ZERO."                     :rem 36
119Ø PRINT "{DOWN}THE FINAL LINE CONTAINS EIGHT 'C
     OLUMN"                                     :rem 229
12ØØ PRINT "CHECKSUMS' AND A 'QQ' TERMINATOR CODE.
     "                                          :rem 179
121Ø PRINT "EACH COLUMN SUM VALUE CAUSES THE SUM"
                                                :rem 128
122Ø PRINT "FOR THAT COLUMN OF DATA TO BE ZERO."
                                                :rem 195
123Ø PRINT "{DOWN}THE '..' CHARACTERS IN THE LAST
     {SPACE}PROGRAM"                            :rem 215
124Ø PRINT "LINE ARE USED FOR PADDING AT THE END."
                                                :rem 34
125Ø GOSUB 129Ø: GOTO 142Ø                      :rem 78
126Ø :                                          :rem 3
127Ø REM"--SUB-- WAIT FOR OPERATOR ---"         :rem 159
128Ø :                                          :rem 5
129Ø : PRINT "{4 SPACES}{DOWN}{RVS} PRESS ANY KEY
     {SPACE}TO CONTINUE {OFF}{DOWN}"            :rem 3
13ØØ : GET KB$: IF KB$<>"" THEN 13ØØ            :rem 186
131Ø : GET KB$: IF KB$="" THEN 131Ø             :rem 127
132Ø RETURN                                     :rem 166
133Ø :                                          :rem 1
134Ø REM"--SUB-- DISK ERROR CHECK ---"          :rem 4Ø
135Ø :                                          :rem 3
136Ø : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN=63 TH
     EN RETURN                                  :rem 249
137Ø IF EN>19 THEN PRINT "{RVS} DISK ERROR {OFF}"
                                                :rem 238
138Ø PRINT "{RVS}"EN;EM$;ET;ES: RETURN          :rem 244
139Ø :                                          :rem 7
```

```
1400 REM"--- LINE-BY-LINE CHECKSUM TEST ---"
                                          :rem 211
1410 :                                      :rem Ø
1420 : DIM SM(9): CR$=CHR$(13): LN=3999: EL=Ø
                                           :rem 18
143Ø RE$="": OPEN 15,8,15: PRINT          :rem 147
144Ø : INPUT "ML PROGRAM FILE NAME{2 SPACES}";NA$
                                          :rem 183
145Ø : INPUT "SAVE ON DISK DRIVE #{4 SPACES}Ø
     {3 LEFT}";DR$                        :rem 1Ø4
146Ø IF DR$<>"Ø" AND DR$<>"1" THEN 145Ø   :rem 119
147Ø : OPEN 8,8,8,RE$+DR$+":"+NA$+",P,W": EC=1
                                          :rem 41
148Ø GOSUB 136Ø: IF EN=Ø THEN 154Ø        :rem 217
149Ø CLOSE 8: EC=Ø: IF EN<>63 THEN 196Ø   :rem 233
15ØØ INPUT "REPLACE EXISTING FILE{3 SPACES}Y
     {3 LEFT}";KB$                         :rem 61
151Ø IF LEFT$(KB$,1)="Y" THEN RE$="@": GOTO 147Ø
                                          :rem 2ØØ
152Ø GOTO 144Ø                            :rem 2Ø2
153Ø :                                      :rem 3
154Ø : PRINT "{DOWN}TESTING DATA LINE & CREATING M
     L CODE{DOWN}"                        :rem 126
155Ø : LN=LN+1: PRINT "{UP}";LN: READ DL$: IF RIGH
     T$(DL$,2)="QQ" THEN EL=2             :rem 1Ø9
156Ø CF=Ø: LS=Ø: SP=Ø: CH=Ø               :rem 132
157Ø FOR JJ=1 TO LEN(DL$)-EL: CV=ASC(MID$(DL$,JJ,1
     ))                                    :rem 83
158Ø IF CV>57 AND CV<65 THEN 165Ø         :rem 174
159Ø CV=CV-48+7*(CV>57): IF CV<Ø OR CV>15 THEN 165
     Ø                                    :rem 158
16ØØ CH=CH+1                              :rem 118
161Ø IF CF=Ø THEN CF=1: SP=SP+1: HN=CV*16: SM(SP)=
     SM(SP)+CV*16: GOTO 165Ø              :rem 246
162Ø CF=Ø: HN=HN+CV: SM(SP)=(SM(SP)+CV) AND 255
                                           :rem 73
163Ø IF (EL=Ø AND EC=1 AND (JJ<LEN(DL$)-2)) THEN P
     RINT# 8,CHR$(HN);                     :rem 14
164Ø LS=(LS+HN) AND 255                   :rem 199
165Ø : IF MID$(DL$,JJ,1)="." THEN SP=SP+.5: CH=CH+
     1                                    :rem 141
166Ø NEXT JJ: IF (LS=Ø AND EL=Ø AND CH=18) THEN 15
     5Ø                                   :rem 149
167Ø IF EL=2 THEN 176Ø: REM"COL SUM       :rem 1Ø9
168Ø :                                      :rem 9
169Ø PRINT "{RVS} DATA ERROR IN LINE{2 SPACES}#";L
     N;"{LEFT} {OFF}"                     :rem 188
17ØØ PRINT LN;"DATA ";DL$: GOSUB 186Ø       :rem Ø
171Ø PRINT "{RVS} C {OFF} CONTINUE CHECKING OR
     {RVS} E {OFF} END ?"                 :rem 194
```

```
1720 : GOSUB 1300: IF KB$="C" THEN PRINT "CHECKING
      LINE{DOWN}": GOTO 1550                    :rem 57
1730 IF KB$<>"E" THEN 1720                       :rem 7
1740 END                                        :rem 163
1750 :                                           :rem 7
1760 : ER=0: EP$="{RVS} COLUMN CHECKSUM ERROR(S) I
     N COLUMN(S)"+CR$                           :rem 206
1770 FOR JJ=1 TO 8                              :rem 147
1780 IF SM(JJ)<>0 THEN ER=1: EP$=EP$+STR$(JJ)+","
                                                :rem 115
1790 NEXT JJ: IF ER=0 AND EC=1 THEN 1940         :rem 55
1800 IF ER=0 THEN PRINT "NO OTHER ERRORS.": GOTO 1
     960                                        :rem 201
1810 GOSUB 1860: PRINT LEFT$(EP$,LEN(EP$)-1)
                                                :rem 172
1820 END                                        :rem 162
1830 :                                           :rem 6
1840 REM"--- CLOSE ML FILES ON ERROR ---" :rem 172
1850 :                                           :rem 8
1860 : IF EC=0 THEN RETURN                      :rem 156
1870 IF RE$="@" THEN PRINT# 15,"I";DR$: GOTO 1890
                                                 :rem 19
1880 CLOSE 8: PRINT# 15,"S";DR$;":";NA$          :rem 46
1890 : GOSUB 1360: CLOSE 8: EC=0                :rem 109
1900 PRINT "{DOWN}{RVS} ML DISK FILE ABORTED.
     {OFF}{DOWN}": RETURN                       :rem 142
1910 :                                           :rem 5
1920 REM"--- PROGRAM TERMINATION ---"           :rem 100
1930 :                                           :rem 7
1940 : PRINT "{2 DOWN} CHECKSUMS ALL OK !!":rem 86
1950 PRINT "{DOWN}{RVS} ML PROGRAM SAVED ON DISK
     {OFF}{DOWN}"                                :rem 82
1960 : CLOSE 8: CLOSE 15: IF EN=0 THEN END:rem 157
1970 PRINT "{RVS}{4 SPACES}PROGRAM TERMINATED
     {4 SPACES}{OFF}": END                       :rem 92
1980 :                                          :rem 12
1990 REM"--- DATA FOR M.L. PROGRAM ---"          :rem 23
2000 :                                          :rem 252
```

DFH SUBS GEN

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
500 REM SAVE "@0:DFH SUBS GEN",8                :rem 227
4000 DATA 00 79 4C 15 79 4C 1E 79 CA            :rem 220
4001 DATA 4C 27 79 20 30 79 4C 10 EF            :rem 199
4002 DATA 7F 00 00 00 00 00 00 20 61            :rem 102
4003 DATA 30 79 20 C4 79 4C CD 79 68            :rem 217
4004 DATA 20 30 79 20 C4 79 4C 60 2E            :rem 180
4005 DATA 7D 20 30 79 20 D8 7A 4C FC            :rem 233
```

```
4006 DATA C6 7E A9 20 20 D2 FF A9 59             :rem 1
4007 DATA 9D 20 D2 FF A9 00 8D 8E AE            :rem 31
4008 DATA 7F AE FE FF E0 48 F0 0F AF            :rem 84
4009 DATA A9 06 8D 8E 7F E0 1B F0 CC            :rem 33
4010 DATA 03 A2 00 2C A2 09 2C A2 B6           :rem 198
4011 DATA 12 A0 00 BD 96 7E 99 B1 33           :rem 208
4012 DATA 7F E8 C8 C0 09 D0 F4 A9 9B            :rem 21
4013 DATA C4 A0 D2 AE FE FF E0 48 F7            :rem 59
4014 DATA D0 04 A9 D1 A0 B8 85 FB DA            :rem 15
4015 DATA B9 00 00 8D 85 7F A4 FB 17           :rem 234
4016 DATA B9 00 00 85 FB B9 01 00 0D           :rem 197
4017 DATA 85 FC A0 00 E0 48 D0 02 E5           :rem 222
4018 DATA A0 03 8C 8F 7F B9 2A 00 E0           :rem 244
4019 DATA 8D 86 7F B9 2B 00 8D 87 76           :rem 246
4020 DATA 7F B9 2C 00 8D 88 7F B9 4F            :rem 11
4021 DATA 2D 00 8D 89 7F B9 2E 00 57           :rem 223
4022 DATA 8D 8A 7F B9 2F 00 8D 8B 6A            :rem 20
4023 DATA 7F B9 77 00 8D 8C 7F B9 00           :rem 248
4024 DATA 78 00 8D 8D 7F 60 20 D8 97           :rem 220
4025 DATA 7A 20 E1 7B 4C 85 7C 18 A5           :rem 235
4026 DATA A5 0D 6A 85 0F A5 0C 6A 35           :rem 242
4027 DATA 85 0E E6 0E D0 02 E6 0F B2           :rem 244
4028 DATA A5 0F D0 4A A5 0E C9 01 B5           :rem 252
4029 DATA D0 44 20 CF 7A A6 0C A4 2D           :rem 253
4030 DATA 0D 20 A1 7B 20 C0 7B A5 B7           :rem 223
4031 DATA 10 18 69 03 85 1A A5 11 17           :rem 157
4032 DATA 69 00 85 1B A0 02 B1 1A 8A           :rem 195
4033 DATA 91 17 88 10 F9 A5 0C D0 46           :rem 205
4034 DATA 02 C6 0D C6 0C A5 0D D0 D7           :rem 249
4035 DATA 27 A5 0C D0 23 20 9D 7B FD           :rem 243
4036 DATA 20 CB 7B 20 F3 7C 20 CF 1C           :rem 251
4037 DATA 7A 30 FB 4C D8 7A A5 0E 0A            :rem 18
4038 DATA D0 02 C6 0F C6 0E A6 0E D1           :rem 253
4039 DATA A4 0F 20 A1 7B 20 C0 7B B6           :rem 232
4040 DATA A5 0E 85 0A A5 0F 85 0B 7A           :rem 238
4041 DATA A5 0A 85 08 A5 0B 85 09 86           :rem 203
4042 DATA 18 26 0A 26 0B A5 0B C5 12           :rem 195
4043 DATA 0D F0 04 90 0A B0 06 A5 0A           :rem 204
4044 DATA 0A C5 0C F0 33 B0 47 20 EB           :rem 228
4045 DATA 96 7B A0 02 B1 17 99 19 D3           :rem 208
4046 DATA 00 88 10 F8 18 A5 17 69 33           :rem 179
4047 DATA 03 85 17 90 02 E6 18 20 B1           :rem 165
4048 DATA D6 7B A2 02 B5 19 95 20 88           :rem 211
4049 DATA CA 10 F9 20 2F 7B A5 16 A8           :rem 248
4050 DATA D0 06 E6 0A D0 02 E6 0B 77           :rem 218
4051 DATA 20 96 7B 20 D6 7B A2 02 BA           :rem 219
4052 DATA B5 12 95 20 CA 10 F9 20 91           :rem 191
4053 DATA 2F 7B A5 16 F0 09 20 9D E5           :rem 236
4054 DATA 7B 20 CB 7B 4C DE 79 20 5C            :rem 10
4055 DATA 9D 7B A5 17 85 1A A5 18 D0           :rem 238
```

```
4056 DATA 85 1B 20 96 7B A0 02 B1 DC        :rem 224
4057 DATA 17 91 1A 88 10 F9 4C 46 1B        :rem 210
4058 DATA 7A A0 00 B1 FB 49 F6 91 6A        :rem 249
4059 DATA FB 60 A2 20 B5 03 48 BD 26        :rem 227
4060 DATA 90 7F 95 03 68 9D 90 7F 45        :rem 213
4061 DATA CA 10 F1 60 A2 03 B5 20 5B        :rem 205
4062 DATA 48 B5 1C 95 20 68 95 1C 19        :rem 199
4063 DATA CA 10 F3 60 A6 06 A0 00 87        :rem 200
4064 DATA 84 1F A5 05 D1 1D D0 0C E9        :rem 245
4065 DATA CA F0 14 E0 01 D0 05 C8 B4        :rem 232
4066 DATA 84 1F D0 01 C8 C4 1C 90 54        :rem 218
4067 DATA EB E0 02 90 02 84 1F A5 59        :rem 219
4068 DATA 1F 18 65 1D 85 1D 90 02 13        :rem 189
4069 DATA E6 1E 98 38 E5 1F 85 1F 84        :rem 251
4070 DATA 60 A5 1C 85 1F A5 20 85 F1        :rem 213
4071 DATA 23 A6 06 D0 03 4C 52 7B 45        :rem 191
4072 DATA C9 00 F0 09 20 EA 7A 20 9A        :rem 222
4073 DATA FA 7A 20 EA 7A A5 1C F0 57         :rem 19
4074 DATA 03 20 FA 7A A0 01 84 16 2E        :rem 201
4075 DATA 88 84 19 A6 07 A5 1F D0 9A        :rem 233
4076 DATA 09 E0 00 D0 24 A5 23 D0 8B        :rem 204
4077 DATA 20 60 A5 23 F0 FB E0 00 ED        :rem 235
4078 DATA F0 03 20 8B 7B B1 1D D1 48        :rem 228
4079 DATA 21 D0 0C C8 C4 23 90 02 C2        :rem 207
4080 DATA B0 07 C4 1F 90 EF 60 90 F7        :rem 237
4081 DATA 03 20 8B 7B 60 A5 16 48 74        :rem 185
4082 DATA A5 19 85 16 68 85 19 60 41        :rem 172
4083 DATA A6 0A A4 0B 4C A1 7B A6 93        :rem 253
4084 DATA 08 A4 09 86 21 84 22 A5 59        :rem 181
4085 DATA 10 85 17 A5 11 85 18 A2 5F        :rem 189
4086 DATA 03 18 A5 21 65 17 85 17 07        :rem 160
4087 DATA A5 22 65 18 85 18 CA D0 85        :rem 214
4088 DATA F0 60 A0 02 B1 17 99 12 9B        :rem 200
4089 DATA 00 88 10 F8 60 A0 02 B9 B5        :rem 201
4090 DATA 12 00 91 17 88 10 F8 60 56        :rem 156
4091 DATA A0 02 B1 17 99 1C 00 88 59        :rem 186
4092 DATA 10 F8 60 A2 46 A0 C1 20 2F        :rem 202
4093 DATA 3F 7C 90 08 20 D8 7A A2 99        :rem 233
4094 DATA E9 4C B1 7F A0 00 B1 0C 3E        :rem 255
4095 DATA 85 03 C8 A9 00 C6 0A F0 47        :rem 220
4096 DATA 02 B1 0C 09 80 85 04 A2 8D        :rem 198
4097 DATA 46 A0 C4 20 3F 7C 90 04 E7        :rem 222
4098 DATA A9 2A D0 04 A0 00 B1 0C FC        :rem 244
4099 DATA 85 05 A2 C6 A0 D3 20 3F 3C        :rem 231
4100 DATA 7C 90 04 A9 00 F0 04 A0 B3        :rem 192
4101 DATA 03 B1 08 85 06 A2 46 A0 31        :rem 163
4102 DATA CF 20 3F 7C A0 00 B1 0C F9        :rem 245
4103 DATA C9 44 F0 02 A9 00 85 07 CC        :rem 210
4104 DATA 60 86 0E 84 0F AD 86 7F C7        :rem 248
4105 DATA 85 08 AD 87 7F 85 09 A0 92        :rem 220
```

```
4106 DATA 00 B1 08 C5 0E D0 07 C8 D5          :rem 220
4107 DATA B1 08 C5 0F F0 18 18 A5 AE          :rem 243
4108 DATA 08 69 07 85 08 90 02 E6 83          :rem 171
4109 DATA 09 A5 08 CD 88 7F A5 09 C8          :rem 253
4110 DATA ED 89 7F 90 DA 60 C8 B1 C8           :rem 20
4111 DATA 08 85 0A C8 B1 08 85 0C 57          :rem 201
4112 DATA C8 B1 08 85 0D 18 60 AD C8          :rem 232
4113 DATA 88 7F 85 08 AD 89 7F 85 32          :rem 234
4114 DATA 09 A0 00 B1 08 C5 03 D0 06          :rem 174
4115 DATA 07 C8 B1 08 C5 04 F0 24 9B          :rem 213
4116 DATA 18 A0 02 B1 08 65 08 48 D8          :rem 184
4117 DATA C8 B1 08 65 09 85 09 68 1B          :rem 200
4118 DATA 85 08 CD 8A 7F A5 09 ED 02            :rem 4
4119 DATA 8B 7F 90 D5 20 D8 7A A0 7F            :rem 6
4120 DATA 24 4C B4 7F A5 08 85 17 14          :rem 201
4121 DATA A5 09 85 18 A0 04 B1 17 49          :rem 180
4122 DATA C9 01 F0 08 20 D8 7A A2 2A          :rem 218
4123 DATA 7A 4C B1 7F A0 06 B1 17 9C          :rem 241
4124 DATA 85 0C 88 B1 17 85 0D A5 E8          :rem 228
4125 DATA 17 18 69 04 85 10 A5 18 12          :rem 154
4126 DATA 69 00 85 11 60 20 85 7C 80          :rem 156
4127 DATA A9 01 85 08 A9 00 85 09 92          :rem 179
4128 DATA 85 19 85 1A 20 9D 7B 20 6B          :rem 209
4129 DATA C0 7B 20 75 7E A5 12 F0 0B          :rem 225
4130 DATA 2C A6 06 F0 22 A0 00 A2 D4          :rem 202
4131 DATA 01 84 1F A5 05 D1 13 D0 FE          :rem 218
4132 DATA 07 E4 06 F0 0E E8 D0 06 53          :rem 212
4133 DATA E4 06 D0 02 E6 1F C8 C4 B3          :rem 243
4134 DATA 12 90 E8 A5 1F F0 06 E6 D6          :rem 237
4135 DATA 19 D0 02 E6 1A E6 08 D0 57          :rem 215
4136 DATA 02 E6 09 A5 0C C5 08 A5 EC          :rem 245
4137 DATA 0D E5 09 B0 B7 A2 C6 A0 96          :rem 247
4138 DATA D4 20 3F 7C B0 0B A0 03 F3          :rem 232
4139 DATA A5 19 91 08 88 A5 1A 91 D1          :rem 212
4140 DATA 08 60 A5 0D F0 03 4C D2 D5          :rem 218
4141 DATA 7C A9 00 85 09 85 19 85 2A          :rem 197
4142 DATA 1C A9 01 85 07 85 08 85 9C          :rem 198
4143 DATA 06 85 20 20 9D 7B 20 C0 3D          :rem 191
4144 DATA 7B A4 12 F0 14 A0 00 B1 7A          :rem 207
4145 DATA 13 C5 05 F0 0C E6 1C C8 5D          :rem 245
4146 DATA C4 12 90 F3 88 A9 00 85 F1          :rem 211
4147 DATA 07 20 C4 7D A5 1C F0 04 E3          :rem 225
4148 DATA A5 08 85 19 E6 08 18 A5 0A          :rem 211
4149 DATA 1C 65 07 65 20 85 20 A9 A5          :rem 190
4150 DATA 00 85 1C C8 C4 12 B0 04 0D          :rem 196
4151 DATA 88 4C 8D 7D A9 00 85 1A DA            :rem 1
4152 DATA 20 4B 7D 4C D8 7A 86 15 DF            :rem 2
4153 DATA 84 1A A9 30 A2 04 20 DD E6          :rem 222
4154 DATA 7D A2 12 20 DD 7D A2 16 9D          :rem 245
4155 DATA 20 DD 7D A2 00 F0 0C 9D 4B          :rem 253
```

```
4156 DATA 62 7F E8 9D 62 7F E8 9D 34        :rem 15
4157 DATA 62 7F 6Ø A5 Ø3 9D 62 7F 99       :rem 227
4158 DATA A5 Ø4 29 7F DØ Ø2 A9 2Ø 14       :rem 197
4159 DATA E8 9D 62 7F EØ Ø1 DØ Ø4 E5       :rem 244
416Ø DATA A2 ØB DØ E7 A2 Ø6 A5 Ø8 47       :rem 22Ø
4161 DATA 2Ø 45 7E A2 14 A5 2Ø 2Ø 82       :rem 169
4162 DATA 45 7E A2 18 A5 1C 2Ø 45 5D       :rem 214
4163 DATA 7E A9 62 AC 8F 7F 99 77 AD        :rem 42
4164 DATA ØØ A9 7F 99 78 ØØ 2Ø D8 CF       :rem 232
4165 DATA 7A 38 AD 62 7F 2Ø B7 7F 6A         :rem 6
4166 DATA 2Ø D8 7A AC 8F 7F AD 8C 9B        :rem 59
4167 DATA 7F 99 77 ØØ AD 8D 7F 99 1F        :rem 16
4168 DATA 78 ØØ A6 15 A4 1A 6Ø 85 2A       :rem 198
4169 DATA 23 A5 23 FØ 29 C6 23 FE 15       :rem 222
417Ø DATA 62 7F BD 62 7F C9 3A DØ AE        :rem 32
4171 DATA FØ A9 3Ø 9D 62 7F CA FE F1        :rem 34
4172 DATA 62 7F BD 62 7F C9 3A DØ AE        :rem 34
4173 DATA ØA A9 3Ø 9D 62 7F CA FE D7        :rem 35
4174 DATA 62 7F E8 E8 DØ D3 6Ø AC AØ        :rem 15
4175 DATA FE FF CØ 42 DØ 19 A4 12 62       :rem 249
4176 DATA FØ 15 A5 13 CD 86 7F A5 CC        :rem 16
4177 DATA 14 ED 87 7F 9Ø Ø9 A5 17 A4       :rem 236
4178 DATA 91 13 C8 A5 18 91 13 6Ø D3       :rem 193
4179 DATA 4C CF B3 4C BC F5 4C 87 62        :rem 41
418Ø DATA B7 4C 57 C3 4C 73 F5 4C E3         :rem 4
4181 DATA Ø2 C7 4C B1 7E 4C C1 7E 31       :rem 245
4182 DATA 4C EF A7 EØ 7A DØ Ø9 A9 42        :rem 13
4183 DATA 9F 85 22 A9 A2 4C 45 A4 3A       :rem 24Ø
4184 DATA A2 13 2C A2 Ø4 6C ØØ Ø3 ØA       :rem 187
4185 DATA A2 57 AØ C3 2Ø 3F 7C BØ 19       :rem 227
4186 DATA ØA A5 ØA FØ Ø6 AØ ØØ B1 ØØ       :rem 197
4187 DATA ØC DØ Ø2 A9 ØØ 85 Ø5 A2 4D       :rem 2Ø6
4188 DATA 57 AØ D3 2Ø 3F 7C 9Ø Ø5 C6       :rem 223
4189 DATA A2 E9 4C B1 7F A2 ØØ A4 B3         :rem 5
419Ø DATA ØA 88 B1 ØC C9 8Ø 9Ø Ø7 D1       :rem 216
4191 DATA 29 7F 91 ØC 4C Ø5 7F E4 Ø7       :rem 231
4192 DATA Ø5 FØ Ø2 29 3F 91 ØC CØ 44       :rem 195
4193 DATA ØØ DØ Ø3 4C D8 7A 88 4C BB       :rem 248
4194 DATA FØ 7E A9 ØØ 8D ØF 79 8D 47         :rem 1
4195 DATA 1Ø 79 8D 13 79 8D 14 79 44       :rem 199
4196 DATA AE 85 7F 2Ø C6 FF 2Ø E4 65        :rem 11
4197 DATA FF A8 AE 8E 7F B5 9Ø DØ 89        :rem 57
4198 DATA 2A CØ ØD FØ ØB EE 13 79 94       :rem 253
4199 DATA DØ EC EE 14 79 4C 24 7F DA        :rem 43
42ØØ DATA 2Ø CF 7A EE ØF 79 DØ Ø3 4E         :rem 3
42Ø1 DATA EE 1Ø 79 AD ØF 79 CD 11 76       :rem 253
42Ø2 DATA 79 DØ D3 AD 1Ø 79 CD 12 CF         :rem 6
42Ø3 DATA 79 DØ CB 2Ø CF 7A 3Ø FB 58         :rem 7
42Ø4 DATA 2Ø CC FF 6Ø 46 41 24 28 E2       :rem 212
42Ø5 DATA 3Ø 3Ø 3Ø 29 B2 CA 28 46 5D       :rem 19Ø
```

```
4206 DATA 41 24 28 3Ø 29 2C 3Ø 3Ø 8E       :rem 161
4207 DATA 3Ø 2C 3Ø 3Ø 3Ø 29 ØØ 3Ø BB       :rem 154
4208 DATA 33 2D 32 39 2D 38 34 .. 9C       :rem 181
4209 DATA D6 C2 D4 63 61 25 91 15 QQ       :rem 249
```

DFH ED.P GEN

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
5ØØ REM SAVE "@Ø:DFH ED.P GEN",8           :rem 173
4ØØØ DATA ØØ 7Ø 4C ØD 7Ø 4C DE 74 29       :rem 2Ø5
4ØØ1 DATA 4C ØD 7Ø 4C ØD 7Ø F3 AE CD         :rem 9
4ØØ2 DATA FE FF EØ 1B FØ 1D A2 6C ED        :rem 63
4ØØ3 DATA BD 65 74 9D F9 73 CA DØ C7        :rem 21
4ØØ4 DATA F7 8E FF 87 EC FF 83 FØ 97        :rem 6Ø
4ØØ5 DATA ØA A9 87 8D F7 73 A9 BØ 76       :rem 249
4ØØ6 DATA 8D F8 73 2Ø F2 74 A2 ØF D1       :rem 235
4ØØ7 DATA BD Ø2 7B 95 79 CA 1Ø F8 E6         :rem 2
4ØØ8 DATA A9 Ø8 8D B2 78 A2 ØØ AØ 56       :rem 214
4ØØ9 DATA 7Ø E4 34 98 E5 35 BØ Ø4 12       :rem 186
4Ø1Ø DATA 86 34 84 35 4C Ø4 75 85 43       :rem 163
4Ø11 DATA B5 86 AD A2 F2 8E ØC 7Ø 7A         :rem 4
4Ø12 DATA A6 78 EØ Ø2 DØ 11 A6 77 Ø2       :rem 187
4Ø13 DATA DØ ØD BA BD Ø2 Ø1 CD Ø1 DB         :rem 4
4Ø14 DATA 74 FØ 14 DØ Ø2 A4 B6 A6 B6       :rem 219
4Ø15 DATA AD A5 B5 C9 3A 9Ø Ø1 6Ø Ø5       :rem 219
4Ø16 DATA C9 2Ø FØ 45 4C 22 7B 2Ø D9       :rem 213
4Ø17 DATA 75 7Ø BØ Ø3 4C 16 73 84 ØF       :rem 184
4Ø18 DATA B6 A2 ØØ 86 81 86 Ø5 A4 72       :rem 186
4Ø19 DATA 77 B9 ØØ Ø2 5D B4 78 FØ 55       :rem 2Ø9
4Ø2Ø DATA 13 C9 8Ø FØ 13 E6 Ø5 E8 CE       :rem 231
4Ø21 DATA BD B3 78 1Ø FA BD B4 78 25         :rem 5
4Ø22 DATA DØ E5 FØ C1 E8 C8 DØ E1 39         :rem 8
4Ø23 DATA 84 77 A5 Ø5 ØA AA BD D1 19       :rem 24Ø
4Ø24 DATA 7A 48 BD DØ 7A 48 2Ø 73 5C       :rem 234
4Ø25 DATA 7Ø 4C 7Ø ØØ 78 AD 2A 74 11       :rem 19Ø
4Ø26 DATA 85 9Ø AD 2B 74 85 91 58 31       :rem 2ØØ
4Ø27 DATA 6Ø BD Ø5 Ø1 CD 6Ø 74 DØ 6C       :rem 222
4Ø28 DATA ØB BD Ø6 Ø1 CD 61 74 DØ BF       :rem 252
4Ø29 DATA Ø3 2Ø 62 74 A5 97 C9 FF Ø3       :rem 21Ø
4Ø3Ø DATA FØ Ø5 CD 4D 75 FØ ØB 8D F4         :rem 5
4Ø31 DATA 4D 75 A9 1Ø 8D 4E 75 4C E9       :rem 247
4Ø32 DATA 29 74 A5 91 CD 2B 74 FØ D1       :rem 234
4Ø33 DATA F6 AD 4E 75 FØ Ø5 CE 4E 89        :rem 23
4Ø34 DATA 75 DØ EC CE 4F 75 DØ E7 86        :rem 25
4Ø35 DATA A9 Ø4 8D 4F 75 A9 ØØ 85 D4       :rem 229
4Ø36 DATA 97 A9 Ø2 85 A8 DØ D8 A9 4Ø       :rem 231
4Ø37 DATA 2F 2C A9 5E 2C A9 5D 2C 4Ø        :rem 1Ø
4Ø38 DATA A9 26 85 B3 C9 5D DØ Ø3 ØØ       :rem 217
4Ø39 DATA A9 Ø1 2C A9 ØØ 85 9D 2Ø 3F       :rem 217
```

```
4Ø4Ø DATA 47 77 A4 D1 DØ Ø3 4C 1D 91      :rem 21Ø
4Ø41 DATA 74 A5 7C 1Ø ØC A5 B3 C9 2E      :rem 24Ø
4Ø42 DATA 5E DØ Ø3 4C EE 7E 4C 63 68        :rem Ø
4Ø43 DATA 75 A9 6Ø 2Ø DE 77 2Ø AØ 4D      :rem 213
4Ø44 DATA 77 2Ø CC FF A5 D2 2Ø 41 C6      :rem 244
4Ø45 DATA 74 2Ø 7Ø ØØ C9 2C DØ 19 1E      :rem 192
4Ø46 DATA A5 B3 C9 5E FØ 1Ø C9 26 92      :rem 239
4Ø47 DATA FØ ØC 2Ø 7Ø ØØ C9 24 DØ B7      :rem 2Ø2
4Ø48 DATA ØC 2Ø 7B 7F BØ Ø7 4C 1D BA      :rem 253
4Ø49 DATA 74 C9 ØØ DØ DC A9 6Ø 85 89      :rem 235
4Ø5Ø DATA D3 2Ø 4A 74 2Ø 5Ø 74 2Ø 4B      :rem 166
4Ø51 DATA 2C 74 A5 D3 2Ø 32 74 2Ø Ø2      :rem 167
4Ø52 DATA 3E 74 85 FB 2Ø 3E 74 85 77      :rem 221
4Ø53 DATA FC A5 8Ø DØ Ø6 A6 28 A4 97      :rem 237
4Ø54 DATA 29 DØ Ø8 C9 24 DØ Ø8 A6 94      :rem 2Ø8
4Ø55 DATA 86 A4 87 86 FB 84 FC A5 A9       :rem 21
4Ø56 DATA B3 C9 26 DØ 1A 38 A5 2A 6D      :rem 246
4Ø57 DATA E9 Ø2 85 FB A5 2B E9 ØØ DC       :rem 14
4Ø58 DATA 85 FC 2Ø F8 7E A2 8B AØ 1C       :rem 15
4Ø59 DATA ØB 2Ø F4 77 4C E3 71 2Ø AA      :rem 227
4Ø6Ø DATA F8 7E 2Ø 4D 74 2Ø 47 74 CE      :rem 227
4Ø61 DATA 2Ø F8 7E 2Ø CØ 77 A5 96 D8      :rem 229
4Ø62 DATA 29 BF FØ 1Ø A5 9D DØ Ø7 FF       :rem 1Ø
4Ø63 DATA AØ ØØ 84 9E AØ 6Ø 2C AØ 72      :rem 2ØØ
4Ø64 DATA 6E 4C 56 74 A9 ØD 2Ø D2 D4      :rem 238
4Ø65 DATA FF A5 8Ø DØ Ø8 A5 C9 85 11      :rem 24Ø
4Ø66 DATA 2A A5 CA 85 2B 2Ø ØE 74 15      :rem 224
4Ø67 DATA 2Ø Ø8 74 A5 B3 C9 5E FØ F5      :rem 242
4Ø68 DATA Ø3 4C D2 7E 2Ø 11 74 4C 7Ø      :rem 197
4Ø69 DATA 14 74 E6 D1 E6 D1 2Ø 76 74      :rem 215
4Ø7Ø DATA ØØ 2Ø D2 FF 2Ø 7Ø ØØ C9 B6      :rem 198
4Ø71 DATA ØØ FØ Ø7 C9 2C DØ F2 2Ø 32      :rem 199
4Ø72 DATA 1A 74 2Ø 76 ØØ 2Ø 9D 7B A4      :rem 192
4Ø73 DATA A9 ØD 2Ø D2 FF A9 6E 4C F6       :rem 34
4Ø74 DATA DA 77 AD B2 78 85 D4 A9 D6       :rem 29
4Ø75 DATA 6F 85 D3 2Ø 2F 74 A5 D3 FE        :rem 7
4Ø76 DATA 2Ø 32 74 AØ ØØ B1 77 FØ 82      :rem 175
4Ø77 DATA Ø6 2Ø 35 74 C8 DØ F6 2Ø 83      :rem 191
4Ø78 DATA 3B 74 2Ø 2C 7F AD B2 78 AF       :rem 16
4Ø79 DATA 85 D4 2Ø 2C 74 A9 6F 85 4A      :rem 237
4Ø8Ø DATA D3 2Ø 32 74 2Ø 3E 74 85 1Ø      :rem 162
4Ø81 DATA 5F 2Ø 3E 74 85 6Ø C9 3Ø F1      :rem 2Ø8
4Ø82 DATA DØ Ø4 C5 5F FØ Ø3 2Ø AD 48      :rem 223
4Ø83 DATA 72 A5 5F 2Ø 33 7F A5 6Ø B3      :rem 217
4Ø84 DATA 2Ø 33 7F C9 ØD FØ Ø5 2Ø 43      :rem 195
4Ø85 DATA 3E 74 DØ F4 2Ø 38 74 24 9A      :rem 212
4Ø86 DATA ØD 1Ø Ø3 2Ø 2C 7F 6Ø AØ 15      :rem 181
4Ø87 DATA ØØ B1 77 C9 4Ø DØ Ø3 C8 34      :rem 199
4Ø88 DATA B1 77 29 FE C9 3Ø FØ Ø3 C5      :rem 245
4Ø89 DATA 4C 1D 74 2Ø 47 77 A2 25 7E      :rem 215
```

```
4090 DATA A0 07 4C F4 77 20 B5 72 5B        :rem 215
4091 DATA A5 7C 10 03 4C DA 75 20 11        :rem 200
4092 DATA 59 74 A9 61 85 D3 20 70 41        :rem 181
4093 DATA 00 C9 00 F0 20 C9 2C D0 62        :rem 203
4094 DATA F5 20 70 00 C9 24 D0 D0 EE        :rem 227
4095 DATA 20 6C 7F 90 CB A5 84 85 EC          :rem 9
4096 DATA FB A5 85 85 FC A5 86 85 AA         :rem 33
4097 DATA C9 A5 87 85 CA A9 F3 8D 93         :rem 27
4098 DATA 0C 70 20 5C 74 4C 37 77 9A        :rem 215
4099 DATA 68 68 20 75 70 20 17 74 80        :rem 158
4100 DATA F0 31 A5 82 05 83 F0 2B 15        :rem 185
4101 DATA A5 11 18 65 82 85 60 A5 C1        :rem 179
4102 DATA 12 65 83 85 5F A2 90 38 B8        :rem 195
4103 DATA 20 20 74 20 26 74 A2 00 F0        :rem 143
4104 DATA BD 00 01 F0 06 9D 6F 02 3E        :rem 217
4105 DATA E8 D0 F5 A9 20 9D 6F 02 7C          :rem 1
4106 DATA E8 86 9E C9 3A F0 CB A6 90         :rem 25
4107 DATA 7C 30 03 4C 02 74 A2 FF EE        :rem 243
4108 DATA A0 00 E8 BD 00 02 C9 20 D0        :rem 207
4109 DATA F0 F8 D0 01 E8 BD 00 02 A0        :rem 231
4110 DATA C9 30 90 04 C9 3A 90 F4 EC        :rem 231
4111 DATA BD 00 02 C9 20 D0 01 E8 9F        :rem 216
4112 DATA BD 00 02 F0 07 99 00 02 AF        :rem 194
4113 DATA E8 C8 D0 F4 99 00 02 C8 29        :rem 227
4114 DATA C8 99 00 02 C8 C8 C8 4C F9        :rem 250
4115 DATA 05 74 A2 80 A9 00 F0 05 C7        :rem 191
4116 DATA AE F7 73 A9 FF 85 86 86 AF         :rem 41
4117 DATA 87 A5 C4 85 84 A5 C5 85 18        :rem 224
4118 DATA 85 24 86 30 16 A9 FF 20 C3        :rem 209
4119 DATA B0 7E A2 01 B5 84 48 B5 F9        :rem 240
4120 DATA 86 95 84 68 95 86 CA 10 04        :rem 188
4121 DATA F3 30 06 AD F8 73 20 B0 EF        :rem 240
4122 DATA 7E A2 00 A9 20 81 84 A9 69        :rem 204
4123 DATA 01 20 B2 7E A5 87 A4 86 59        :rem 205
4124 DATA C4 84 E5 85 B0 EB 68 68 E3        :rem 252
4125 DATA 4C FC 73 20 45 7F 90 08 C9        :rem 229
4126 DATA 85 B6 20 45 7F AA A5 B6 DC         :rem 11
4127 DATA 60 83 D8 4C 77 C3 4C 92 E1        :rem 232
4128 DATA C3 4C 99 C3 4C AE C3 4C 8C         :rem 35
4129 DATA B1 C3 4C 42 C4 4C 2C C5 FD         :rem 21
4130 DATA 4C 72 C5 4C A7 C5 4C C4 B5          :rem 6
4131 DATA C6 4C 73 C8 4C F8 CD 4C 56         :rem 22
4132 DATA 03 CE 4C 55 DB 4C D9 DC B2         :rem 37
4133 DATA 4C E9 DC 4C 2E E6 4C B6 8D         :rem 53
4134 DATA F0 4C BA F0 4C 28 F1 4C 69         :rem 10
4135 DATA 6F F1 4C 7F F1 4C 83 F1 24          :rem 4
4136 DATA 4C 8C F1 4C AE F2 4C 01 FE         :rem 41
4137 DATA F3 4C 55 F3 4C 0A F4 4C E3         :rem 13
4138 DATA 2E F4 4C 66 F4 4C 24 F5 D3          :rem 7
4139 DATA 4C 70 F5 4C 8D F6 4C BE 76         :rem 30
```

```
4140 DATA F6 4C E6 F8 4C 7B FC 4C D1         :rem 52
4141 DATA ED B3 4C Ø6 B4 4C ØD B4 4D         :rem 14
4142 DATA 4C 22 B4 4C 25 B4 4C B6 B7        :rem 242
4143 DATA B4 4C A3 B5 4C E9 B5 4C 72         :rem 1Ø
4144 DATA 22 B6 4C 4A B7 4C F6 B8 E1         :rem 11
4145 DATA 4C F5 BE 4C ØØ BF 4C 7F 2B         :rem 4Ø
4146 DATA CD 4C 83 CF 4C 93 CF 4C 9B         :rem 57
4147 DATA 55 E4 4C D2 FØ 4C D5 FØ A8         :rem 15
4148 DATA 4C 43 F1 4C 9E F1 4C AE AB         :rem 44
4149 DATA F1 4C B9 F1 4C CØ F1 4C DØ         :rem 24
415Ø DATA E2 F2 4C 35 F3 4C 8F F3 EA         :rem 31
4151 DATA 4C 49 F4 4C 6D F4 4C A5 D9         :rem 24
4152 DATA F4 4C 63 F5 4C AF F5 4C 2C         :rem 32
4153 DATA CC F6 4C FD F6 4C 2B F9 8F         :rem 78
4154 DATA 4C CØ FC A2 52 AØ 16 2Ø 2E        :rem 23Ø
4155 DATA F4 77 2Ø F2 74 4C E4 73 6C        :rem 233
4156 DATA A5 77 85 B5 A5 78 85 AD 5B          :rem Ø
4157 DATA 2Ø F2 74 A5 B5 85 77 A5 7F        :rem 235
4158 DATA AD 85 78 6Ø A2 18 BD 1Ø 6F        :rem 242
4159 DATA 7B 95 6F CA DØ F8 A9 F3 53         :rem 33
416Ø DATA 8D ØC 7Ø 4C CA 7Ø A2 41 8E        :rem 242
4161 DATA AØ 11 2Ø F4 77 A2 ØØ 86 9C        :rem 192
4162 DATA 8Ø A9 F2 A2 78 85 82 86 3E        :rem 224
4163 DATA 83 A6 8Ø BD B7 78 FØ 26 55        :rem 232
4164 DATA 48 29 7F 2Ø D2 FF E6 8Ø B9        :rem 255
4165 DATA 68 1Ø EE A9 2Ø 2Ø D2 FF EØ        :rem 248
4166 DATA 2Ø D2 FF AØ ØØ B1 82 2Ø 1C        :rem 211
4167 DATA D2 FF E6 82 DØ Ø2 E6 83 8C         :rem 1Ø
4168 DATA C9 ØD DØ EF FØ D3 2Ø 7A ØE         :rem 24
4169 DATA 77 2Ø 5Ø 75 4C 92 7F ØØ 47        :rem 185
417Ø DATA 1Ø 1Ø A9 1Ø 8D 4E 75 8D 4A        :rem 213
4171 DATA 4F 75 78 A2 D7 86 9Ø A2 93        :rem 224
4172 DATA 7Ø 86 91 58 6Ø 2Ø ØE 76 1D        :rem 179
4173 DATA 2Ø EC 77 85 81 A5 96 FØ 4C        :rem 233
4174 DATA Ø3 4C 37 77 A5 B3 C9 26 BC        :rem 24Ø
4175 DATA DØ ØA A2 8B AØ ØB 2Ø F4 3A        :rem 24Ø
4176 DATA 77 4C 85 75 2Ø 4D 74 A9 B9        :rem 228
4177 DATA Ø1 2Ø 3B 76 A5 81 4C 97 25        :rem 191
4178 DATA 75 A9 Ø1 2Ø 3B 76 2Ø CF 21        :rem 199
4179 DATA FF C9 ØD FØ 13 2Ø 41 76 51        :rem 223
418Ø DATA BØ Ø8 A5 D2 2Ø 41 74 4C BØ        :rem 2ØØ
4181 DATA F2 71 A5 96 DØ ØB FØ E6 B1        :rem 247
4182 DATA A9 ØØ 2Ø 41 76 A5 96 FØ 55        :rem 188
4183 DATA D8 98 2Ø 3E 76 2Ø CØ 77 65        :rem 2Ø4
4184 DATA A5 5C 85 2A A5 5D 85 2B 9E          :rem 7
4185 DATA A5 D2 2Ø 41 74 2Ø CC FF C9        :rem 25Ø
4186 DATA A5 B3 C9 5D FØ Ø3 4C EA 59         :rem 17
4187 DATA 7B 4C 13 72 2Ø 28 72 2Ø DA        :rem 199
4188 DATA 81 77 A6 D2 2Ø C9 FF 4C 5C         :rem 12
4189 DATA FØ 75 A9 ØD 2Ø D2 FF 2Ø D4        :rem 254
```

```
4190 DATA 5B 76 2Ø 6B 76 FØ Ø7 A2 95     :rem 209
4191 DATA ØØ 2Ø D7 7B BØ ØB A5 D2 5C     :rem 231
4192 DATA 2Ø 41 74 4C 13 72 2Ø D2 68     :rem 166
4193 DATA FF 2Ø 6B 76 FØ DC DØ F6 6E      :rem 38
4194 DATA 2Ø 7Ø ØØ C9 ØØ FØ Ø6 C9 E8     :rem 198
4195 DATA 2C DØ F5 FØ ØC A5 B3 C9 F2      :rem 25
4196 DATA 26 DØ Ø9 2Ø 73 76 4C 33 79     :rem 185
4197 DATA 76 4C 1D 74 2Ø 76 ØØ 2Ø F7     :rem 193
4198 DATA 17 74 2Ø ØB 74 A9 6E 2Ø 9F     :rem 216
4199 DATA DA 77 4C 4A 74 2Ø 3E 76 D1     :rem 247
42ØØ DATA 2Ø 41 76 38 A4 9D FØ 11 AF     :rem 2Ø8
42Ø1 DATA C9 ØØ FØ 11 C9 Ø1 FØ ØD 6F     :rem 216
42Ø2 DATA AØ ØØ D1 5C FØ Ø7 18 9Ø 94     :rem 188
42Ø3 DATA Ø4 AØ ØØ 91 5C E6 5C DØ 5D     :rem 218
42Ø4 DATA Ø2 E6 5D 6Ø A5 5C DØ Ø2 88     :rem 212
42Ø5 DATA C6 5D C6 5C 6Ø 2Ø 5B 76 6A     :rem 235
42Ø6 DATA AØ ØØ B1 5C 6Ø A5 2A 85 9F     :rem 221
42Ø7 DATA 5C A5 2B 85 5D 2Ø 62 76 FA     :rem 237
42Ø8 DATA 4C 62 76 2Ø 7Ø ØØ FØ Ø3 59     :rem 164
42Ø9 DATA 2Ø 1A 74 2Ø 9D 7B A9 ØD 64     :rem 216
421Ø DATA 2Ø D2 FF DØ ØB C8 98 18 BC        :rem 5
4211 DATA 65 5C 85 5C 9Ø Ø2 E6 5D 89     :rem 218
4212 DATA 2Ø 6B 76 FØ Ø6 2Ø D7 7B 97     :rem 2Ø4
4213 DATA 9Ø Ø1 6Ø 68 68 2Ø 7A 77 2E     :rem 175
4214 DATA 85 B3 4C 13 72 2Ø 44 74 1F     :rem 183
4215 DATA FØ ØC 2Ø E4 FF C9 2Ø DØ 48     :rem 244
4216 DATA Ø5 2Ø E4 FF FØ FB 6Ø DØ DD      :rem 19
4217 DATA Ø3 2Ø 2C 7F C9 24 FØ ØD 48     :rem 215
4218 DATA C9 23 DØ Ø3 4C 85 7F 2Ø D1     :rem 217
4219 DATA 5Ø 72 4C E4 73 A9 49 AØ Ø9     :rem 211
422Ø DATA ØØ 91 77 A5 D2 85 BB A5 9C     :rem 233
4221 DATA BØ 85 BA 2Ø 5Ø 72 2Ø 87 88     :rem 182
4222 DATA 77 A9 24 AØ ØØ 91 77 2Ø F4     :rem 183
4223 DATA 47 77 A9 6Ø 2Ø DE 77 AØ 24     :rem 2Ø6
4224 DATA Ø6 84 D1 2Ø AØ 77 C6 D1 D7     :rem 215
4225 DATA DØ F9 2Ø 92 77 A6 86 A5 3D     :rem 231
4226 DATA 87 2Ø 23 74 A9 2Ø 2Ø D1 Ø8     :rem 164
4227 DATA 77 2Ø AØ 77 C9 ØØ DØ ØC AD     :rem 224
4228 DATA 2Ø 92 77 A9 ØD 2Ø D1 77 B9     :rem 211
4229 DATA AØ Ø4 DØ D5 2Ø 92 77 A5 E9     :rem 22Ø
423Ø DATA 87 2Ø D1 77 2Ø B3 76 DØ F8     :rem 2Ø5
4231 DATA EØ A9 ØD 2Ø D2 FF 2Ø 7Ø E9     :rem 242
4232 DATA 72 A5 D2 2Ø 41 74 4C AF 47     :rem 213
4233 DATA 77 AØ FF C8 84 D1 B1 77 A5        :rem 4
4234 DATA 91 2A FØ Ø7 C9 2C DØ F3 96     :rem 238
4235 DATA C8 B1 77 85 8Ø A9 ØØ 85 DD     :rem 232
4236 DATA 96 AA A4 D1 BD 3C 78 91 49        :rem 6
4237 DATA 2A C8 E8 EØ Ø4 DØ F5 AD DØ       :rem 22
4238 DATA B2 78 85 D4 A5 2A 85 DA 4F        :rem 8
4239 DATA A5 2B 85 DB A9 ØØ 85 82 2Ø     :rem 22Ø
```

```
4240 DATA 85 83 6Ø 2Ø CC FF 2Ø 7Ø 1D          :rem 212
4241 DATA 72 18 A5 5F 65 6Ø C9 6Ø 84          :rem 199
4242 DATA FØ FØ DØ 1D 2Ø CC FF A6 A2           :rem 25
4243 DATA BA EØ Ø3 FØ E5 A6 BB 4C E1           :rem 26
4244 DATA C9 FF A5 87 85 86 A2 ØE 51          :rem 254
4245 DATA 2Ø EE 77 85 87 A4 96 FØ 45          :rem 226
4246 DATA D1 2Ø CØ 77 2Ø CC FF A5 48          :rem 248
4247 DATA D2 2Ø 41 74 2Ø 7Ø 72 4C ØB          :rem 174
4248 DATA D2 7E A2 38 AØ Ø5 2Ø F4 1D          :rem 227
4249 DATA 77 A5 96 2Ø ØA 7F A9 2Ø DC          :rem 243
425Ø DATA 4C D2 FF 2Ø D2 FF 2Ø CC Ø6           :rem 16
4251 DATA FF 4C 3B 74 E6 D1 E6 D1 98           :rem 26
4252 DATA 85 D3 A9 ØE 85 D2 AD B2 3B           :rem 11
4253 DATA 78 85 D4 4C 53 74 A6 D2 A4          :rem 232
4254 DATA 2Ø C6 FF 4C CF FF BD FF 45           :rem 87
4255 DATA 77 2Ø D2 FF E8 88 DØ F6 62            :rem 1
4256 DATA 6Ø 12 2Ø 49 4D 42 45 44 ØD          :rem 166
4257 DATA 44 45 44 2Ø 51 55 4F 54 CA          :rem 19Ø
4258 DATA 45 2Ø ØD ØD 12 2Ø 44 41 CA          :rem 182
4259 DATA 54 41 2Ø 3E 37 34 2Ø 43 3F          :rem 169
426Ø DATA 48 41 52 2Ø ØD ØD 53 41 57          :rem 161
4261 DATA 56 49 4E 47 2Ø 12 2Ø 44 36          :rem 155
4262 DATA 49 53 4B 2Ø 44 45 56 2Ø FA          :rem 186
4263 DATA 23 2Ø 53 54 3D 24 2C 53 36          :rem 164
4264 DATA 2C 57 93 12 2Ø 44 46 48 E6          :rem 181
4265 DATA 2Ø 45 44 49 54 4F 52 2Ø F9          :rem 182
4266 DATA 92 ØD ØØ 12 2Ø 44 46 48 5D          :rem 167
4267 DATA 2Ø 45 44 49 54 4F 52 2Ø F9          :rem 184
4268 DATA 4B 49 4C 4C 45 44 2Ø 92 99          :rem 2Ø7
4269 DATA ØD 2Ø 4F 4F 5Ø 53 21 2Ø 51          :rem 177
427Ø DATA 12 2Ø 54 45 58 54 2Ø 2Ø 49          :rem 133
4271 DATA 4D 4F 44 45 2Ø ØD 12 2Ø 7C          :rem 192
4272 DATA 42 41 53 49 43 2Ø 4D 4F E2          :rem 187
4273 DATA 44 45 2Ø ØD ØD 41 5Ø 5Ø 5C          :rem 172
4274 DATA 45 4E 44 49 4E 47 2Ø 2Ø ØB          :rem 188
4275 DATA 2A 3D 24 12 2Ø 43 41 4E 71          :rem 174
4276 DATA 27 54 2Ø 41 4C 54 45 52 ED          :rem 191
4277 DATA 2Ø 4E 45 58 54 2Ø 4C 49 EC          :rem 214
4278 DATA 4E 45 2Ø ØD FF FF 52 55 9B           :rem 11
4279 DATA CE BE AF A6 DD DE DF 3B 4A          :rem 13Ø
428Ø DATA 41 C4 3B 41 D5 3B 43 D3 59          :rem 216
4281 DATA 3B 44 C5 3B 45 C4 3B 45 F8          :rem 239
4282 DATA D5 3B 46 C3 3B 46 C9 3B 62          :rem 24Ø
4283 DATA 4D C2 3B 4D CB 3B 4D D4 42           :rem 21
4284 DATA 3B 51 D4 3B 52 CE 3B 55 B5          :rem 249
4285 DATA CE BB ØØ ØØ ØØ ØØ ØØ ØØ 77          :rem 167
4286 DATA ØØ ØØ ØØ ØØ 2Ø 2Ø 44 49 33          :rem 1Ø9
4287 DATA 53 4B 2Ø 43 4F 4D 4D 41 D5          :rem 222
4288 DATA 4E 44 53 ØD 2Ø 2Ø 4C 4F 33          :rem 2ØØ
4289 DATA 41 44 2Ø 5Ø 52 4F 47 52 D1          :rem 173
```

```
429Ø DATA 41 4D 53 2Ø 4F 52 2Ø 54 EA      :rem 194
4291 DATA 45 58 54 ØD 2Ø 2Ø 41 5Ø 31      :rem 143
4292 DATA 5Ø 45 4E 44 2Ø 5Ø 52 4F C8      :rem 189
4293 DATA 47 52 41 4D 53 2Ø 4F 52 C5      :rem 19Ø
4294 DATA 2Ø 54 45 58 54 ØD 2Ø 2Ø 4E      :rem 168
4295 DATA 56 45 52 49 46 59 2Ø 5Ø BB      :rem 185
4296 DATA 52 4F 47 52 41 4D 53 2Ø C5      :rem 193
4297 DATA 4F 52 2Ø 54 45 58 54 ØD ED      :rem 216
4298 DATA 2Ø 2Ø 4C 4F 41 44 2F 52 1F      :rem 2Ø1
4299 DATA 55 4E 2Ø 42 41 53 49 43 DB      :rem 196
43ØØ DATA 2Ø 5Ø 52 47 ØD 2Ø 2Ø 53 57      :rem 134
43Ø1 DATA 41 56 45 2Ø 5Ø 52 4F 47 CC      :rem 18Ø
43Ø2 DATA 52 41 4D 53 2Ø 4F 52 2Ø EC      :rem 188
43Ø3 DATA 54 45 58 54 ØD 41 44 44 E5      :rem 175
43Ø4 DATA 2Ø 54 52 41 49 4C 49 4E CD      :rem 2Ø5
43Ø5 DATA 47 2Ø 43 48 41 52 41 43 F7      :rem 155
43Ø6 DATA 54 45 52 ØD 41 55 54 4F CF      :rem 2Ø7
43Ø7 DATA 2Ø 4C 49 4E 45 2Ø 4E 55 F5      :rem 2Ø8
43Ø8 DATA 4D 42 45 52 ØD 43 48 41 Ø1      :rem 164
43Ø9 DATA 4E 47 45 2Ø 53 43 52 45 D9      :rem 181
431Ø DATA 45 4E 2Ø 43 41 53 45 ØD 24      :rem 157
4311 DATA 44 45 4C 45 54 45 2Ø 4C E1      :rem 181
4312 DATA 49 4E 45 53 ØD 45 52 41 EC      :rem 2Ø4
4313 DATA 53 45 2Ø 53 43 52 45 45 D6      :rem 154
4314 DATA 4E 2Ø 44 4F 57 4E ØD 45 Ø8      :rem 2Ø5
4315 DATA 52 41 53 45 2Ø 53 43 52 CD      :rem 163
4316 DATA 45 45 4E 2Ø 55 5Ø ØD 46 1Ø      :rem 163
4317 DATA 49 4E 44 2Ø 26 2Ø 43 48 34      :rem 157
4318 DATA 41 4E 47 45 2Ø 53 54 52 CC      :rem 188
4319 DATA 49 4E 47 53 ØD 46 49 4E E5      :rem 226
432Ø DATA 44 2Ø 53 54 52 49 4E 47 C5      :rem 174
4321 DATA 53 ØD 53 45 54 2Ø 42 41 11      :rem 137
4322 DATA 53 49 43 2Ø 4D 4F 44 45 DC      :rem 2Ø5
4323 DATA ØD 4B 49 4C 4C 2Ø 44 46 1D      :rem 212
4324 DATA 48 2Ø 45 44 49 54 4F 52 D1      :rem 178
4325 DATA ØD 53 45 54 2Ø 54 45 58 F6      :rem 179
4326 DATA 54 2Ø 45 44 49 54 4F 52 C5      :rem 18Ø
4327 DATA 2Ø 4D 4F 44 45 ØD 49 4E 17      :rem 2Ø9
4328 DATA 53 45 52 54 2Ø 4C 45 41 DØ      :rem 167
4329 DATA 44 49 4E 47 2Ø 51 55 4F C9      :rem 2Ø5
433Ø DATA 54 45 ØD 52 45 4E 55 4D D3      :rem 2Ø4
4331 DATA 42 45 52 2Ø 54 45 58 54 C2      :rem 154
4332 DATA 2Ø 4C 49 4E 45 53 ØD 55 Ø3      :rem 183
4333 DATA 4E 2D 4E 45 57 2Ø 28 43 1Ø      :rem 184
4334 DATA 41 4E 43 45 4C 2Ø 4E 45 EA      :rem 215
4335 DATA 57 2Ø 43 4D 44 29 ØD 2Ø 5F      :rem 186
4336 DATA 2Ø 44 49 53 5Ø 4C 41 59 CA      :rem 188
4337 DATA 2Ø 44 46 48 2Ø 45 44 49 1C      :rem 159
4338 DATA 54 4F 52 2Ø 4D 45 4E 55 B6      :rem 212
4339 DATA ØD 2D 55 31 2D 2D 52 45 4F      :rem 219
```

```
434Ø DATA 53 45 52 56 45 44 2D 2D DD      :rem 2Ø5
4341 DATA 2D 2D 2D ØD 2D 55 32 2D 8B      :rem 241
4342 DATA 2D 52 45 53 45 52 56 45 B7      :rem 178
4343 DATA 44 2D 2D 2D 2D 2D ØD 2D A1      :rem 25Ø
4344 DATA 55 33 2D 2D 52 45 53 45 EF      :rem 2Ø9
4345 DATA 52 56 45 44 2D 2D 2D 2D 1B      :rem 216
4346 DATA 2D ØD DC 7E C4 76 24 71 9D        :rem 7
4347 DATA 2D 71 2A 71 27 71 D2 72 EB      :rem 22Ø
4348 DATA 65 7C 28 7B 7B 7E 3A 7B CE       :rem 31
4349 DATA 9D 73 97 73 E5 7C F7 7C 12      :rem 253
435Ø DATA BB 7E DØ 74 C3 7E 4E 7C 78       :rem 25
4351 DATA E9 7B 86 7E Ø3 75 Ø2 7Ø AE      :rem 23Ø
4352 DATA Ø5 7Ø Ø8 7Ø 4C 55 7Ø 8Ø 82      :rem 157
4353 DATA 4C 7E 7Ø ØØ ØØ ØØ ØØ ØØ C6      :rem 156
4354 DATA ØØ ØØ ØØ E6 77 DØ Ø2 E6 EB      :rem 2Ø3
4355 DATA 78 AD ØØ Ø2 C9 3A BØ ØA 1C      :rem 232
4356 DATA C9 2Ø FØ EF 38 E9 3Ø 38 AF        :rem 9
4357 DATA E9 DØ 6Ø 2Ø 2F 7B 4C 84 4D      :rem 243
4358 DATA 7B 2Ø 17 74 A5 11 85 82 1D      :rem 192
4359 DATA A5 12 85 83 6Ø 2Ø 9B 7B AB      :rem 223
436Ø DATA A5 5C A6 5D 85 21 86 22 AE      :rem 237
4361 DATA 2Ø ØB 74 9Ø ØE AØ Ø1 B1 71      :rem 178
4362 DATA 5C FØ Ø8 AA 88 B1 5C 85 E8        :rem 6
4363 DATA 5C 86 5D A5 21 38 E5 5C 82      :rem 233
4364 DATA AA A5 22 E5 5D A8 BØ 1E D7       :rem 22
4365 DATA 8A 18 65 2A 85 2A 98 65 23      :rem 2Ø5
4366 DATA 2B 85 2B AØ ØØ B1 5C 91 E7      :rem 228
4367 DATA 21 C8 DØ F9 E6 5D E6 22 Ø3      :rem 245
4368 DATA A5 2B C5 22 BØ EF 2Ø Ø8 82      :rem 23Ø
4369 DATA 74 A5 1F A6 2Ø 18 69 Ø2 7F      :rem 216
437Ø DATA 85 2A 9Ø Ø1 E8 86 2B 2Ø Ø7      :rem 186
4371 DATA ØE 74 4C FC 73 FØ Ø8 9Ø 3B      :rem 239
4372 DATA Ø9 FØ Ø7 C9 2D FØ Ø3 4C CB      :rem 247
4373 DATA 1D 74 2Ø 17 74 A5 11 85 89      :rem 182
4374 DATA 4Ø A5 12 85 41 2Ø ØB 74 A4      :rem 175
4375 DATA 2Ø 76 ØØ FØ ØA C9 2D DØ AA      :rem 235
4376 DATA E6 2Ø 7Ø ØØ 2Ø 17 74 A5 3A      :rem 177
4377 DATA 11 Ø5 12 DØ Ø6 A9 FF 85 D5      :rem 223
4378 DATA 11 85 12 2Ø E1 7B 9Ø CF 7D      :rem 224
4379 DATA 6Ø 2Ø 6B 76 85 4Ø 2Ø 6B 4F      :rem 195
438Ø DATA 76 85 41 A5 11 C5 4Ø A5 64      :rem 188
4381 DATA 12 E5 41 6Ø 2Ø D8 7E A9 49      :rem 211
4382 DATA ØØ 85 9D 85 83 A9 ØA 85 9E      :rem 227
4383 DATA 82 A9 E8 85 DA A9 Ø3 85 5D        :rem 2
4384 DATA DB 2Ø 76 ØØ 9Ø Ø5 FØ 21 E9      :rem 199
4385 DATA 4C 1D 74 2Ø 2F 7B 2Ø 76 C3      :rem 221
4386 DATA ØØ FØ 16 2Ø 1A 74 2Ø 17 15      :rem 155
4387 DATA 74 A5 11 85 DA A5 12 85 3B      :rem 224
4388 DATA DB 2Ø 76 ØØ FØ Ø3 2Ø 1A 62      :rem 187
4389 DATA 74 2Ø 9D 7B 2Ø 9E 76 2Ø ØØ      :rem 196
```

```
4390 DATA 62 76 A5 DA 20 41 76 18 BA      :rem 218
4391 DATA 65 82 85 DA A5 DB 08 20 12      :rem 219
4392 DATA 41 76 28 65 83 85 DB 20 B9      :rem 206
4393 DATA 6B 76 D0 FB 20 5B 76 D0 93      :rem 244
4394 DATA DB 8D B3 78 C9 53 D0 03 7E       :rem 11
4395 DATA 20 70 00 20 D8 7E A9 22 2F      :rem 200
4396 DATA 85 47 C6 77 A9 40 D0 0A 34      :rem 217
4397 DATA 20 D8 7E 85 47 A9 80 8D 08      :rem 233
4398 DATA B3 78 85 46 20 81 76 B0 43      :rem 190
4399 DATA 03 20 93 76 C8 C0 01 D0 7B      :rem 205
4400 DATA 17 A9 00 85 81 24 46 50 80      :rem 151
4401 DATA 0F AD B3 78 C9 53 F0 EC 21        :rem 7
4402 DATA B1 5C C9 22 F0 E6 D0 21 41      :rem 220
4403 DATA B1 5C F0 0D C9 22 D0 DC 5F       :rem 16
4404 DATA C8 B1 5C F0 04 E6 81 D0 00      :rem 222
4405 DATA D3 84 B5 24 46 30 0A C0 90      :rem 193
4406 DATA 4C B0 2B A5 81 D0 27 F0 CC      :rem 253
4407 DATA C0 84 05 84 B5 E6 B5 88 5B      :rem 233
4408 DATA C8 B1 5C D0 FB C0 FB 90 15       :rem 22
4409 DATA 03 4C 4F 7D 20 2F 7E 20 F8      :rem 240
4410 DATA 63 7E A4 05 A5 47 91 5C 9D      :rem 226
4411 DATA E6 2A D0 02 E6 2B 20 AA 43      :rem 226
4412 DATA 7D B1 5C F0 94 C8 D0 F9 61        :rem 3
4413 DATA 20 D8 7E 85 05 A2 00 86 D8      :rem 207
4414 DATA 30 20 0F 7E A2 02 86 46 B3      :rem 192
4415 DATA D0 09 20 D8 7E 85 05 A2 85      :rem 209
4416 DATA 00 86 46 20 0F 7E 20 81 E6      :rem 187
4417 DATA 76 B0 03 20 93 76 84 52 D8      :rem 182
4418 DATA E6 52 A4 52 A6 2E A5 2F 2A      :rem 248
4419 DATA 85 87 B1 5C F0 ED DD 00 2D       :rem 16
4420 DATA 02 D0 ED E8 C8 C6 87 D0 74        :rem 2
4421 DATA F1 88 84 05 84 B5 A5 46 DA      :rem 229
4422 DATA F0 74 20 2F 7E A5 31 38 C1      :rem 214
4423 DATA E5 2F 85 86 F0 41 A0 00 10      :rem 193
4424 DATA C8 B1 5C D0 FB A5 86 10 25      :rem 246
4425 DATA 17 18 98 65 86 C9 02 B0 D3      :rem 203
4426 DATA 19 A2 9A A0 19 20 F4 77 67      :rem 211
4427 DATA A9 53 8D B3 78 4C A4 7D DF       :rem 30
4428 DATA 18 98 65 86 B0 EB C9 FC 05        :rem 2
4429 DATA B0 E7 A5 86 10 02 C6 87 DF      :rem 244
4430 DATA 18 65 05 85 B5 B0 05 20 6F      :rem 183
4431 DATA 63 7E F0 03 20 4B 7E A5 9E      :rem 236
4432 DATA B5 38 E5 31 A8 C8 A5 31 B7      :rem 238
4433 DATA F0 0F 85 B6 A6 30 BD 00 33      :rem 223
4434 DATA 02 91 5C E8 C8 C6 B6 D0 15      :rem 240
4435 DATA F5 18 A5 2A 65 86 85 2A 8A      :rem 233
4436 DATA A5 2B 65 87 85 2B 20 AA CA      :rem 248
4437 DATA 7D 4C 0C 7D A6 40 A5 41 E2      :rem 249
4438 DATA 20 23 74 A0 00 84 81 A9 FB      :rem 199
4439 DATA 20 20 D2 FF C8 B1 5C F0 2A        :rem 3
```

```
4440 DATA 14 C9 22 DØ F4 CØ Ø2 9Ø EB          :rem 222
4441 DATA FØ C8 B1 5C FØ Ø2 E6 81 E2          :rem 246
4442 DATA 88 A9 22 DØ E4 A9 ØD 2Ø 23          :rem 216
4443 DATA D2 FF CØ 4C BØ Ø6 A5 81 47          :rem 247
4444 DATA DØ Ø8 FØ 18 2Ø Ø2 7E 4C 34          :rem 195
4445 DATA EB 7D 2Ø Ø8 7E AD B3 78 1A           :rem 14
4446 DATA C9 53 DØ Ø8 A9 ØD 2Ø D2 64          :rem 22Ø
4447 DATA FF 4C 84 7B 2Ø B3 76 FØ 7D           :rem 1Ø
4448 DATA F3 A4 B5 6Ø A2 13 AØ 12 ED          :rem 237
4449 DATA DØ Ø4 A2 ØØ AØ 13 4C F4 97          :rem 2Ø8
445Ø DATA 77 A4 77 C8 94 2E A9 ØØ 3B          :rem 23Ø
4451 DATA 8D B3 78 95 2F B9 ØØ Ø2 C9          :rem 231
4452 DATA FØ ØC C5 Ø5 FØ Ø5 F6 2F 2Ø          :rem 224
4453 DATA C8 DØ F2 84 77 6Ø 4C A5 2A          :rem 241
4454 DATA 7B A5 5C 85 1F A5 5D 85 59            :rem Ø
4455 DATA 2Ø A5 2A 85 21 A5 2B 85 16          :rem 197
4456 DATA 22 6Ø A5 1F C5 21 DØ Ø4 ØØ          :rem 183
4457 DATA A5 2Ø C5 22 6Ø A4 Ø5 C8 83          :rem 2ØØ
4458 DATA B1 1F A4 B5 C8 91 1F 2Ø 3F          :rem 252
4459 DATA 4Ø 7E DØ Ø1 6Ø E6 1F DØ 3C          :rem 231
446Ø DATA EC E6 2Ø DØ E8 A4 Ø5 B1 FC           :rem 2Ø
4461 DATA 21 A4 B5 91 21 2Ø 4Ø 7E F6          :rem 195
4462 DATA DØ Ø1 6Ø A5 21 DØ Ø2 C6 71          :rem 181
4463 DATA 22 C6 21 4C 63 7E AD 4C D1          :rem 248
4464 DATA E8 49 Ø2 8D 4C E8 4C E4 DC           :rem 31
4465 DATA 73 AØ ØØ B1 28 DØ 11 C8 6B          :rem 2Ø6
4466 DATA B1 28 DØ ØC AØ Ø4 B1 28 CE          :rem 236
4467 DATA FØ Ø6 C8 DØ F9 4C F9 73 C1           :rem 13
4468 DATA 98 AØ ØØ 18 65 28 91 28 6A          :rem 187
4469 DATA A9 ØØ 65 29 C8 91 28 4C FC          :rem 24Ø
447Ø DATA 84 7B C6 85 18 65 84 85 3Ø          :rem 196
4471 DATA 84 9Ø Ø2 E6 85 6Ø A9 ØØ 76          :rem 182
4472 DATA A2 7D AØ ØE DØ Ø6 A9 8Ø 34          :rem 226
4473 DATA A2 6F AØ ØE 85 7C 2Ø F4 2C          :rem 249
4474 DATA 77 4C E4 73 A5 7C 3Ø EE A7           :rem 1Ø
4475 DATA 1Ø E4 A6 7C 1Ø 12 6Ø A5 C3          :rem 2Ø7
4476 DATA 7C 3Ø ØD 2Ø 7A 77 A2 FF 95          :rem 245
4477 DATA 86 B5 E8 86 77 4C 75 7Ø AF          :rem 255
4478 DATA A2 68 AØ Ø7 2Ø F4 77 4C 78          :rem 218
4479 DATA D2 7E A2 96 AØ Ø4 2Ø F4 CØ          :rem 234
448Ø DATA 77 A2 Ø1 B5 FA 48 B5 FB 3F           :rem 14
4481 DATA 2Ø ØA 7F 68 48 4A 4A 4A C9          :rem 24Ø
4482 DATA 4A 2Ø 22 7F AA 68 29 ØF AB          :rem 248
4483 DATA 2Ø 22 7F 48 8A 2Ø D2 FF 7C          :rem 243
4484 DATA 68 4C D2 FF 18 69 F6 9Ø 74          :rem 253
4485 DATA Ø2 69 Ø6 69 3A 6Ø A5 ØD DA          :rem 222
4486 DATA 49 FF 85 ØD 6Ø 24 ØD 3Ø 65          :rem 216
4487 DATA FB 4C D2 FF C9 3A Ø8 29 B4           :rem 43
4488 DATA ØF 28 9Ø Ø2 69 Ø8 6Ø A9 BD          :rem 218
4489 DATA ØØ 8D ØØ Ø1 2Ø 7Ø ØØ DØ 12          :rem 144
```

```
4490 DATA Ø6 2Ø CC FF 4C 1D 74 2Ø 12          :rem 225
4491 DATA 3A 7F ØA ØA ØA ØA 8D ØØ 92          :rem 232
4492 DATA Ø1 2Ø 7Ø ØØ FØ EB 2Ø 3A 3A          :rem 19Ø
4493 DATA 7F ØD ØØ Ø1 38 6Ø 2Ø E9 D2          :rem 199
4494 DATA 73 9Ø DE 85 85 86 84 E6 25          :rem 225
4495 DATA 77 DØ Ø2 E6 78 2Ø E9 73 DD          :rem 24Ø
4496 DATA 9Ø CF 85 87 86 86 6Ø 2Ø Ø9          :rem 2Ø4
4497 DATA 7Ø ØØ BØ C5 2Ø 17 74 A5 CB          :rem 211
4498 DATA 11 8D B2 78 A2 2C AØ ØC BE            :rem 7
4499 DATA 2Ø F4 77 AE B2 78 A9 ØØ F4          :rem 251
45ØØ DATA 2Ø 23 74 A9 2Ø 2Ø D2 FF 8F          :rem 211
45Ø1 DATA 2Ø D2 FF 4C D2 7E .. .. 73          :rem 211
45Ø2 DATA 2D BD 13 D9 1E 3E Ø9 AD QQ           :rem 64
```

DFH ED.6 GEN

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
5ØØ REM SAVE "@Ø:DFH ED.6 GEN",8              :rem 147
4ØØØ DATA ØØ 9Ø 4C ØD 9Ø 4C Ø2 94 A5          :rem 183
4ØØ1 DATA 4C ØD 9Ø 4C ØD 9Ø F3 2Ø 1B          :rem 213
4ØØ2 DATA 1A 94 A2 ØF BD 34 9A 95 81          :rem 23Ø
4ØØ3 DATA 7C 86 FF CA 1Ø F6 A5 D2 B8           :rem 28
4ØØ4 DATA 29 FC 8D F3 93 Ø9 Ø3 8D 2F          :rem 247
4ØØ5 DATA F4 93 A9 Ø8 8D E4 97 A2 1E          :rem 249
4ØØ6 DATA ØØ AØ 9Ø E4 37 98 E5 38 ØØ          :rem 183
4ØØ7 DATA BØ Ø4 86 37 84 38 4C 2C 5B          :rem 2Ø5
4ØØ8 DATA 94 8D ED 93 8E EE 93 A2 AE           :rem 48
4ØØ9 DATA F2 8E ØC 9Ø A6 7B EØ Ø2 E1          :rem 245
4Ø1Ø DATA DØ 11 A6 7A DØ ØD BA BD AB           :rem 24
4Ø11 DATA Ø2 Ø1 C9 A4 FØ 17 DØ Ø3 B6          :rem 193
4Ø12 DATA AC EF 93 AE EE 93 AD ED Ø9           :rem 79
4Ø13 DATA 93 C9 3A 9Ø Ø1 6Ø C9 2Ø 9Ø          :rem 181
4Ø14 DATA FØ 46 4C 54 9A 2Ø 61 9Ø 7F          :rem 2Ø4
4Ø15 DATA BØ Ø3 4C ØA 93 8C EF 93 56          :rem 234
4Ø16 DATA A2 ØØ 86 84 86 ØB A4 7A A5          :rem 213
4Ø17 DATA B9 ØØ Ø2 5D E6 97 FØ 13 68          :rem 2Ø7
4Ø18 DATA C9 8Ø FØ 13 E6 ØB E8 BD 1E           :rem 11
4Ø19 DATA E5 97 1Ø FA BD E6 97 DØ 7Ø            :rem 6
4Ø2Ø DATA E5 FØ BD E8 C8 DØ E1 84 89           :rem 21
4Ø21 DATA 7A A5 ØB ØA AA BD Ø3 9A C8           :rem 18
4Ø22 DATA 48 BD Ø2 9A 48 2Ø 5E 9Ø Ø9          :rem 2Ø2
4Ø23 DATA 4C 73 ØØ 78 A9 31 8D 14 4E          :rem 2Ø3
4Ø24 DATA Ø3 A9 EA 8D 15 Ø3 58 6Ø ØD          :rem 211
4Ø25 DATA BD Ø5 Ø1 C9 BE DØ ØA BD 1F           :rem 22
4Ø26 DATA Ø6 Ø1 C9 F8 DØ Ø3 2Ø 93 B2          :rem 192
4Ø27 DATA FC A5 C5 C9 FF FØ Ø5 CD 1Ø           :rem 41
4Ø28 DATA 75 94 FØ ØB 8D 75 94 A9 BD          :rem 254
4Ø29 DATA 1Ø 8D 76 94 4C 31 EA AD 45          :rem 239
```

```
4Ø3Ø DATA 15 Ø3 C9 EA FØ F6 AD 76 2C          :rem 5
4Ø31 DATA 94 FØ Ø5 CE 76 94 DØ EC E3          :rem Ø
4Ø32 DATA CE 77 94 DØ E7 A9 Ø4 8D 36        :rem 249
4Ø33 DATA 77 94 A9 ØØ 85 C5 FØ DC 36        :rem 228
4Ø34 DATA A9 2F 2C A9 5E 2C A9 5D C3         :rem 3Ø
4Ø35 DATA 2C A9 26 8D FØ 93 C9 5D CF         :rem 25
4Ø36 DATA DØ Ø3 A9 Ø1 2C A9 ØØ 85 29        :rem 192
4Ø37 DATA 93 85 ØA 2Ø 77 96 A4 B7 56        :rem 2Ø2
4Ø38 DATA DØ Ø3 4C Ø8 AF A5 7F 1Ø F6        :rem 246
4Ø39 DATA ØD AD FØ 93 C9 5E DØ Ø3 C9         :rem 14
4Ø4Ø DATA 4C 2C 9E 4C 8D 94 A9 6Ø 74        :rem 247
4Ø41 DATA 2Ø 1Ø 97 2Ø D2 96 2Ø CC C5        :rem 189
4Ø42 DATA FF A5 B8 2Ø C3 FF 2Ø 73 2F          :rem 8
4Ø43 DATA ØØ C9 2C DØ 1A AD FØ 93 F1        :rem 25Ø
4Ø44 DATA C9 5E FØ 1Ø C9 26 FØ ØC EE         :rem 1Ø
4Ø45 DATA 2Ø 73 ØØ C9 24 DØ ØC 2Ø 84        :rem 17Ø
4Ø46 DATA B9 9E BØ Ø7 4C Ø8 AF C9 26          :rem 4
4Ø47 DATA ØØ DØ DB A9 ØØ 85 B8 A9 C6        :rem 247
4Ø48 DATA 6Ø 85 B9 2Ø AF F5 2Ø D5 A9        :rem 241
4Ø49 DATA F3 A5 BA 2Ø B4 FF A5 B9 7D         :rem 45
4Ø5Ø DATA 2Ø 96 FF 2Ø A5 FF 85 AE 54          :rem Ø
4Ø51 DATA 2Ø A5 FF 85 AF A5 83 DØ 1Ø        :rem 242
4Ø52 DATA Ø6 A6 2B A4 2C DØ Ø8 C9 B8        :rem 242
4Ø53 DATA 24 DØ Ø8 A6 89 A4 8A 86 21        :rem 2Ø7
4Ø54 DATA AE 84 AF AD FØ 93 C9 26 ØØ         :rem 11
4Ø55 DATA DØ 1A 38 A5 2D E9 Ø2 85 9C        :rem 238
4Ø56 DATA AE A5 2E E9 ØØ 85 AF 2Ø 42        :rem 246
4Ø57 DATA 36 9E A2 8B AØ ØB 2Ø 26 ØE        :rem 224
4Ø58 DATA 97 4C D8 91 2Ø 36 9E 2Ø AØ        :rem 212
4Ø59 DATA D2 F5 2Ø F3 F4 2Ø 36 9E 3E        :rem 243
4Ø6Ø DATA 2Ø F2 96 A5 9Ø 29 BF FØ 4B        :rem 235
4Ø61 DATA ØC A5 93 DØ Ø3 A2 1D 2C FE        :rem 249
4Ø62 DATA A2 1C 4C 37 A4 A9 ØD 2Ø 45        :rem 22Ø
4Ø63 DATA D2 FF A5 83 DØ Ø8 A5 AE DC         :rem 41
4Ø64 DATA 85 2D A5 AF 85 2E 2Ø 59 CE          :rem 4
4Ø65 DATA A6 2Ø 33 A5 AD FØ 93 C9 69        :rem 24Ø
4Ø66 DATA 5E FØ Ø3 4C ØF 9E 2Ø 8E Ø8        :rem 239
4Ø67 DATA A6 4C AE A7 E6 B7 E6 B7 7F         :rem 6Ø
4Ø68 DATA A2 25 AØ Ø7 2Ø 26 97 2Ø 95        :rem 168
4Ø69 DATA 79 ØØ 2Ø D2 FF 2Ø 73 ØØ Ø3        :rem 176
4Ø7Ø DATA C9 ØØ FØ Ø7 C9 2C DØ F2 89        :rem 236
4Ø71 DATA 2Ø FD AE 2Ø 79 ØØ 2Ø CF AD        :rem 25Ø
4Ø72 DATA 9A A9 ØD 2Ø D2 FF A9 6E A8         :rem 33
4Ø73 DATA 4C ØC 97 AD E4 97 85 BA AA         :rem 34
4Ø74 DATA 2Ø B1 FF A9 6F 85 B9 2Ø BA         :rem 13
4Ø75 DATA 93 FF AØ ØØ B1 7A FØ Ø6 AD        :rem 255
4Ø76 DATA 2Ø A8 FF C8 DØ F6 2Ø AE DD         :rem 45
4Ø77 DATA FF 2Ø 6A 9E AD E4 97 85 2C         :rem 31
4Ø78 DATA BA 2Ø B4 FF A9 6F 85 B9 1D         :rem 39
4Ø79 DATA 2Ø 96 FF 2Ø A5 FF 85 62 AØ        :rem 245
```

```
4080 DATA 20 A5 FF 85 63 C9 30 D0 8B          :rem 237
4081 DATA 04 C5 62 F0 03 20 A4 92 8C          :rem 193
4082 DATA A5 62 20 71 9E A5 63 20 A2          :rem 194
4083 DATA 71 9E C9 0D F0 05 20 A5 61          :rem 217
4084 DATA FF D0 F4 20 AB FF 24 FF 50           :rem 44
4085 DATA 10 03 20 6A 9E 60 A0 00 C5          :rem 181
4086 DATA B1 7A C9 40 D0 03 C8 B1 80          :rem 228
4087 DATA 7A 29 FE C9 30 F0 03 4C 27          :rem 243
4088 DATA 08 AF 4C 77 96 20 AC 92 92          :rem 237
4089 DATA A5 7F 10 03 4C 06 95 A5 3D          :rem 222
4090 DATA 2B 85 87 A5 2C 85 88 A6 45          :rem 223
4091 DATA 2D A4 2E 86 89 84 8A AD 37          :rem 253
4092 DATA E4 97 85 BA 20 73 00 C9 EA          :rem 240
4093 DATA 00 F0 10 C9 2C D0 F5 20 26          :rem 203
4094 DATA 73 00 C9 24 D0 C9 20 AA 3D          :rem 226
4095 DATA 9E 90 C4 A9 F3 8D 0C 90 49            :rem 2
4096 DATA A9 87 A6 89 A4 8A 20 D8 7B            :rem 3
4097 DATA FF 4C 67 96 68 68 20 61 67          :rem 223
4098 DATA 90 20 6B A9 F0 31 A5 85 F1          :rem 224
4099 DATA 05 86 F0 2B A5 14 18 65 24          :rem 194
4100 DATA 85 85 63 A5 15 65 86 85 69          :rem 179
4101 DATA 62 A2 90 38 20 49 BC 20 EF          :rem 208
4102 DATA DD BD A2 00 BD 01 01 F0 15          :rem 230
4103 DATA 06 9D 77 02 E8 D0 F5 A9 8E          :rem 248
4104 DATA 20 9D 77 02 E8 86 C6 C9 CD          :rem 249
4105 DATA 3A F0 CB A6 7F 30 03 4C 67          :rem 241
4106 DATA 9F A4 A2 FF A0 00 E8 BD D7           :rem 39
4107 DATA 00 02 C9 20 F0 F8 D0 01 5C          :rem 199
4108 DATA E8 BD 00 02 C9 30 90 04 CC          :rem 222
4109 DATA C9 3A 90 F4 BD 00 02 C9 F1          :rem 246
4110 DATA 20 D0 01 E8 BD 00 02 F0 78          :rem 193
4111 DATA 07 99 00 02 E8 C8 D0 F4 EA          :rem 232
4112 DATA 99 00 02 C8 C8 99 00 02 3A          :rem 180
4113 DATA C8 C8 C8 4C A2 A4 AE F3 75           :rem 30
4114 DATA 93 A9 00 F0 05 AE F4 93 9A          :rem 235
4115 DATA A9 FF 85 89 86 8A A5 D1 C4           :rem 18
4116 DATA 85 87 A5 D2 85 88 24 89 C3          :rem 216
4117 DATA 30 16 A9 FF 20 ED 9D A2 C6           :rem 11
4118 DATA 01 B5 87 48 B5 89 95 87 21          :rem 195
4119 DATA 68 95 89 CA 10 F3 30 05 78          :rem 203
4120 DATA A9 D8 20 ED 9D A2 00 A9 8A            :rem 5
4121 DATA 20 81 87 A9 01 20 EF 9D 82          :rem 203
4122 DATA A5 8A A4 89 C4 87 E5 88 EC           :rem 16
4123 DATA B0 EB 68 68 4C 83 A4 20 02          :rem 212
4124 DATA 83 9E 90 0A 8D EF 93 20 16          :rem 228
4125 DATA 83 9E AA AD EF 93 60 41 65            :rem 3
4126 DATA 00 9E FF 03 07 04 07 A2 AC          :rem 221
4127 DATA 52 A0 16 20 26 97 20 1A E1          :rem 171
4128 DATA 94 4C D8 93 A5 7A 8D ED 1C           :rem 28
4129 DATA 93 A5 7B 8D EE 93 20 1A 05          :rem 240
```

```
413Ø DATA 94 AD ED 93 85 7A AD EE A5        :rem 51
4131 DATA 93 85 7B 6Ø A2 18 BD 42 54        :rem 2Ø4
4132 DATA 9A 95 72 CA DØ F8 A9 F3 31        :rem 2
4133 DATA 8D ØC 9Ø 4C B9 9Ø A2 41 5F        :rem 235
4134 DATA AØ 11 2Ø 26 97 A2 ØØ 86 4A        :rem 169
4135 DATA 83 A9 24 A2 98 85 85 86 E6        :rem 217
4136 DATA 86 A6 83 BD E9 97 FØ 26 FE        :rem 21
4137 DATA 48 29 7F 2Ø D2 FF E6 83 B6        :rem 255
4138 DATA 68 1Ø EE A9 2Ø 2Ø D2 FF EØ        :rem 248
4139 DATA 2Ø D2 FF AØ ØØ B1 85 2Ø 19        :rem 2Ø4
414Ø DATA D2 FF E6 85 DØ Ø2 E6 86 86        :rem 25Ø
4141 DATA C9 ØD DØ EF FØ D3 2Ø AA DE        :rem 45
4142 DATA 96 2Ø 78 94 4C DØ 9E 4Ø 44        :rem 199
4143 DATA Ø5 Ø2 A9 1Ø 8D 76 94 8D 1C        :rem 2Ø6
4144 DATA 77 94 78 A2 C6 8E 14 Ø3 7Ø        :rem 2Ø1
4145 DATA A2 9Ø 8E 15 Ø3 58 6Ø 2Ø 5Ø        :rem 164
4146 DATA 3A 95 2Ø 1E 97 85 84 A5 AE        :rem 232
4147 DATA 9Ø FØ Ø3 4C 67 96 AD FØ 97        :rem 234
4148 DATA 93 C9 26 DØ ØA A2 8B AØ D7        :rem 249
4149 DATA ØB 2Ø 26 97 4C BØ 94 2Ø 68        :rem 19Ø
415Ø DATA D2 F5 A9 Ø1 2Ø 68 95 A5 CD        :rem 235
4151 DATA 84 4C C2 94 A9 Ø1 2Ø 68 A8        :rem 2Ø6
4152 DATA 95 2Ø CF FF C9 ØD FØ 13 A4        :rem 1Ø
4153 DATA 2Ø 6E 95 BØ Ø8 A5 B8 2Ø A8        :rem 215
4154 DATA C3 FF 4C E7 91 A5 9Ø DØ 75        :rem 6
4155 DATA ØB FØ E6 A9 ØØ 2Ø 6E 95 53        :rem 217
4156 DATA A5 9Ø FØ D8 98 2Ø 6B 95 4B        :rem 234
4157 DATA 2Ø F2 96 A5 5F 85 2D A5 FD        :rem 8
4158 DATA 6Ø 85 2E A5 B8 2Ø C3 FF AE        :rem 16
4159 DATA 2Ø CC FF AD FØ 93 C9 5D BF        :rem 69
416Ø DATA FØ Ø3 4C 1C 9B 4C Ø4 92 28        :rem 212
4161 DATA 2Ø 1A 92 2Ø B1 96 A6 B8 6F        :rem 214
4162 DATA 2Ø C9 FF 4C 1C 95 A9 ØD 65        :rem 3
4163 DATA 2Ø D2 FF 2Ø 88 95 2Ø 98 1A        :rem 2Ø9
4164 DATA 95 FØ Ø7 A2 ØØ 2Ø Ø9 9B ØE        :rem 194
4165 DATA BØ ØB A5 B8 2Ø C3 FF 4C BA        :rem 28
4166 DATA Ø4 92 2Ø D2 FF 2Ø 98 95 2C        :rem 212
4167 DATA FØ DC DØ F6 2Ø 73 ØØ C9 12        :rem 228
4168 DATA ØØ FØ Ø6 C9 2C DØ F5 FØ 6Ø        :rem 229
4169 DATA ØD AD FØ 93 C9 26 DØ Ø9 FB        :rem 18
417Ø DATA 2Ø AØ 95 4C 6Ø 95 4C Ø8 16        :rem 184
4171 DATA AF 2Ø 79 ØØ 2Ø 6B A9 2Ø 64        :rem 192
4172 DATA 13 A6 A9 6E 2Ø ØC 97 4C 21        :rem 215
4173 DATA AF F5 2Ø 6B 95 2Ø 6E 95 19        :rem 232
4174 DATA 38 A4 93 FØ 11 C9 ØØ FØ D7        :rem 219
4175 DATA 11 C9 Ø1 FØ ØD AØ ØØ D1 B7        :rem 211
4176 DATA 5F FØ Ø7 18 9Ø Ø4 AØ ØØ 5E        :rem 197
4177 DATA 91 5F E6 5F DØ Ø2 E6 6Ø B3        :rem 244
4178 DATA 6Ø A5 5F DØ Ø2 C6 6Ø C6 DE        :rem 252
4179 DATA 5F 6Ø 2Ø 88 95 AØ ØØ B1 B3        :rem 2Ø1
```

```
4180 DATA 5F 6Ø A5 2D 85 5F A5 2E B8          :rem 3
4181 DATA 85 6Ø 2Ø 8F 95 4C 8F 95 67          :rem 217
4182 DATA 2Ø 73 ØØ FØ Ø3 2Ø FD AE AF          :rem 231
4183 DATA 2Ø CF 9A A9 ØD 2Ø D2 FF DØ          :rem 21
4184 DATA DØ ØB C8 98 18 65 5F 85 64          :rem 227
4185 DATA 5F 9Ø Ø2 E6 6Ø 2Ø 98 95 7C          :rem 2Ø6
4186 DATA FØ Ø6 2Ø Ø9 9B 9Ø Ø1 6Ø 55          :rem 169
4187 DATA 68 68 2Ø AA 96 8D FØ 93 CØ          :rem 238
4188 DATA 4C Ø4 92 2Ø ED F6 FØ ØC 1F          :rem 252
4189 DATA 2Ø E4 FF C9 2Ø DØ Ø5 2Ø 1F          :rem 231
419Ø DATA E4 FF FØ FB 6Ø DØ Ø3 2Ø DF          :rem 2Ø
4191 DATA 6A 9E C9 24 FØ ØD C9 23 22          :rem 239
4192 DATA DØ Ø3 4C C3 9E 2Ø 49 92 85          :rem 211
4193 DATA 4C D8 93 A9 49 AØ ØØ 91 26          :rem 212
4194 DATA 7A A5 B8 8D F2 93 A5 9A D8          :rem 32
4195 DATA 8D F1 93 2Ø 49 92 2Ø B7 1D          :rem 214
4196 DATA 96 A9 24 AØ ØØ 91 7A 2Ø D2          :rem 2ØØ
4197 DATA 77 96 A9 6Ø 2Ø 1Ø 97 AØ 83          :rem 187
4198 DATA Ø6 84 B7 2Ø D2 96 C6 B7 BA          :rem 247
4199 DATA DØ F9 2Ø C2 96 A6 89 A5 EB          :rem 15
42ØØ DATA 8A 2Ø CD BD A9 2Ø 2Ø Ø3 EØ          :rem 222
42Ø1 DATA 97 2Ø D2 96 C9 ØØ DØ ØC 3C          :rem 2Ø9
42Ø2 DATA 2Ø C2 96 A9 ØD 2Ø Ø3 97 18          :rem 18Ø
42Ø3 DATA AØ Ø4 DØ D5 2Ø C2 96 A5 9A          :rem 219
42Ø4 DATA 8A 2Ø Ø3 97 2Ø E1 95 DØ 56          :rem 183
42Ø5 DATA EØ A9 ØD 2Ø D2 FF 2Ø 67 F2          :rem 243
42Ø6 DATA 92 A5 B8 2Ø C3 FF 4C E1 Ø2          :rem 244
42Ø7 DATA 96 AØ FF C8 84 B7 B1 7A 9D          :rem 27
42Ø8 DATA 91 2D FØ Ø7 C9 2C DØ F3 93          :rem 239
42Ø9 DATA C8 B1 7A 85 83 A9 ØØ 85 D7          :rem 233
421Ø DATA 9Ø AA A4 B7 BD 6E 97 91 18          :rem 254
4211 DATA 2D C8 E8 EØ Ø4 DØ F5 AD CD          :rem 36
4212 DATA E4 97 85 BA A5 2D 85 BB 34          :rem Ø
4213 DATA A5 2E 85 BC A9 ØØ 85 85 39          :rem 227
4214 DATA 85 86 6Ø 2Ø CC FF 2Ø 67 23          :rem 2Ø6
4215 DATA 92 18 A5 62 65 63 C9 6Ø 5E          :rem 2ØØ
4216 DATA FØ FØ DØ 1F 2Ø CC FF AE 98          :rem 41
4217 DATA F1 93 EØ Ø3 FØ E4 AE F2 25          :rem 247
4218 DATA 93 4C C9 FF A5 8A 85 89 1C          :rem 21
4219 DATA A2 ØE 2Ø 2Ø 97 85 8A A4 C6          :rem 218
422Ø DATA 9Ø FØ CF 2Ø F2 96 2Ø CC 1D          :rem 24Ø
4221 DATA FF A5 B8 2Ø C3 FF 2Ø 67 3B          :rem 7
4222 DATA 92 4C ØF 9E A2 38 AØ Ø5 F6          :rem 234
4223 DATA 2Ø 26 97 A5 9Ø 2Ø 48 9E E8          :rem 199
4224 DATA A9 2Ø 4C D2 FF 2Ø D2 FF 29          :rem 1Ø
4225 DATA 2Ø CC FF 4C AE FF E6 B7 7F          :rem 85
4226 DATA E6 B7 85 B9 A9 ØE 85 B8 31          :rem 254
4227 DATA AD E4 97 85 BA 4C CØ FF 8E          :rem 58
4228 DATA A6 B8 2Ø C6 FF 4C CF FF A3          :rem 66
4229 DATA BD 31 97 2Ø D2 FF E8 88 1A          :rem 8
```

```
4230 DATA DØ F6 6Ø 12 2Ø 49 4D 42 DØ      :rem 189
4231 DATA 45 44 44 45 44 2Ø 51 55 E4      :rem 153
4232 DATA 4F 54 45 2Ø ØD ØD 12 2Ø AC      :rem 196
4233 DATA 44 41 54 41 2Ø 3E 37 34 1D      :rem 161
4234 DATA 2Ø 43 48 41 52 2Ø ØD ØD 88      :rem 162
4235 DATA 53 41 56 49 4E 47 2Ø 12 Ø6      :rem 156
4236 DATA 2Ø 44 49 53 4B 2Ø 44 45 ØC      :rem 164
4237 DATA 56 2Ø 23 2Ø 53 54 3D 24 3F      :rem 165
4238 DATA 2C 53 2C 57 93 12 2Ø 44 F5      :rem 189
4239 DATA 46 48 2Ø 45 44 49 54 4F DD      :rem 2Ø5
424Ø DATA 52 2Ø 92 ØD ØØ 12 2Ø 44 79      :rem 137
4241 DATA 46 48 2Ø 45 44 49 54 4F DD      :rem 198
4242 DATA 52 2Ø 4B 49 4C 4C 45 44 D9      :rem 2Ø6
4243 DATA 2Ø 92 ØD 2Ø 4F 4F 5Ø 53 EØ      :rem 192
4244 DATA 21 2Ø 12 2Ø 54 45 58 54 48      :rem 134
4245 DATA 2Ø 2Ø 4D 4F 44 45 2Ø ØD 6E      :rem 193
4246 DATA 12 2Ø 42 41 53 49 43 2Ø 4C      :rem 143
4247 DATA 4D 4F 44 45 2Ø ØD ØD 41 6Ø      :rem 195
4248 DATA 5Ø 5Ø 45 4E 44 49 4E 47 AB      :rem 212
4249 DATA 2Ø 2Ø 2A 3D 24 12 2Ø 43 CØ      :rem 16Ø
425Ø DATA 41 4E 27 54 2Ø 41 4C 54 F5      :rem 183
4251 DATA 45 52 2Ø 4E 45 58 54 2Ø EA      :rem 184
4252 DATA 4C 49 4E 45 2Ø ØD Ø8 FF A4      :rem 236
4253 DATA 52 55 CE BE AF A6 DD DE BD       :rem 93
4254 DATA DF 3B 41 C4 3B 41 D5 3B 55      :rem 246
4255 DATA 43 D3 3B 44 C5 3B 45 C4 62      :rem 218
4256 DATA 3B 45 D5 3B 46 C3 3B 46 E6      :rem 241
4257 DATA C9 3B 4D C2 3B 4D CB 3B 5F       :rem 44
4258 DATA 4D D4 3B 51 D4 3B 52 CE 24      :rem 25Ø
4259 DATA 3B 55 CE BB ØØ ØØ ØØ ØØ E7      :rem 213
426Ø DATA ØØ ØØ ØØ ØØ ØØ ØØ 2Ø 2Ø CØ       :rem 93
4261 DATA 44 49 53 4B 2Ø 43 4F 4D D6      :rem 2Ø7
4262 DATA 4D 41 4E 44 53 ØD 2Ø 2Ø 4Ø      :rem 17Ø
4263 DATA 4C 4F 41 44 2Ø 5Ø 52 4F CF      :rem 216
4264 DATA 47 52 41 4D 53 2Ø 4F 52 C5      :rem 188
4265 DATA 2Ø 54 45 58 54 ØD 2Ø 2Ø 4E      :rem 166
4266 DATA 41 5Ø 5Ø 45 4E 44 2Ø 5Ø D8      :rem 168
4267 DATA 52 4F 47 52 41 4D 53 2Ø C5      :rem 191
4268 DATA 4F 52 2Ø 54 45 58 54 ØD ED      :rem 214
4269 DATA 2Ø 2Ø 56 45 52 49 46 59 EB      :rem 186
427Ø DATA 2Ø 5Ø 52 4F 47 52 41 4D C8      :rem 185
4271 DATA 53 2Ø 4F 52 2Ø 54 45 58 DB      :rem 186
4272 DATA 54 ØD 2Ø 2Ø 4C 4F 41 44 3F      :rem 193
4273 DATA 2F 52 55 4E 2Ø 42 41 53 E6      :rem 188
4274 DATA 49 43 2Ø 5Ø 52 47 ØD 2Ø 3E      :rem 166
4275 DATA 2Ø 53 41 56 45 2Ø 5Ø 52 EF      :rem 168
4276 DATA 4F 47 52 41 4D 53 2Ø 4F C8      :rem 213
4277 DATA 52 2Ø 54 45 58 54 ØD 41 FB      :rem 192
4278 DATA 44 44 2Ø 54 52 41 49 4C DC      :rem 193
4279 DATA 49 4E 47 2Ø 43 48 41 52 E4      :rem 187
```

```
4280 DATA 41 43 54 45 52 ØD 41 55 EE       :rem 186
4281 DATA 54 4F 2Ø 4C 49 4E 45 2Ø F5       :rem 2Ø9
4282 DATA 4E 55 4D 42 45 52 ØD 43 E7       :rem 21Ø
4283 DATA 48 41 4E 47 45 2Ø 53 43 E7       :rem 182
4284 DATA 52 45 45 4E 2Ø 43 41 53 DF       :rem 19Ø
4285 DATA 45 ØD 44 45 4C 45 54 45 FB       :rem 213
4286 DATA 2Ø 4C 49 4E 45 53 ØD 45 13       :rem 191
4287 DATA 52 41 53 45 2Ø 53 43 52 CD       :rem 171
4288 DATA 45 45 4E 2Ø 44 4F 57 4E DØ       :rem 216
4289 DATA ØD 45 52 41 53 45 2Ø 53 1Ø       :rem 15Ø
429Ø DATA 43 52 45 45 4E 2Ø 55 5Ø CE       :rem 187
4291 DATA ØD 46 49 4E 44 2Ø 26 2Ø 6C       :rem 187
4292 DATA 43 48 41 4E 47 45 2Ø 53 E7       :rem 182
4293 DATA 54 52 49 4E 47 53 ØD 46 D6       :rem 2Ø5
4294 DATA 49 4E 44 2Ø 53 54 52 49 C3       :rem 184
4295 DATA 4E 47 53 ØD 53 45 54 2Ø FF       :rem 214
4296 DATA 42 41 53 49 43 2Ø 4D 4F E2       :rem 193
4297 DATA 44 45 ØD 4B 49 4C 4C 2Ø 1E       :rem 222
4298 DATA 44 46 48 2Ø 45 44 49 54 E8       :rem 181
4299 DATA 4F 52 ØD 53 45 54 2Ø 54 F2       :rem 196
43ØØ DATA 45 58 54 2Ø 45 44 49 54 C9       :rem 165
43Ø1 DATA 4F 52 2Ø 4D 4F 44 45 ØD ØD       :rem 2Ø8
43Ø2 DATA 49 4E 53 45 52 54 2Ø 4C BF       :rem 2Ø3
43Ø3 DATA 45 41 44 49 4E 47 2Ø 51 E7       :rem 175
43Ø4 DATA 55 4F 54 45 ØD 52 45 4E D1       :rem 2Ø5
43Ø5 DATA 55 4D 42 45 52 2Ø 54 45 CC       :rem 184
43Ø6 DATA 58 54 2Ø 4C 49 4E 45 53 B9       :rem 2ØØ
43Ø7 DATA ØD 55 4E 2D 4E 45 57 2Ø 19       :rem 2Ø7
43Ø8 DATA 28 43 41 4E 43 45 4C 2Ø 12       :rem 164
43Ø9 DATA 4E 45 57 2Ø 43 4D 44 29 F9       :rem 2Ø3
431Ø DATA ØD 2Ø 2Ø 44 49 53 5Ø 4C 37       :rem 157
4311 DATA 41 59 2Ø 44 46 48 2Ø 45 ØF       :rem 151
4312 DATA 44 49 54 4F 52 2Ø 4D 45 CC       :rem 2Ø4
4313 DATA 4E 55 ØD 2D 55 31 2D 2D 43       :rem 211
4314 DATA 52 45 53 45 52 56 45 44 AØ       :rem 155
4315 DATA 2D 2D 2D 2D 2D ØD 2D 55 9Ø       :rem 242
4316 DATA 32 2D 2D 52 45 53 45 52 F3       :rem 186
4317 DATA 56 45 44 2D 2D 2D 2D 2D 4Ø       :rem 215
4318 DATA ØD 2D 55 33 2D 2D 52 45 4D       :rem 216
4319 DATA 53 45 52 56 45 44 2D 2D DD       :rem 211
432Ø DATA 2D 2D 2D ØD 19 9E F2 95 2E       :rem 254
4321 DATA ØD 91 16 91 13 91 1Ø 91 76       :rem 153
4322 DATA C2 92 97 9B 5A 9A B8 9D 31       :rem 251
4323 DATA 6C 9A 92 93 8B 93 1A 9C Ø1       :rem 229
4324 DATA 2C 9C F8 9D F4 93 ØØ 9E 7E        :rem 19
4325 DATA 8Ø 9B 1B 9B C3 9D 2B 94 1Ø       :rem 238
4326 DATA Ø2 9Ø Ø5 9Ø Ø8 9Ø 4C 3F B6       :rem 187
4327 DATA 9Ø 8Ø 4C 6C 9Ø ØØ ØØ ØØ A8       :rem 173
4328 DATA ØØ ØØ ØØ ØØ ØØ E6 7A DØ DØ       :rem 166
4329 DATA Ø2 E6 7B AD ØØ Ø2 C9 3A EB          :rem Ø
```

```
4330 DATA BØ ØA C9 2Ø FØ EF 38 E9 5D        :rem 8
4331 DATA 3Ø 38 E9 DØ 6Ø 2Ø 61 9A 64      :rem 184
4332 DATA 4C B6 9A 2Ø 6B A9 A5 14 77      :rem 236
4333 DATA 85 85 A5 15 85 86 6Ø 2Ø B1      :rem 179
4334 DATA CD 9A A5 5F A6 6Ø 85 24 E6        :rem 5
4335 DATA 86 25 2Ø 13 A6 9Ø ØE AØ 3E      :rem 194
4336 DATA Ø1 B1 5F FØ Ø8 AA 88 B1 14      :rem 225
4337 DATA 5F 85 5F 86 6Ø A5 24 38 D6      :rem 227
4338 DATA E5 5F AA A5 25 E5 6Ø A8 5B       :rem 16
4339 DATA BØ 1E 8A 18 65 2D 85 2D 4C      :rem 242
434Ø DATA 98 65 2E 85 2E AØ ØØ B1 D1      :rem 213
4341 DATA 5F 91 24 C8 DØ F9 E6 6Ø 15      :rem 23Ø
4342 DATA E6 25 A5 2E C5 25 BØ EF 99        :rem 4
4343 DATA 2Ø 33 A5 A5 22 A6 23 18 6Ø      :rem 171
4344 DATA 69 Ø2 85 2D 9Ø Ø1 E8 86 E4      :rem 2Ø3
4345 DATA 2E 2Ø 59 A6 4C 83 A4 FØ 5Ø      :rem 218
4346 DATA Ø8 9Ø Ø9 FØ Ø7 C9 2D FØ 82      :rem 212
4347 DATA Ø3 4C Ø8 AF 2Ø 6B A9 A5 21      :rem 226
4348 DATA 14 85 43 A5 15 85 44 2Ø 81      :rem 162
4349 DATA 13 A6 2Ø 79 ØØ FØ ØA C9 EB      :rem 229
435Ø DATA 2D DØ E6 2Ø 73 ØØ 2Ø 6B FF      :rem 221
4351 DATA A9 A5 14 Ø5 15 DØ Ø6 A9 Ø5      :rem 192
4352 DATA FF 85 14 85 15 2Ø 13 9B ØØ      :rem 186
4353 DATA 9Ø CF 6Ø 2Ø 98 95 85 43 2C      :rem 2Ø3
4354 DATA 2Ø 98 95 85 44 A5 14 C5 6C      :rem 2Ø4
4355 DATA 43 A5 15 E5 44 6Ø 2Ø 15 45      :rem 167
4356 DATA 9E A9 ØØ 85 93 85 86 A9 ED      :rem 251
4357 DATA ØA 85 85 A9 E8 85 BB A9 72        :rem 3
4358 DATA Ø3 85 BC 2Ø 79 ØØ 9Ø Ø5 8E      :rem 192
4359 DATA FØ 21 4C Ø8 AF 2Ø 61 9A D1      :rem 23Ø
436Ø DATA 2Ø 79 ØØ FØ 16 2Ø FD AE 96      :rem 215
4361 DATA 2Ø 6B A9 A5 14 85 BB A5 2E      :rem 245
4362 DATA 15 85 BC 2Ø 79 ØØ FØ Ø3 1E      :rem 194
4363 DATA 2Ø FD AE 2Ø CF 9A 2Ø CB C1       :rem 28
4364 DATA 95 2Ø 8F 95 A5 BB 2Ø 6E 39      :rem 234
4365 DATA 95 18 65 85 85 BB A5 BC C8        :rem 2
4366 DATA Ø8 2Ø 6E 95 28 65 86 85 3D      :rem 199
4367 DATA BC 2Ø 98 95 DØ FB 2Ø 88 84      :rem 238
4368 DATA 95 DØ DB 8D E5 97 C9 53 9B       :rem 28
4369 DATA DØ Ø3 2Ø 73 ØØ 2Ø 15 9E C7      :rem 179
437Ø DATA A9 22 85 4A C6 7A A9 4Ø 3D      :rem 238
4371 DATA DØ ØA 2Ø 15 9E 85 4A A9 DB      :rem 246
4372 DATA 8Ø 8D E5 97 85 49 2Ø AE DB        :rem Ø
4373 DATA 95 BØ Ø3 2Ø CØ 95 C8 CØ BB      :rem 227
4374 DATA Ø1 DØ 17 A9 ØØ 85 84 24 42      :rem 168
4375 DATA 49 5Ø ØF AD E5 97 C9 53 13      :rem 236
4376 DATA FØ EC B1 5F C9 22 FØ E6 53       :rem 19
4377 DATA DØ 22 B1 5F FØ ØD C9 22 16      :rem 23Ø
4378 DATA DØ DC C8 B1 5F FØ Ø4 E6 A2       :rem 28
4379 DATA 84 DØ D3 8C ED 93 24 49 6Ø      :rem 241
```

```
4380 DATA 3Ø ØA CØ 4C BØ 2D A5 84 B4        :rem 231
4381 DATA DØ 29 FØ BF 84 ØB 8C ED 5Ø         :rem 14
4382 DATA 93 EE ED 93 88 C8 B1 5F 9F         :rem 46
4383 DATA DØ FB CØ FB 9Ø Ø3 4C 85 16        :rem 25Ø
4384 DATA 9C 2Ø 6A 9D 2Ø 9F 9D A4 3D          :rem 9
4385 DATA ØB A5 4A 91 5F E6 2D DØ 33        :rem 251
4386 DATA Ø2 E6 2E 2Ø E4 9C B1 5F 3A        :rem 252
4387 DATA FØ 91 C8 DØ F9 2Ø 15 9E 1B        :rem 247
4388 DATA 85 ØB A2 ØØ 86 33 2Ø 4A AB        :rem 2Ø9
4389 DATA 9D A2 Ø2 86 49 DØ Ø9 2Ø F7        :rem 219
4390 DATA 15 9E 85 ØB A2 ØØ 86 49 4C        :rem 21Ø
4391 DATA 2Ø 4A 9D 2Ø AE 95 BØ Ø3 E3        :rem 226
4392 DATA 2Ø CØ 95 84 55 E6 55 A4 D3        :rem 214
4393 DATA 55 A6 31 A5 32 85 8A B1 3D        :rem 221
4394 DATA 5F FØ ED DD ØØ Ø2 DØ ED 28         :rem 25
4395 DATA E8 C8 C6 8A DØ F1 88 84 33          :rem 6
4396 DATA ØB 8C ED 93 A5 49 FØ 78 93          :rem 6
4397 DATA 2Ø 6A 9D A5 34 38 E5 32 B1        :rem 225
4398 DATA 85 89 FØ 42 AØ ØØ C8 B1 A7        :rem 227
4399 DATA 5F DØ FB A5 89 1Ø 17 18 69        :rem 242
44ØØ DATA 98 65 89 C9 Ø2 BØ 19 A2 44        :rem 196
44Ø1 DATA 9A AØ 19 2Ø 26 97 A9 53 D4        :rem 2Ø4
44Ø2 DATA 8D E5 97 4C DE 9C 18 98 81          :rem 9
44Ø3 DATA 65 89 BØ EB C9 FC BØ E7 1B         :rem 32
44Ø4 DATA A5 89 1Ø Ø2 C6 8A 18 65 F3        :rem 2Ø7
44Ø5 DATA ØB 8D ED 93 BØ Ø5 2Ø 9F 74        :rem 237
44Ø6 DATA 9D FØ Ø3 2Ø 86 9D AD ED 93          :rem 5
44Ø7 DATA 93 38 E5 34 A8 C8 A5 34 D3        :rem 233
44Ø8 DATA FØ 11 8D EF 93 A6 33 BD 5A         :rem 14
44Ø9 DATA ØØ Ø2 91 5F E8 C8 CE EF A1         :rem 15
441Ø DATA 93 DØ F4 18 A5 2D 65 89 D1        :rem 227
4411 DATA 85 2D A5 2E 65 8A 85 2E D9        :rem 249
4412 DATA 2Ø E4 9C 4C 41 9C A6 43 4E        :rem 235
4413 DATA A5 44 2Ø CD BD AØ ØØ 84 49        :rem 221
4414 DATA 84 A9 2Ø 2Ø D2 FF C8 B1 49        :rem 238
4415 DATA 5F FØ 14 C9 22 DØ F4 CØ 2E        :rem 246
4416 DATA Ø2 9Ø FØ C8 B1 5F FØ Ø2 B4        :rem 225
4417 DATA E6 84 88 A9 22 DØ E4 A9 E6          :rem 1
4418 DATA ØD 2Ø D2 FF CØ 4C BØ Ø6 4Ø        :rem 233
4419 DATA A5 84 DØ Ø8 FØ 18 2Ø 3D 9A        :rem 22Ø
442Ø DATA 9D 4C 25 9D 2Ø 43 9D AD A8          :rem Ø
4421 DATA E5 97 C9 53 DØ Ø8 A9 ØD DA          :rem 2
4422 DATA 2Ø D2 FF 4C B6 9A 2Ø E1 72        :rem 244
4423 DATA 95 FØ F3 AC ED 93 6Ø A2 5A         :rem 12
4424 DATA 13 AØ 12 DØ Ø4 A2 ØØ AØ 25        :rem 163
4425 DATA 13 4C 26 97 A4 7A C8 94 6A        :rem 232
4426 DATA 31 A9 ØØ 8D E5 97 95 32 56        :rem 2Ø4
4427 DATA B9 ØØ Ø2 FØ ØC C5 ØB FØ 89        :rem 226
4428 DATA Ø5 F6 32 C8 DØ F2 84 7A 4B        :rem 243
4429 DATA 6Ø 4C D7 9A A5 5F 85 22 38        :rem 236
```

```
4430 DATA A5 6Ø 85 23 A5 2D 85 24 D8          :rem 206
4431 DATA A5 2E 85 25 6Ø A5 22 C5 97          :rem 2Ø7
4432 DATA 24 DØ Ø4 A5 23 C5 25 6Ø F6          :rem 193
4433 DATA A4 ØB C8 B1 22 AC ED 93 8A           :rem 19
4434 DATA C8 91 22 2Ø 7B 9D DØ Ø1 7C          :rem 217
4435 DATA 6Ø E6 22 DØ EB E6 23 DØ Ø4          :rem 226
4436 DATA E7 A4 ØB B1 24 AC ED 93 69           :rem 15
4437 DATA 91 24 2Ø 7B 9D DØ Ø1 6Ø E2          :rem 198
4438 DATA A5 24 DØ Ø2 C6 25 C6 24 9Ø          :rem 199
4439 DATA 4C 9F 9D AD 18 DØ 49 Ø2 98            :rem 3
444Ø DATA 8D 18 DØ 4C D8 93 AØ ØØ 34          :rem 214
4441 DATA B1 2B DØ 11 C8 B1 2B DØ CF            :rem 3
4442 DATA ØC AØ Ø4 B1 2B FØ Ø6 C8 B6          :rem 23Ø
4443 DATA DØ F9 4C 62 A4 98 AØ ØØ AD          :rem 247
4444 DATA 18 65 2B 91 2B A9 ØØ 65 8E          :rem 21Ø
4445 DATA 2C C8 91 2B 4C B6 9A C6 EE           :rem 37
4446 DATA 88 18 65 87 85 87 9Ø Ø2 D6          :rem 192
4447 DATA E6 88 6Ø A9 ØØ A2 7D AØ CA          :rem 251
4448 DATA ØE DØ Ø6 A9 8Ø A2 6F AØ 42          :rem 229
4449 DATA ØE 85 7F 2Ø 26 97 4C D8 ED            :rem 4
445Ø DATA 93 A5 7F 3Ø EE 1Ø E4 A6 91          :rem 238
4451 DATA 7F 1Ø 13 6Ø A5 7F 3Ø ØE 9C          :rem 215
4452 DATA 2Ø AA 96 A2 FF 8E ED 93 F1           :rem 36
4453 DATA E8 86 7A 4C 61 9Ø A2 68 D1          :rem 234
4454 DATA AØ Ø7 2Ø 26 97 4C ØF 9E 83          :rem 211
4455 DATA A2 96 AØ Ø4 2Ø 26 97 A2 A5          :rem 198
4456 DATA Ø1 B5 AD 48 B5 AE 2Ø 48 8A          :rem 25Ø
4457 DATA 9E 68 48 4A 4A 4A 4A 2Ø 6A          :rem 243
4458 DATA 6Ø 9E AA 68 29 ØF 2Ø 6Ø 38          :rem 215
4459 DATA 9E 48 8A 2Ø D2 FF 68 4C EB           :rem 35
446Ø DATA D2 FF 18 69 F6 9Ø Ø2 69 BD          :rem 254
4461 DATA Ø6 69 3A 6Ø A5 FF 49 FF ØB            :rem 5
4462 DATA 85 FF 6Ø 24 FF 3Ø FB 4C 82            :rem 7
4463 DATA D2 FF C9 3A Ø8 29 ØF 28 C4            :rem 7
4464 DATA 9Ø Ø2 69 Ø8 6Ø A9 ØØ 8D 67          :rem 183
4465 DATA ØØ Ø1 2Ø 73 ØØ DØ Ø6 2Ø 76          :rem 131
4466 DATA CC FF 4C Ø8 AF 2Ø 78 9E FC           :rem 62
4467 DATA ØA ØA ØA ØA 8D ØØ Ø1 2Ø 2A          :rem 197
4468 DATA 73 ØØ FØ EB 2Ø 78 9E ØD 6F          :rem 246
4469 DATA ØØ Ø1 38 6Ø 2Ø DD 93 9Ø 47          :rem 173
447Ø DATA DE 85 88 86 87 E6 7A DØ D8           :rem 15
4471 DATA Ø2 E6 7B 2Ø DD 93 9Ø CF AE           :rem 14
4472 DATA 85 8A 86 89 6Ø 2Ø 73 ØØ EF          :rem 2Ø5
4473 DATA BØ C5 2Ø 6B A9 A5 14 8D 11          :rem 227
4474 DATA E4 97 A2 2C AØ ØC 2Ø 26 C5          :rem 228
4475 DATA 97 AE E4 97 A9 ØØ 2Ø CD AA           :rem 18
4476 DATA BD A9 2Ø 2Ø D2 FF 2Ø D2 97          :rem 253
4477 DATA FF 4C ØF 9E .. .. .. .. Ø8          :rem 191
4478 DATA 9C BØ B8 39 28 7C 37 65 QQ           :rem 32
```

BASIC Programs

This section contains listings for the six BASIC programs in the DFH family. Please refer to Chapter 9 for complete instructions before entering these programs. Each of these programs must be saved to disk with the filenames shown in the first line of each program.

In order to allow these programs to work on the PET, it was necessary to indicate the drive number when one DFH program loads and runs another. For this reason, PET owners should place the program disk in drive 0. PET owners who usually don't need a drive 0 for LOADs and who may want to use either drive 1 or drive 0 should remove the **"0:"+** from the OPEN statements on the following lines:

DFH BOOT	**1870**
DFH SORT	**4350**
DFH MERGE	**3160**
DFH SWAP	**3660**
DFH SPLIT	**4640**

and change line 6030 in DFH PRINT to read:

6030 : DR$="0": PS$="DFH BOOT":OPEN 8,8,8,PS$+",P,R

DFH BOOT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:DFH BOOT",8                    :rem 44
1Ø1Ø :                                          :rem 252
1Ø2Ø REM" A BOOTSTRAP LOADER, MEMORY CONDITIONER A
     ND MASTER MENU FOR"                        :rem 159
1Ø3Ø REM" DATA FILE HANDLER (DFH) PROGRAMS."
                                                :rem 224
1Ø4Ø :                                          :rem 255
1Ø7Ø :                                            :rem 2
1Ø8Ø IF PEEK (824)<>249 OR LF<>249 THEN 118Ø: REM"
     --NOT PET EDITOR REQ.--"                   :rem 151
1Ø9Ø :                                            :rem 4
11ØØ REM"--- RESTORE PET & ACTIVATE EDITOR AT $7ØØ
     Ø ---"                                     :rem 195
111Ø :                                          :rem 253
112Ø POKE 52,Ø: POKE 53,112: CLR                 :rem 98
113Ø KB$="SYS28672"+CHR$(13): FOR JJ=1 TO LEN(KB$)
                                                :rem 135
114Ø POKE 622+JJ,ASC(MID$(KB$,JJ,1)): NEXT JJ: POK
     E 158,LEN(KB$)                             :rem 224
```

```
115Ø IF PEEK (5859Ø)=2Ø8 THEN PRINT CHR$(14): REM"
     --8Ø COL SCREEN EXPAND--"                  :rem 253
116Ø NEW                                         :rem 178
117Ø :                                            :rem 3
118Ø : IF PEEK (824)<>25Ø OR LF<>25Ø THEN 13ØØ: RE
     M"--NOT C64 EDITOR REQ.--"                 :rem 128
119Ø :                                            :rem 5
12ØØ REM"--- RESTORE C64 & ACTIVATE EDITOR AT $9ØØ
     Ø ---"                                     :rem 138
121Ø :                                          :rem 254
122Ø KB$="SYS36864"+CHR$(13): FOR JJ=1 TO LEN(KB$)
                                                :rem 137
123Ø POKE 63Ø+JJ,ASC(MID$(KB$,JJ,1)): NEXT JJ: POK
     E 198, LEN(KB$)                            :rem 227
124Ø : POKE 55,Ø: POKE 56,144: POKE 56578, PEEK (5
     6578) OR 3                                 :rem 14Ø
125Ø POKE 56576, (PEEK (56576) AND 252) OR 3
                                                 :rem 68
126Ø POKE 53272,21: POKE 648,4                  :rem 245
127Ø POKE 5328Ø,254: POKE 53281,246: PRINT CHR$(15
     4);                                        :rem 228
128Ø POKE 43,1: POKE 44,8: POKE 2Ø48,Ø: PRINT "
     {CLR}": NEW                                 :rem 72
129Ø :                                            :rem 6
13ØØ : IF PEEK (65534)=72 THEN 143Ø: REM"-- PET OR
      C-64 ?"                                     :rem 6
131Ø :                                          :rem 255
132Ø REM"--- SETUP FOR 'PET' COMPUTERS ---"
                                                :rem 1Ø9
133Ø :                                            :rem 1
134Ø IF PEEK (28684)=242 THEN SYS 28675: REM"-- DE
     ACTIVATE EDITOR --"                        :rem 159
135Ø POKE 52,Ø: POKE 53,121: REM"-- TOP OF MEMORY
     {SPACE}= $79ØØ"                            :rem 154
136Ø IF PEEK (5859Ø)=2Ø8 THEN PRINT CHR$(142): REM
     "8Ø COL SCREEN CONDENSE"                    :rem 13
137Ø IF PEEK (824)<>248 THEN 154Ø: REM"-- NOT A PR
     OG'D LOAD"                                  :rem 48
138Ø POKE 42,PEEK (2Ø1): POKE 43,PEEK (2Ø2): REM"-
     - END OF BASIC"                             :rem 38
139Ø GOTO 154Ø                                  :rem 2Ø8
14ØØ :                                          :rem 255
141Ø REM"--- SETUP FOR 'C-64' COMPUTERS ---"
                                                 :rem 94
142Ø :                                            :rem 1
143Ø : POKE 55,Ø: POKE 56,121: REM"-- TOP OF MEMOR
     Y = $79ØØ"                                 :rem 217
144Ø POKE 56578, PEEK (56578) OR 3: REM"-- I/O CON
     T. TO OUTPUT"                                :rem 5
```

```
145Ø POKE 56576, (PEEK (56576) AND 252) OR 1: REM"
     -- SCREEN BANK = $8ØØØ"                    :rem 5
146Ø POKE 53272,5: REM"-- OFFSET = $ØØØØ"    :rem 2
147Ø POKE 648,128: REM"-- SCREEN EDITOR = $8ØØØ"
                                                :rem 2Ø8
148Ø POKE 5328Ø,14: POKE 53281,14: PRINT CHR$(31);
     : REM"-- COLOR CONT."                     :rem 225
149Ø IF PEEK (824)<>248 THEN 154Ø: REM"-- NOT A PR
     OG'D LOAD"                                 :rem 51
15ØØ POKE 45,PEEK (174): POKE 46,PEEK (175): REM"-
     - END OF BASIC"                            :rem 56
151Ø :                                          :rem 1
152Ø REM"--- INITIALIZE & ARE M.L. SUBROUTINES LOA
     DED ? ---"                                 :rem 46
153Ø :                                          :rem 3
154Ø : POKE 824,Ø: CLR : OPEN 15,8,15: TY=2
                                                :rem 1Ø2
155Ø IF PEEK (65534)=72 THEN TY=6: YØ$=CHR$(31): Y
     1$=CHR$(158)                               :rem 187
156Ø :                                          :rem 6
157Ø DIM PG$(1Ø): CR$=CHR$(13): CU$=CR$+"{UP}": TA
     =18                                        :rem 91
158Ø PG$(Ø)="DFH SUBS$79": PG$(1)="DFH SORT": PG$(
     2)=PG$(1)                                  :rem 5Ø
159Ø PG$(3)=PG$(1): PG$(4)="DFH MERGE": PG$(5)="DF
     H PRINT"                                   :rem 32
16ØØ PG$(6)="DFH SPLIT": PG$(7)=PG$(6): PG$(8)="DF
     H SWAP"                                    :rem 24Ø
161Ø PG$(9)="DFH ED.PET$7Ø": PG$(1Ø)="DFH ED.C64$9
     Ø"                                         :rem 118
162Ø IF PEEK (3Ø977)=21 AND PEEK (3Ø98Ø)=3Ø THEN 2
     Ø5Ø: REM"ML SUBS LOADED"                   :rem 6Ø
163Ø :                                          :rem 4
164Ø RD$="{RVS}{39 SPACES}{OFF}": PRINT "{CLR}";RD
     $                                          :rem 61
165Ø PRINT "{RVS}{3 SPACES}D A T A{3 SPACES}F I L
     {SPACE}E{3 SPACES}H A N D L E R{3 SPACES}
     {OFF}";CR$;RD$                             :rem 98
166Ø PRINT "{RVS} Ø7-15-84{2 SPACES}BY ---- BLAINE
      D. STANDAGE, {OFF}"                       :rem 18Ø
167Ø PRINT "{RVS} JOHN L. DARLING & KENNETH D. STA
     NDAGE {OFF}";CR$;RD$                       :rem 194
168Ø PRINT "{RVS} A FAMILY OF COORDINATED PROGRAMS
      FOR{2 SPACES}{OFF}"                       :rem 9Ø
169Ø PRINT "{RVS} PREPARATION AND COMPLETE PROCESS
     ING{3 SPACES}{OFF}"                        :rem 179
17ØØ PRINT "{RVS} OF SEQUENTIAL DATA FILES CONTAIN
     ING{3 SPACES}{OFF}"                        :rem 68
171Ø PRINT "{RVS} EITHER SINGLE-FIELD OR MULTI-FIE
     LD{4 SPACES}{OFF}"                         :rem 15
```

```
172Ø PRINT "{RVS} DATA RECORDS.{25 SPACES}{OFF}";C
     R$;RD$                                :rem 13Ø
173Ø PRINT "{RVS} MAXIMUM DATA CAPACITY:
     {16 SPACES}{OFF}"                       :rem Ø
174Ø PRINT "{RVS}{3 SPACES}5Ø{2 SPACES}FILES ON UP
      TO 5Ø DISKS{9 SPACES}{OFF}"          :rem 225
175Ø PRINT "{RVS}{3 SPACES}65Ø RECORDS PER FILE (*
     ){12 SPACES}{OFF}"                    :rem 113
176Ø PRINT "{RVS}{3 SPACES}2Ø{2 SPACES}FIELDS PER
     {SPACE}RECORD{15 SPACES}{OFF}"          :rem 2
177Ø PRINT "{RVS}{3 SPACES}74{2 SPACES}CHARACTERS
     {SPACE}PER RECORD{11 SPACES}{OFF}"     :rem 53
178Ø PRINT "{RVS}{3 SPACES}(*)=SOME EXCEPTIONS ALL
     OWED.{8 SPACES}{OFF}";CR$;RD$          :rem 82
179Ø PRINT "{RVS} ESSENTIAL OPERATOR INSTRUCTIONS
     {SPACE}ARE{3 SPACES}{OFF}"            :rem 231
18ØØ PRINT "{RVS} PRESENTED DURING PROGRAM OPERATI
     ON.{3 SPACES}{OFF}";CR$;RD$;CR$       :rem 133
181Ø GOSUB 193Ø: PS$=PG$(Ø)                :rem 145
182Ø : GOSUB 187Ø: IF EN<>Ø THEN 182Ø       :rem 85
183Ø PRINT "{RVS} LOADING ";PS$;" {OFF}": LOAD PS$
     ,8,1                                  :rem 162
184Ø :                                       :rem 7
185Ø REM"--SUB--- TEST FOR NEEDED PROGRAM FILE ---
     "                                      :rem 76
186Ø :                                       :rem 9
187Ø : OPEN 8,8,8,"Ø:"+PS$+",P,R": GOSUB 2ØØØ: CLO
     SE 8: IF EN=Ø THEN RETURN             :rem 173
188Ø PRINT "{DOWN}INSTALL A DISK CONTAINING:"
                                            :rem 9Ø
189Ø PRINT "{RVS} ";PS$;" {OFF} -- THEN --": GOSUB
      193Ø: RETURN                          :rem 76
19ØØ :                                       :rem 4
191Ø REM"--SUB--- WAIT FOR OPERATOR ---"   :rem 2Ø5
192Ø :                                       :rem 6
193Ø : PRINT "PRESS ANY KEY TO CONTINUE"    :rem 62
194Ø : GET KB$: IF KB$<>"" THEN 194Ø       :rem 2Ø6
195Ø : GET KB$: IF KB$="" THEN 195Ø        :rem 147
196Ø RETURN                                :rem 176
197Ø :                                      :rem 11
198Ø REM"--SUB--- DISK ERROR TEST ---"      :rem 65
199Ø :                                      :rem 13
2ØØØ : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø THEN RETURN
                                            :rem 23
2Ø1Ø PRINT Y1$;"{DOWN}{RVS} DISK ERROR {OFF}"YØ$:
     {SPACE}PRINT EN;EM$;ET;ES: RETURN     :rem 16Ø
2Ø2Ø :                                     :rem 254
2Ø3Ø REM"--- FUNCTION SELECT MENU ---"      :rem 86
2Ø4Ø :                                       :rem Ø
2Ø5Ø : K1$="1"                              :rem 83
```

```
2Ø6Ø : PRINT "{CLR}{RVS}{4 SPACES}DATA{2 SPACES}FI
     LE{2 SPACES}HANDLER{2 SPACES}FUNCTIONS
     {4 SPACES}{OFF}"                         :rem 251
2Ø7Ø PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CREATE OR
      EDIT A DATA FILE"                       :rem 118
2Ø8Ø PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}LIST (HAR
     D COPY FOR EDITING)"                      :rem 84
2Ø9Ø PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}SORT BY R
     ECORD OR FIELD CONTENT"                   :rem 7Ø
21ØØ PRINT "{DOWN}{RVS} 4 {OFF}{2 SPACES}MERGE SOR
     TED FILES                                 :rem 15
211Ø PRINT "{DOWN}{RVS} 5 {OFF}{2 SPACES}PRINT PER
      USER DEFINED FORMAT"                    :rem 234
212Ø PRINT "{DOWN}{RVS} 6 {OFF}{2 SPACES}SPLIT FIL
     ES BY FIELD CONTENT"                     :rem 154
213Ø PRINT "{DOWN}{RVS} 7 {OFF}{2 SPACES}EXTRACT R
     ECORDS BY FIELD CONTENT"                 :rem 2Ø2
214Ø PRINT "{DOWN}{RVS} 8 {OFF}{2 SPACES}RE-STRUCT
     URE DATA RECORDS"                         :rem 7Ø
215Ø PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}ACTIVATE
     {SPACE}DFH EDITOR &{RIGHT}DOS"           :rem 154
216Ø PRINT "{DOWN}{RVS} 1Ø {OFF}{2 SPACES}QUIT
     {DOWN}"                                    :rem 4
217Ø PRINT "YOUR CHOICE ---";TAB(TA+2);K1$;"
     {2 SPACES}";CU$;TAB(TA);                 :rem 245
218Ø INPUT K1$: SE=VAL(K1$): IF SE<1 OR SE>1Ø THEN
      2Ø6Ø                                    :rem 123
219Ø IF SE=1Ø THEN 242Ø                       :rem 136
22ØØ :                                        :rem 254
221Ø LF=248: IF SE<>9 THEN 225Ø                :rem 62
222Ø IF TY=2 THEN LF=249: GOTO 225Ø            :rem 74
223Ø LF=25Ø: SE=1Ø                            :rem 157
224Ø :                                          :rem 2
225Ø : PS$=PG$(SE): GOSUB 187Ø: IF EN=Ø THEN 232Ø
                                              :rem 244
226Ø IF TY=6 THEN DR$="Ø": GOTO 229Ø           :rem 83
227Ø : PRINT "WHICH DRIVE";TAB(TA+2);"Ø";CU$;TAB(T
     A);                                      :rem 241
228Ø INPUT DR$: DR$=LEFT$(DR$,1): IF DR$<"Ø" OR DR
     $>"1" THEN 227Ø                           :rem 57
229Ø : PRINT# 15,"I";DR$;CR$;: GOSUB 2ØØØ :rem 118
23ØØ IF EN<>Ø THEN PRINT "CAN'T INITIALIZE, TRY AG
     AIN!": GOTO 227Ø                         :rem 186
231Ø GOTO 2Ø6Ø                                :rem 199
232Ø : CLOSE 15: POKE 824,LF: PRINT "{RVS} LOADING
      ";PS$;" {OFF}"                          :rem 2Ø8
233Ø IF TY=2 THEN 236Ø                        :rem 1Ø9
234Ø IF SE>8 THEN 237Ø                         :rem 97
235Ø POKE 43,1: POKE 44,4: POKE 1Ø24,Ø: REM"-- STA
     RT OF BASIC = $Ø4Ø1"                     :rem 225
```

```
2360 : LOAD PS$,8: REM"--LOAD DFH PROGRAM--"
                                             :rem 112
2370 : POKE 30977,195: POKE 30980,195: REM"--VOID
     {SPACE}SUBS--"                          :rem 199
2380 LOAD PS$,8,1: REM"--LOAD DFH EDITOR--":rem 68
2390 :                                         :rem 8
2400 REM"--- PROGRAM EXIT ROUTINES ---"    :rem 199
2410 :                                         :rem 1
2420 : CLOSE 15: POKE 30977,195: POKE 30980,195: R
     EM"--VOID SUBS--"                       :rem 217
2430 IF TY=6 THEN 1240                       :rem 110
2440 POKE 52,0: POKE 53,128                   :rem 84
2450 IF PEEK (58590)=208 THEN PRINT CHR$(14): REM"
     --80 COL SCREEN EXPAND--"                 :rem 1
2460 NEW                                     :rem 182
```

DFH SORT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1000 REM SAVE "@0:DFH SORT",8                 :rem 64
1010 :                                       :rem 252
1020 REM" A DATA FILE HANDLER PROGRAM FOR DATA ENT
     RY, EDITING,"                           :rem 107
1030 REM" SORTING AND LISTING MULTI-FIELD SEQUENTI
     AL DATA FILES."                         :rem 209
1040 :                                       :rem 255
1070 :                                         :rem 2
1080 REM"---SET TOP OF BASIC IF REQUIRED--"
                                             :rem 152
1090 :                                         :rem 4
1100 IF PEEK (65534)=72 THEN 1140: REM"--C-64 COMP
     UTER--"                                 :rem 200
1110 IF PEEK (824)<>248 THEN 1180: REM"--NOT PROG'
     D LOAD--"                                :rem 65
1120 POKE 42,PEEK (201): POKE 43,PEEK (202): GOTO
     {SPACE}1180                             :rem 209
1130 :                                       :rem 255
1140 : POKE 53280,14: POKE 53281,14: PRINT CHR$(31
     );: REM"--COLORS--"                      :rem 95
1150 IF PEEK (824)<>248 THEN 1180: REM"--NOT PROG'
     D LOAD--"                                :rem 69
1160 POKE 45,PEEK (174): POKE 46,PEEK (175)
                                             :rem 176
1170 :                                         :rem 3
1180 : CLR : POKE 824,0: GOSUB 1550: GOTO 2400
                                               :rem 6
1190 :                                         :rem 5
1200 REM"====== START OF SUBROUTINES======":rem 77
1210 :                                       :rem 254
```

```
1220 REM"--SUB--INPUT RECORDS--"            :rem 135
1230 :                                        :rem Ø
1240 : INPUT# 8,DA$(NR): TT=ST: NC=NC+LEN(DA$(NR))
     : NR=NR+1                                :rem 25
1250 PRINT "{UP}";NR: IF TT<>Ø OR NC>MC OR NR>MR T
     HEN RETURN                               :rem 77
1260 GOTO 1240                               :rem 201
1270 :                                        :rem 4
1280 REM"--SUB--DATA ENTRY--"                :rem 151
1290 :                                        :rem 6
1300 : TS$(Ø)=DA$(PL): NC=NC-LEN(DA$(ML%)): DA$(ML
     %)="": SYS SP: MF=FT%                   :rem 227
1310 FOR JJ=1 TO NF: IF JJ>MF THEN TS$(JJ)=""
                                             :rem 115
1320 : PRINT "{DOWN}{RVS} LINE#";ML%*1Ø+BN;"{LEFT}
     , FIELD";JJ;"{LEFT} {OFF}";CR$;" ";TS$(JJ);
                                              :rem 64
1330 AS=ASC(RIGHT$(TS$(JJ),1)+ZR$)           :rem 187
1340 IF AS=32 OR AS=16Ø THEN PRINT "{LEFT}{RVS}";C
     HR$(16Ø);                               :rem 164
1350 PRINT CR$;"{UP}";QT$;CR$;"{UP}";         :rem 98
1360 INPUT# 1,TC$(Ø): PRINT : LT=LEN(TC$(Ø))
                                              :rem 96
1370 JA=Ø: IF LT<1 THEN 145Ø                 :rem 146
1380 : JA=JA+1: IF MID$(TC$(Ø),JA,1)<>QT$ THEN 14Ø
     Ø                                        :rem 84
1390 PRINT Y1$;"{DOWN} {RVS} QUOTE INSIDE TEXT
     {2 SPACES}{OFF}";YØ$: GOTO 144Ø         :rem 239
14ØØ : IF JA<LT THEN 138Ø                    :rem 24Ø
141Ø FA$="TC": SYS SP: FA$="TS"              :rem 114
142Ø IF FT%<2 AND RIGHT$(TC$(Ø),1)<>FD$ THEN 145Ø
                                               :rem 3
143Ø PRINT Y1$;"{DOWN} {RVS} DELIMITER IN TEXT
     {2 SPACES}{OFF} ";YØ$;QT$;FD$;QT$       :rem 139
144Ø : PRINT " {RVS} RE-ENTER THE FIELD {OFF}": GO
     TO 132Ø                                  :rem 56
145Ø : AS=ASC(RIGHT$(TC$(Ø),1)+ZR$)          :rem 132
146Ø IF AS=16Ø THEN TC$(Ø)=LEFT$(TC$(Ø),LT-1)+" "
                                             :rem 223
147Ø DA$(ML%)=DA$(ML%)+TC$(Ø)+FD$: NEXT JJ:rem 198
148Ø :                                        :rem 7
149Ø NC=NC+LEN(DA$(ML%)): IF LEN(DA$(ML%))<74 THEN
      RETURN                                 :rem 167
15ØØ PRINT "{RVS} LINE IS";LEN(DA$(ML%))-73;"
     {LEFT}{RVS} CHARACTERS TOO LONG "        :rem 87
151Ø GOTO 13ØØ                               :rem 196
152Ø :                                        :rem 2
153Ø REM"--SUB--INITIALIZE--"                :rem 219
154Ø :                                        :rem 4
```

```
155Ø : YØ$="": Y1$="": MR=7ØØ: MC=14ØØØ: REM"--MAX
      RECORDS & CHR'S--"                       :rem 2Ø8
156Ø DIM DA$(MR),TS$(2Ø),TC$(8Ø)               :rem 137
157Ø TY=2: IF PEEK (65534)=72 THEN TY=6: YØ$=CHR$(
     31): Y1$=CHR$(158)                         :rem 19
158Ø LC=59468: IF TY=6 THEN LC=53272           :rem 145
159Ø CA$="": IF (PEEK (LC) AND 2)<>Ø THEN CA$="
     {DOWN}"                                    :rem 47
16ØØ CR$=CHR$(13): QT$=CHR$(34): CU$=CR$+"{UP}": S
     1=22: TA=6: TB=18                         :rem 111
161Ø ID$="Ø": FT%=Ø: M1=2: SS=3Ø976: SP=SS+3: RN=1
     ØØØ: BN=1Ø1Ø: LN=13                        :rem 17
162Ø RL$=Y1$+"{DOWN}{RVS}{3 SPACES}OUT OF RANGE
     {2 SPACES}{OFF}"+YØ$: YC$="YOUR CHOICE -----"
                                               :rem 194
163Ø NR=Ø: LL=-1: ZR$=CHR$(Ø): AD$=Y1$+"{DOWN}
     {RVS} ALREADY DELETED {OFF}"+YØ$          :rem 2Ø8
164Ø OPEN 1,Ø: OPEN 15,8,15: RETURN            :rem 1Ø8
165Ø :                                           :rem 6
166Ø REM"--SUB-- WAIT FOR OPERATOR --"         :rem 117
167Ø :                                           :rem 8
168Ø : PRINT "{DOWN}PRESS ANY KEY TO CONTINUE"
                                                :rem 81
169Ø : GET KB$: IF KB$<>"" THEN 169Ø           :rem 21Ø
17ØØ : GET KB$: IF KB$="" THEN 17ØØ            :rem 133
171Ø RETURN                                    :rem 169
172Ø :                                           :rem 4
173Ø REM"--SUB--WAIT FOR YES OR NO--"           :rem 54
174Ø :                                           :rem 6
175Ø : KB$="Y"                                 :rem 146
176Ø : PRINT CU$;SPC(S1);"? ";KB$;CU$;SPC(S1+2);
                                               :rem 118
177Ø : INPUT# 1,KB$: PRINT : KB$=LEFT$(KB$,1)
                                               :rem 1Ø3
178Ø IF KB$="Y" OR KB$="N" THEN RETURN          :rem 26
179Ø PRINT "{RVS} Y {OFF} YES OR {RVS} N {OFF} NO"
     ;CU$;SPC(S1)"? ";: GOTO 177Ø              :rem 173
18ØØ :                                           :rem 3
181Ø REM"--SUB--TEST/PRINT DISK ERROR--"       :rem 155
182Ø :                                           :rem 5
183Ø : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN=63 TH
     EN RETURN                                 :rem 251
184Ø PRINT Y1$;"{DOWN}{RVS} DISK ERROR {OFF}";YØ$;
     CR$;EN;EM$;ET;ES: RETURN                   :rem 77
185Ø :                                           :rem 8
186Ø REM"--SUB--STRING INPUT--"                 :rem 86
187Ø :                                          :rem 1Ø
188Ø : PRINT CU$;SPC(S1);"? ";K1$;CU$;SPC(S1+2);
                                               :rem 1Ø4
189Ø INPUT# 1,K1$: PRINT : RETURN               :rem 99
```

```
1900 :                                        :rem 4
1910 REM"--SUB--NUMERIC INPUT--"            :rem 142
1920 :                                        :rem 6
1930 : PRINT CU$;SPC(S1);"?";K2;CU$;SPC(S1+2);
                                             :rem 65
1940 INPUT# 1,KB$: PRINT : K2=VAL(KB$): RETURN
                                             :rem 73
1950 :                                        :rem 9
1960 REM"--SUB--TEST DELIMITER--"           :rem 207
1970 :                                       :rem 11
1980 : EF=Ø: IF FD$<>"" THEN 2ØØØ           :rem 42
1990 PRINT "{RVS} ENCLOSE COMMA/COLON/SPACE IN QUO
     TES {OFF}": EF=1: RETURN               :rem 2ØØ
2ØØØ : IF LEN(FD$)<>1 OR ASC(FD$)<32 THEN EF=1
                                             :rem 84
2Ø1Ø IF ASC(FD$)>127 AND ASC(FD$)<161 THEN EF=1
                                            :rem 165
2Ø2Ø IF FD$="Ø" OR VAL(FD$)<>Ø THEN EF=1     :rem 7
2Ø3Ø IF EF=1 THEN PRINT Y1$;"{RVS} ILLEGAL DELIMIT
     ER {OFF}{2 SPACES}";YØ$;FD$             :rem 68
2Ø4Ø RETURN                                 :rem 166
2Ø5Ø :                                        :rem 1
2Ø6Ø REM"--SUB--DISK CHANGE/INITIALIZE--" :rem 218
2Ø7Ø :                                        :rem 3
2Ø8Ø : ER=Ø                                   :rem 8
2Ø9Ø PRINT "{DOWN}NEED A NEW DISK";: GOSUB 175Ø: I
     F KB$="N" THEN ER=1: RETURN            :rem 12Ø
21ØØ : IF TY=6 THEN ID$="Ø": GOTO 213Ø     :rem 118
211Ø PRINT "WHICH DRIVE";: K1$=ID$: GOSUB 188Ø: ID
     $=K1$                                  :rem 216
212Ø IF ID$<"Ø" OR ID$>"1" THEN 21ØØ        :rem 172
213Ø : PRINT "{DOWN}INSTALL NEW DISK -- THEN{UP}":
      GOSUB 168Ø                            :rem 177
214Ø PRINT# 15,"I";ID$: GOSUB 183Ø: IF EN=Ø THEN R
     ETURN                                  :rem 224
215Ø PRINT "CAN'T INITIALIZE -- TRY AGAIN.": GOTO
     {SPACE}21ØØ                            :rem 245
216Ø :                                        :rem 3
217Ø REM"--SUB--CHANGE CASE--"              :rem 172
218Ø :                                        :rem 5
219Ø : CV=PEEK(LC)                          :rem 225
22ØØ IF (CV AND 2)=2 THEN POKE LC,(CV AND 253): CA
     $="": RETURN                           :rem 158
221Ø POKE LC,(CV OR 2): CA$="{DOWN}": RETURN
                                            :rem 25Ø
222Ø :                                        :rem Ø
223Ø REM"--SUB--TEST #BLOCKS FREE--"         :rem 42
224Ø :                                        :rem 2
225Ø : OPEN 14,8,Ø,"$"+ID$+":": GOSUB 183Ø: IF EN=
     Ø THEN 229Ø                            :rem 19Ø
```

```
226Ø CLOSE 14: PRINT Y1$;"{RVS} CAN'T READ OUTPUT
     {SPACE}DISK DIRECTORY {OFF}";YØ$         :rem 95
227Ø GOSUB 2Ø8Ø: IF ER=1 THEN RETURN        :rem 242
228Ø ER=Ø: GOTO 225Ø                          :rem 12
229Ø : FOR JJ=1 TO 18: GET #14,X1$,X2$: NEXT JJ: C
     LOSE 14                                 :rem 115
23ØØ BF%=ASC(X1$+ZR$)+ASC(X2$+ZR$)*256       :rem 66
231Ø PRINT "{DOWN}";BF%;TAB(TA);"BLOCKS FREE"
                                             :rem 231
232Ø IF BF%>(MC+2*MR)/254+2 THEN RETURN     :rem 126
233Ø ER=1: PRINT Y1$;"{RVS} NOT ENOUGH BLOCKS FREE
      {OFF}";YØ$: M1=7: RETURN                :rem 42
234Ø :                                         :rem 3
235Ø REM"====== START MAIN PROGRAM ======":rem 153
236Ø :                                         :rem 5
237Ø REM"---MENU AND SELECTION--"           :rem 131
238Ø :                                         :rem 7
239Ø : GOSUB 168Ø                             :rem 87
24ØØ : PRINT "{CLR}{RVS}{4 SPACES}DATA ENTRY AND S
     ORTING FUNCTIONS{3 SPACES}{OFF}"        :rem 1Ø2
241Ø PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CHANGE DI
     SPLAY/PRINT CASE"                        :rem 18
242Ø PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}LOAD DATA
      FILE FROM DISK"                         :rem 57
243Ø IF NR>Ø THEN PRINT "{DOWN}{RVS} 3 {OFF}
     {2 SPACES}SORT THE FILE"                :rem 151
244Ø IF NR>Ø THEN PRINT "{DOWN}{RVS} 4 {OFF}
     {2 SPACES}SAVE THE FILE"                :rem 128
245Ø IF NR<1 THEN PRINT "{DOWN}{RVS} 5 {OFF}
     {2 SPACES}CREATE A NEW FILE": GOTO 248Ø
                                             :rem 145
246Ø IF NR<>LL+1 THEN PRINT "{DOWN}{RVS} 5 {OFF}
     {2 SPACES}EDIT OR DELETE RECORDS": GOTO 248Ø
                                              :rem 32
247Ø PRINT "{DOWN}{RVS} 5 {OFF}{2 SPACES}ADD, EDIT
     , OR DELETE RECORDS"                     :rem 53
248Ø : IF LL>-1 THEN PRINT "{DOWN}{RVS} 6 {OFF}
     {2 SPACES}LIST THE FILE"                :rem 243
249Ø PRINT "{DOWN}{RVS} 7 {OFF}{2 SPACES}INITIALIZ
     E ANOTHER DISK"                         :rem 186
25ØØ PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR G
     O TO MASTER MENU{DOWN}"                 :rem 181
251Ø : PRINT YC$;: K2=M1: GOSUB 193Ø: IF K2=1 THEN
      GOSUB 219Ø                             :rem 178
252Ø IF K2=2 THEN 265Ø                        :rem 64
253Ø IF K2=7 THEN GOSUB 21ØØ: GOTO 24ØØ     :rem 245
254Ø IF K2=9 THEN 43ØØ                        :rem 67
255Ø IF NR<1 THEN 259Ø                       :rem 1Ø3
256Ø IF K2=3 THEN 294Ø                        :rem 71
257Ø IF K2=4 THEN M1=3: GOTO 3Ø9Ø            :rem 167
```

```
258Ø IF K2=5 THEN M1=4: GOTO 353Ø          :rem 169
259Ø : IF K2=5 AND NR<1 THEN M1=4: GOTO 362Ø
                                           :rem 196
26ØØ IF K2=6 AND LL>-1 THEN M1=5: GOTO 413Ø
                                           :rem 168
261Ø PRINT "{UP}";: GOTO 251Ø              :rem 161
262Ø :                                       :rem 4
263Ø REM"---LOAD FILE FROM DISK--"         :rem 115
264Ø :                                       :rem 6
265Ø : CLR : PRINT : GOSUB 155Ø: IF TY=6 THEN K1$=
     "Ø": GOTO 268Ø                        :rem 224
266Ø : PRINT "INPUT FROM DRIVE #";         :rem 117
267Ø K1$=ID$: GOSUB 188Ø: IF K1$<"Ø" OR K1$>"1" TH
     EN 266Ø                               :rem 242
268Ø : ID$=K1$                             :rem 152
269Ø :                                      :rem 11
27ØØ : PRINT "SOURCE FILE NAME";: K1$=IL$: GOSUB 1
     88Ø: IL$=K1$: NA$=IL$                  :rem 47
271Ø IF LEN(IL$)>LN THEN IL$=LEFT$(IL$,LN): GOTO 2
     7ØØ                                   :rem 154
272Ø OPEN 8,8,8,ID$+":"+IL$+",S,R": GOSUB 183Ø
                                            :rem 89
273Ø IF EN<>Ø THEN CLOSE 8: GOSUB 2Ø8Ø: GOTO 24ØØ
                                            :rem 5Ø
274Ø :                                       :rem 7
275Ø INPUT# 8,X1$: IF ST<>Ø THEN 288Ø      :rem 173
276Ø FD$=LEFT$(X1$,1): GOSUB 198Ø: IF EF<>Ø THEN 2
     89Ø                                   :rem 24Ø
277Ø NR=Ø: PRINT "{DOWN}";TAB(TA);"DATA RECORDS LO
     ADED."                                :rem 115
278Ø GOSUB 124Ø: IF NC>MC OR NR>MR THEN 285Ø
                                            :rem 92
279Ø FOR JJ=1 TO LEN(DA$(Ø)): IF MID$(DA$(Ø),JJ,1)
     =FD$ THEN NF=NF+1                     :rem 242
28ØØ NEXT JJ: PRINT TAB(TA);"(";INT((NC+2*NR)/254+
     1);"DISK BLOCKS ){DOWN}"               :rem 16
281Ø PRINT NF;TAB(TA);"FIELDS PER DATA RECORD."
                                             :rem 8
282Ø PRINT " ";QT$;FD$;QT$;TAB(TA);"IS THE FIELD D
     ELIMITER."                            :rem 179
283Ø M1=3: LL=NR-1: GOTO 29ØØ                :rem 6
284Ø :                                       :rem 8
285Ø : PRINT Y1$;"{RVS} FILE TOO LARGE TO LOAD.
     {2 SPACES}MORE THAN: {OFF}"           :rem 1Ø9
286Ø PRINT "{RVS}";MR;"{LEFT} RECORDS{2 SPACES}-OR
     - ";MC;"{LEFT} CHARACTERS {OFF}";YØ$  :rem 247
287Ø CLOSE 8: CLR : GOSUB 155Ø: GOTO 239Ø   :rem 96
288Ø : PRINT Y1$;"{RVS} NO DATA RECORDS IN THE FIL
     E {OFF} ";YØ$                         :rem 179
```

```
2890 : M1=2: PRINT Y1$;"{DOWN} INPUT TERMINATED
     {OFF}";Y0$                                :rem 246
2900 : CLOSE 8: GOTO 2390                      :rem 244
2910 :                                           :rem 6
2920 REM"---SORT DATA BY FIELD--"               :rem 55
2930 :                                           :rem 8
2940 : PRINT "{DOWN}FIELD TO BE SORTED";      :rem 133
2950 K2=0: GOSUB 1930: FS%=K2: IF FS%<0 OR FS%>20
     {SPACE}THEN 2940                          :rem 227
2960 : PRINT "{RVS} A {OFF} ASCENDING OR";CR$;"
     {RVS} D {OFF} DESCENDING ORDER";          :rem 212
2970 K1$="A": GOSUB 1880: FO$=K1$: IF K1$<>"A" AND
      K1$<>"D" THEN 2960                       :rem 108
2980 :                                          :rem 13
2990 : FA$="DA": SYS SS: IF FS%=0 THEN NR=FT%
                                                :rem 93
3000 PRINT CR$;NR;TAB(TA);"TOTAL DATA RECORDS"
                                                :rem 11
3010 PRINT FT%;TAB(TA);"DATA RECORDS SORTED"
                                               :rem 132
3020 IF NR=FT% THEN 3040                       :rem 230
3030 PRINT NR-FT%;TAB(TA);"RECORDS WITH NULL IN FI
     ELD";FS%                                  :rem 211
3040 : M1=4: LL=FT%-1: IF DF<>0 THEN DF=0: M1=3
                                               :rem 103
3050 GOTO 2390                                 :rem 207
3060 :                                           :rem 3
3070 REM"---SAVE THE FILE--"                     :rem 3
3080 :                                           :rem 5
3090 : IF NR=LL+1 THEN SF=0: GOTO 3220         :rem 213
3100 PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}SAVE COMP
     LETE FILE"                                 :rem 35
3110 PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}SAVE ONLY
      THE {RVS}";LL+1;"{LEFT} {OFF} RECORDS"
                                               :rem 208
3120 PRINT TAB(TA);"WITH DATA IN FIELD {RVS}";FS%;
     "{LEFT} {OFF}{DOWN}"                      :rem 170
3130 : PRINT YC$;: K2=2: GOSUB 1930: IF K2=1 THEN
     {SPACE}SF=1: GOTO 3170                     :rem 94
3140 IF K2=2 THEN SF=0: GOTO 3220              :rem 178
3150 GOTO 3130                                 :rem 201
3160 :                                           :rem 4
3170 : PRINT "{DOWN}FILE WILL BE ERASED FROM MEMOR
     Y"                                        :rem 135
3180 PRINT "DURING THIS SAVE.{2 SPACES}PRESS {RVS}
      M {OFF} FOR"                              :rem 96
3190 PRINT "ANOTHER MENU SELECTION OR --{UP}": GOS
     UB 1680                                   :rem 159
3200 IF KB$="M" THEN 2400                      :rem 200
3210 :                                           :rem 0
```

```
3220 : PRINT                                    :rem 142
3230 IF IL$<>"" THEN PRINT "ORIGINAL FILE NAME";CU
     $;SPC(S1+2);IL$                            :rem 162
3240 : PRINT "NEW FILE NAME";: K1$=NA$: GOSUB 1880
     : NA$=K1$                                  :rem 89
3250 IF LEN(NA$)>LN THEN NA$=LEFT$(NA$,LN): GOTO 3
     240                                        :rem 136
3260 :                                          :rem 5
3270 IF TY=6 THEN K1$="Ø": GOTO 33ØØ            :rem 52
3280 : PRINT "OUTPUT TO DRIVE #";: K1$=ID$: GOSUB
     {SPACE}188Ø                                :rem 151
3290 IF K1$<"Ø" OR K1$>"1" THEN 328Ø            :rem 157
3300 : ID$=K1$: RE$=""                          :rem 4
3310 :                                          :rem 1
3320 GOSUB 225Ø: IF ER=1 THEN ER=Ø: GOTO 239Ø
                                                :rem 83
3330 : OPEN 8,8,8,RE$+ID$+":"+NA$+",S,W": GOSUB 18
     3Ø: IF EN=Ø THEN 339Ø                      :rem 61
3340 CLOSE 8: IF EN=63 THEN 336Ø                :rem 117
3350 GOSUB 2Ø8Ø: GOTO 24ØØ                      :rem 78
3360 : PRINT "REPLACE EXISTING FILE";: GOSUB 175Ø:
      IF KB$="N" THEN 24ØØ                      :rem 94
3370 RE$="@": GOTO 333Ø                         :rem 133
3380 :                                          :rem 8
3390 : PRINT# 8,QT$;FD$;"←@Ø:";NA$;CR$;         :rem 116
3400 PRINT "SAVING -- PLEASE WAIT --": IF SF=1 THE
     N 343Ø                                     :rem 2Ø4
3410 FOR JJ=Ø TO LL: PRINT# 8,QT$;DA$(JJ);CR$;: NE
     XT JJ: GOTO 347Ø                           :rem 72
3420 :                                          :rem 3
3430 : FOR JJ=Ø TO LL: PRINT# 8,QT$;DA$(JJ);CR$;:
     {SPACE}DA$(JJ)="": NEXT JJ                 :rem 14Ø
3440 FS%=Ø: FA$="DA": SYS SS                    :rem 192
3450 FOR JJ=Ø TO FT%-1: PRINT# 8,QT$;DA$(JJ);CR$;:
      DA$(JJ)="": NEXT JJ                       :rem 217
3460 NR=Ø: LL=-1: M1=2                          :rem 11Ø
3470 : GOSUB 183Ø: CLOSE 8                      :rem 6Ø
3480 IF EN<>Ø THEN PRINT Y1$;"{RVS} FILE NOT SAVED
      CORRECTLY {OFF}";YØ$                      :rem 75
3490 GOTO 239Ø                                  :rem 215
3500 :                                          :rem 2
3510 REM"---EDIT/ADD RECORDS--"                 :rem 2
3520 :                                          :rem 4
3530 : IF NR=LL+1 THEN PR$="A": GOTO 368Ø       :rem 96
3540 PRINT "{DOWN}{RVS} NOTE {OFF} - PRESENT SORT
     {SPACE}ON FIELD {RVS} #";FS%;"{LEFT}{RVS}
     {OFF}"                                     :rem 13
3550 PRINT "HAS PRODUCED {RVS}";NR-LL-1;"{LEFT}
     {OFF} RECORDS (OUT OF";NR                  :rem 48
3560 PRINT "TOTAL) WHICH CAN'T BE EDITED DUE TO"
                                               :rem 195
3570 PRINT "NULLS IN THAT FIELD.{DOWN}"       :rem 153
3580 PRINT "NEW RECORDS CAN NOT BE ADDED IN"
                                               :rem 208
3590 PRINT "PRESENT SORT CONDITION.{2 DOWN}":rem 2
3600 PR$="E": GOTO 3680                        :rem 153
3610 :                                           :rem 4
3620 : NF=2: FD$="!": PR$="A"                   :rem 84
3630 : PRINT "{DOWN}# FIELDS PER RECORD"; :rem 163
3640 K2=NF: GOSUB 1930: NF=K2: IF NF<1 OR NF>20 TH
     EN 3630                                   :rem 196
3650 : PRINT "DELIMITER TO BE USED";: K1$=FD$: GOS
     UB 1880: FD$=K1$                           :rem 35
3660 GOSUB 1980: IF EF=1 THEN 3650             :rem 224
3670 :                                          :rem 10
3680 : FA$="TS": EL%=LL*10+RN                  :rem 249
3690 : PRINT "{DOWN}";: IF NR=LL+1 THEN PRINT "
     {RVS} A {OFF} ADD{2 SPACES}";             :rem 172
3700 IF LL>-1 THEN PRINT "{RVS} D {OFF} DELETE";CR
     $;"{RVS} E {OFF} EDIT ";                  :rem 229
3710 PRINT "{RVS} F {OFF} FINISHED";            :rem 11
3720 K1$=PR$: GOSUB 1880: PR$=K1$              :rem 215
3730 IF PR$="A" AND NR=LL+1 THEN 3790          :rem 138
3740 IF PR$="E" AND LL>-1 THEN 3840            :rem 238
3750 IF PR$="D" AND LL>-1 THEN 3930            :rem 238
3760 IF PR$="F" THEN 4050                      :rem 228
3770 GOTO 3690                                 :rem 220
3780 :                                          :rem 12
3790 : IF NR<MR AND NC<MC THEN 3810             :rem 66
3800 PRINT Y1$;"{RVS} FILE SIZE LIMIT REACHED
     {OFF}";Y0$: M1=4: GOTO 2390                :rem 65
3810 : PL=LL: LL=LL+1: ML%=LL: NR=NR+1: IF LL=0 TH
     EN PL=0                                    :rem 96
3820 EL%=LL*10+BN: GOSUB 1300: GOTO 3690        :rem 97
3830 :                                           :rem 8
3840 : PRINT "{DOWN}EDIT LINE #{2 SPACES}-------";
     : K2=EL%+10                                 :rem 8
3850 IF K2>LL*10+BN THEN K2=LL*10+BN           :rem 191
3860 GOSUB 1930: EL%=K2                        :rem 200
3870 IF EL%<BN THEN PRINT TAB(TB);RL$: EL%=RN: GOT
     O 3840                                    :rem 105
3880 IF EL%>LL*10+BN THEN PRINT TAB(TB);RL$: EL%=L
     L*10+RN: GOTO 3840                          :rem 8
3890 ML%=(EL%-BN)/10+.5: PL=ML%                :rem 130
3900 IF DA$(ML%)="" THEN PRINT TAB(TB);AD$: GOTO 3
     690                                        :rem 55
3910 GOSUB 1300: GOTO 3690                      :rem 86
3920 :                                           :rem 8
```

```
393Ø : PRINT "{DOWN}DELETE LINE #{2 SPACES}-----";
     : K2=EL%+1Ø                                :rem 59
394Ø IF K2>LL*1Ø+BN THEN K2=LL*1Ø+BN            :rem 191
395Ø GOSUB 193Ø: EL%=K2                          :rem 2ØØ
396Ø IF EL%<BN THEN PRINT TAB(TB);RL$: EL%=RN: GOT
     O 393Ø                                     :rem 1Ø5
397Ø IF EL%>LL*1Ø+BN THEN PRINT TAB(TB);RL$: EL%=L
     L*1Ø+RN: GOTO 393Ø                           :rem 8
398Ø ML%=(EL%-BN)/1Ø+.5                          :rem 177
399Ø IF DA$(ML%)="" THEN PRINT TAB(TB);AD$: GOTO 3
     69Ø                                         :rem 64
4ØØØ PRINT "{DOWN}";EL%;" ";QT$;DA$(ML%)     :rem 13
4Ø1Ø PRINT "{DOWN}ARE YOU SURE";: KB$="N": GOSUB 1
     76Ø: IF KB$="N" THEN 369Ø                  :rem 13Ø
4Ø2Ø NC=NC-LEN(DA$(ML%)): DA$(ML%)=""         :rem 173
4Ø3Ø DF=1: M1=3: PRINT TAB(S1+4);"{UP}{RVS} DELETE
     D {OFF}": GOTO 369Ø                        :rem 169
4Ø4Ø :                                            :rem 2
4Ø5Ø : IF DF=Ø THEN 24ØØ                        :rem 126
4Ø6Ø PRINT "{DOWN}DUE TO DELETIONS, THE FILE IS NO
     W BEING"                                   :rem 246
4Ø7Ø PRINT "SORTED ON FIELD #Ø IN ASCENDING ORDER.
     "                                          :rem 142
4Ø8Ø PRINT "-- YOU MAY RE-SORT AS DESIRED --"
                                                :rem 213
4Ø9Ø FO$="A": FS%=Ø: GOTO 299Ø                  :rem 244
41ØØ :                                          :rem 255
411Ø REM"---LIST THE FILE--"                     :rem 12
412Ø :                                            :rem 1
413Ø : IF NR=LL+1 THEN 416Ø                      :rem 92
414Ø PRINT "{DOWN}ONLY THE {RVS}";LL+1;"{LEFT}
     {OFF} RECORDS WITH DATA"                    :rem 37
415Ø PRINT "IN FIELD {RVS}";FS%;"{LEFT} {OFF} WILL
      BE LISTED."                                 :rem 1
416Ø : PRINT "{DOWN}PRESS ANY KEY TO PAUSE, THEN -
     -"                                          :rem 27
417Ø PRINT "{RVS} Q {OFF} TO QUIT LISTING OR ANY O
     THER"                                      :rem 157
418Ø PRINT "KEY TO CONTINUE."                    :rem 189
419Ø OL=1ØØØ: LP=1: OPEN 4,4: PRINT# 4,;OL;QT$;FD$
     ;"←@Ø:";IL$;CR$;                            :rem 99
42ØØ FOR JJ=Ø TO LL: OL=OL+1Ø: PRINT# 4,OL;CA$;QT$
     ;DA$(JJ);CR$;                              :rem 23Ø
421Ø LP=LP+1: IF LP=>6Ø THEN LP=Ø: FOR JA=1 TO 6:
     {SPACE}PRINT# 4,CR$;: NEXT JA               :rem 75
422Ø GET KB$: IF KB$="" THEN 425Ø                :rem 78
423Ø : GET KB$: IF KB$="" THEN 423Ø             :rem 135
424Ø IF KB$="Q" THEN JJ=LL                      :rem 116
425Ø : NEXT JJ: FOR JA=1 TO 66-LP: PRINT# 4,CR$;:
     {SPACE}NEXT JA                              :rem 12
```

```
4260 CLOSE 4: M1=5: GOTO 24ØØ                  :rem 217
4270 :                                          :rem 7
4280 REM"---PROGRAM TERMINATION--"              :rem 57
4290 :                                          :rem 9
43ØØ : CLOSE 4: CLOSE 8: CLOSE 14              :rem 168
431Ø : PRINT "{DOWN}PRESS {RVS} Q {OFF} TO QUIT OR
     --"                                        :rem 71
432Ø PRINT "ANY OTHER KEY FOR MASTER MENU":rem 213
433Ø GOSUB 169Ø: IF KB$<>"Q" THEN 435Ø         :rem 158
434Ø PRINT "{RVS} PROGRAM TERMINATED {OFF}";: CLOS
     E 1: CLOSE 15: END                        :rem 136
435Ø : PS$="DFH BOOT": OPEN 8,8,8,"Ø:"+PS$+",P,R"
                                               :rem 245
436Ø GOSUB 183Ø: CLOSE 8: IF EN=Ø THEN 439Ø
                                               :rem 2Ø1
437Ø PRINT "{DOWN}TRYING TO LOAD {RVS} ";PS$;"
     {OFF}"                                    :rem 117
438Ø GOSUB 2Ø8Ø: GOTO 431Ø                      :rem 84
439Ø : CLOSE 1: CLOSE 15: PRINT "{DOWN}{RVS} LOADI
     NG ";PS$;" {OFF}"                           :rem 6
44ØØ POKE 824,248: LOAD PS$,8                  :rem 228
```

DFH PRINT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:DFH PRINT",8                 :rem 133
1Ø1Ø :                                         :rem 252
1Ø2Ø REM" A DATA FILE HANDLER PROGRAM FOR PRINTING
      MULTI-FIELD SEQUENTIAL"                  :rem 229
1Ø3Ø REM" DATA FILES TO SCREEN, PAPER, OR WORDPRO
     {SPACE}FILES UNDER CONTROL OF"             :rem 33
1Ø4Ø REM" A USER DEFINED FORMAT WHICH CAN BE STORE
     D FOR FUTURE USE."                        :rem 159
1Ø5Ø :                                           :rem Ø
1Ø8Ø :                                           :rem 3
1Ø9Ø REM"---SET TOP OF BASIC IF REQUIRED--"
                                               :rem 153
11ØØ :                                         :rem 252
111Ø IF PEEK (65534)=72 THEN 115Ø: REM"-- C-64 COM
     PUTER --"                                 :rem 2Ø2
112Ø IF PEEK (824)<>248 THEN 119Ø: REM"-- NOT A PR
     OG'D LOAD --"                             :rem 132
113Ø POKE 42,PEEK (2Ø1): POKE 43,PEEK (2Ø2): GOTO
     {SPACE}119Ø                               :rem 211
114Ø :                                           :rem Ø
115Ø : POKE 5328Ø,14: POKE 53281,14: PRINT CHR$(31
     );: REM"--COLORS--"                        :rem 96
116Ø IF PEEK (824)<>248 THEN 119Ø: REM"-- NOT A PR
     OG'D LOAD --"                             :rem 136
```

```
1170 POKE 45,PEEK (174): POKE 46,PEEK (175)
                                           :rem 177
1180 :                                      :rem 4
1190 : CLR : POKE 824,0: TY=2: Y0$="": Y1$=""
                                           :rem 112
1200 IF PEEK (65534)=72 THEN TY=6: Y0$=CHR$(31): Y
     1$=CHR$(158)                          :rem 179
1210 GOTO 4080                             :rem 201
1220 :                                     :rem 255
1230 REM"========= START OF SUBROUTINES =========
     "                                     :rem 251
1240 :                                      :rem 1
1250 REM"--SUB--PARTITION & PRINT RECORDS--"
                                           :rem 103
1260 :                                      :rem 3
1270 : INPUT# 8,DA$(0): TT=ST: IF LEN(DA$(0))>80 T
     HEN 5950                              :rem 214
1280 SYS SP?: WS$="": FOR JB=1 TO CC: WS$=WS$+LEFT$
     (BL$,FB(JB))                           :rem 91
1290 IF NF(JB)>FT% THEN WS$=WS$+LEFT$(BL$,FC(JB)):
      GOTO 1340                             :rem 27
1300 LE=FC(JB)-LEN(DA$(NF(JB))): IF -LE>OL(JB) THE
     N OL(JB)=-LE                          :rem 159
1310 IF LE<0 THEN OC(JB)=OC(JB)+1: LE=0     :rem 47
1320 IF FJ$(JB)="L" THEN WS$=WS$+DA$(NF(JB))+LEFT$
     (BL$,LE): GOTO 1340                   :rem 198
1330 WS$=WS$+LEFT$(BL$,LE)+DA$(NF(JB))     :rem 112
1340 : NEXT JB: IF SE%<3 THEN PRINT# DV,CA$;WS$;CR
     $;                                     :rem 71
1350 IF SE%>1 THEN GOSUB 1950               :rem 2
1360 IF SE%<3 AND DV<>3 THEN PL%=PL%+1     :rem 233
1370 LL%=LL%+1: DL%=DL%+1: IF LL%=DM% THEN PN=PN+1
     : GOSUB 1460: GOSUB 1580              :rem 165
1380 GET KB$: IF KB$="" THEN 1410           :rem 77
1390 : GET KB$: IF KB$="" THEN 1390        :rem 143
1400 IF KB$="Q" THEN RETURN                :rem 230
1410 : IF TT=0 THEN 1270                   :rem 157
1420 RETURN                                :rem 167
1430 :                                      :rem 2
1440 REM"--SUB--TOP OF FORM & NEW WORDPRO FILE--"
                                            :rem 2
1450 :                                      :rem 4
1460 : LL%=0: IF DV<>3 AND PL%<>0 THEN BK=66-PL%:
     {SPACE}GOSUB 1740: PL%=0              :rem 190
1470 IF SE%<2 THEN RETURN                  :rem 149
1480 GOSUB 2030: GOSUB 1910: GOSUB 1880: PRINT# 9,
     " ";: CLOSE 9                         :rem 217
1490 : GOSUB 2560: IF OK<>0 THEN 1490       :rem 95
1500 IF SF>49 THEN 1530                    :rem 145
```

```
1510 IF TY<>6 THEN PRINT DC$: GOSUB 2460: GOTO 149
     0                                          :rem 225
1520 GOTO 5930                                  :rem 210
1530 : NA$=FW$: DR$=DO$: GOSUB 2290: IF EN<>0 THEN
      GOSUB 2460: GOTO 1490                     :rem 229
1540 GOSUB 1890: RETURN                          :rem 54
1550 :                                           :rem 5
1560 REM"--SUB--PRINT PAGE/FIELD HEADINGS--"
                                                :rem 108
1570 :                                           :rem 7
1580 : BK=B1: GOSUB 1750                        :rem 204
1590 WS$=HA$: IF HE=0 THEN WS$=WS$+STR$(PN)+" "+HB
     $                                          :rem 209
1600 IF SE%<3 THEN PRINT# DV,CA$;WS$;CR$;        :rem 7
1610 IF SE%>1 THEN GOSUB 1950                    :rem 1
1620 BK=B2: GOSUB 1750: WS$="": FOR JH=1 TO CC
                                                 :rem 97
1630 WS$=WS$+LEFT$(BL$,FB(JH))                  :rem 183
1640 IF FJ$(JH)="R" THEN WS$=WS$+LEFT$(BL$,LH(JH))
     +FH$(JH): GOTO 1660                        :rem 236
1650 WS$=WS$+FH$(JH)+LEFT$(BL$,LH(JH))          :rem 133
1660 : NEXT JH: IF SE%<3 THEN PRINT# DV,CA$;WS$;CR
     $;                                          :rem 82
1670 IF SE%>1 THEN GOSUB 1950                    :rem 7
1680 BK=B3: GOSUB 1750: LL%=2+B1+B2+B3          :rem 216
1690 IF SE%<3 AND DV<>3 THEN PL%=PL%+LL%: IF PL%>6
     6 THEN PL%=PL%-66                           :rem 54
1700 RETURN                                     :rem 168
1710 :                                           :rem 3
1720 REM"--SUB--PRINT BLANK LINES--"             :rem 90
1730 :                                           :rem 5
1740 : FP=1                                      :rem 10
1750 : IF BK<1 THEN 1790                        :rem 144
1760 TS=39: FOR JJ=1 TO BK: IF SE%>1 AND FP=0 THEN
      GOSUB 1990                                :rem 190
1770 IF SE%<3 THEN PRINT# DV,CR$;                :rem 35
1780 NEXT JJ                                    :rem 163
1790 : FP=0: RETURN                              :rem 40
1800 :                                           :rem 3
1810 REM"--SUB--CLOSE WORDPRO FILE--"           :rem 173
1820 :                                           :rem 5
1830 : IF SE%>1 THEN GOSUB 1910: GOSUB 1920: PRINT
     # 9," ";: CLOSE 9                          :rem 120
1840 SE%=1: GOSUB 1460: RETURN                  :rem 151
1850 :                                           :rem 8
1860 REM"--SUB--WORDPRO MESSAGES, CONVERSION & LIN
     E PAD--"                                   :rem 201
1870 :                                           :rem 10
1880 : WS$=CK$+"NX:"+FW$+NP$: GOTO 1930          :rem 48
```

```
189Ø : PRINT# 9,BF$;: WS$=CK$+"CM:"+FW$+NP$: GOTO
     {SPACE}193Ø                                :rem 81
19ØØ : WS$=CK$+"PP66:PG62:LM1:RM79": GOTO 193Ø
                                                :rem 2
191Ø : WS$=CK$+"FP": GOTO 193Ø                  :rem 7
192Ø : WS$=CK$+"CM:END"                        :rem 211
193Ø : WC$="{DOWN}": GOTO 196Ø                 :rem 154
194Ø :                                          :rem 8
195Ø : WC$=CA$: IF WS$="" THEN RETURN          :rem 211
196Ø : LS=LEN(WS$): TS=39-LS-4Ø*(LS>39)-4Ø*(LS>79)
     -4Ø*(LS>119)                              :rem 1Ø8
197Ø IF LEFT$(WS$,1)=CK$ THEN PRINT# 9,CK$;: WS$=M
     ID$(WS$,2)                                 :rem 11
198Ø SYS SW: PRINT# 9,WS$;                     :rem 211
199Ø : PRINT# 9,CHR$(31);LEFT$(BL$,TS);: RETURN
                                               :rem 141
2ØØØ :                                         :rem 252
2Ø1Ø REM"--SUB--INCREMENT & FORMAT WORDPRO FILE NU
     MBER--"                                   :rem 141
2Ø2Ø :                                         :rem 254
2Ø3Ø : PC=PC+1: NP$="": IF PC>99 THEN 2Ø5Ø :rem 61
2Ø4Ø NP$="Ø": IF PC<1Ø THEN NP$=NP$+"Ø"    :rem 193
2Ø5Ø : NP$="."+NP$+MID$(STR$(PC),2): RETURN:rem 82
2Ø6Ø :                                          :rem 2
2Ø7Ø REM"--SUB--TEST DELIMITER--"              :rem 2ØØ
2Ø8Ø :                                          :rem 4
2Ø9Ø : EF=Ø: IF FD$<>"" THEN 211Ø             :rem 37
21ØØ PRINT "{RVS} ENCLOSE COMMA/COLON/SPACE IN QUO
     TES {OFF}": EF=1: RETURN                  :rem 184
211Ø : IF LEN(FD$)<>1 OR ASC(FD$)<32 THEN EF=1
                                                :rem 86
212Ø IF ASC(FD$)>127 AND ASC(FD$)<161 THEN EF=1
                                               :rem 167
213Ø IF FD$="Ø" OR VAL(FD$)<>Ø THEN EF=1      :rem 9
214Ø IF EF=1 THEN PRINT Y1$;"{RVS} ILLEGAL DELIMIT
     ER {OFF}{2 SPACES}";YØ$;FD$               :rem 7Ø
215Ø RETURN                                    :rem 168
216Ø :                                          :rem 3
217Ø REM"--SUB--SET UP INPUT FILE--"           :rem 43
218Ø :                                          :rem 5
219Ø : PRINT : DR$=DI$: OPEN 8,8,8,DR$+":"+NA$+",S
     ,R"                                       :rem 186
22ØØ GOSUB 3Ø1Ø: IF EN<>Ø THEN 224Ø            :rem 5
221Ø INPUT# 8,X1$: IF ST=Ø THEN 223Ø          :rem 92
222Ø PRINT Y1$;"{RVS} NO DATA RECORDS IN THE FILE
     {SPACE}{OFF}";YØ$: GOTO 224Ø              :rem 168
223Ø : FD$=LEFT$(X1$,1): GOSUB 2Ø9Ø: IF EF=Ø THEN
     {SPACE}RETURN                             :rem 235
224Ø : CLOSE 8: ER=1: RETURN                   :rem 9
225Ø :                                          :rem 3
```

```
226Ø REM"--SUB--SET UP OUTPUT FILE--"      :rem 14Ø
227Ø :                                        :rem 5
228Ø : NE$="": TF$=",S,W": GOTO 23ØØ      :rem 241
229Ø : NE$=NP$: TF$=",P,W"                 :rem 53
23ØØ : RE$="": PRINT                        :rem 2
231Ø : OPEN 9,8,9,RE$+DR$+":"+NA$+NE$+TF$  :rem 82
232Ø GOSUB 3Ø1Ø: IF EN=Ø THEN RETURN      :rem 227
233Ø CLOSE 9: IF EN<>63 THEN RETURN       :rem 197
234Ø PRINT "REPLACE EXISTING FILE";        :rem 92
235Ø GOSUB 272Ø: IF KB$="Y" THEN RE$="@": GOTO 231
     Ø                                     :rem 77
236Ø RETURN                               :rem 171
237Ø :                                        :rem 6
238Ø REM"--SUB--GET FILE NAME--"           :rem 14
239Ø :                                        :rem 8
24ØØ : PRINT "FILE NAME";: K1$=NA$: GOSUB 3Ø8Ø: NA
     $=K1$                                :rem 1Ø2
241Ø IF LEN(NA$)=<LN THEN RETURN           :rem 91
242Ø NA$=LEFT$(NA$,LN): PRINT "{RVS} NAME TOO LONG
      {OFF}": GOTO 24ØØ                   :rem 252
243Ø :                                        :rem 3
244Ø REM"--SUB--DISK CHANGE/INITIALIZE--" :rem 22Ø
245Ø :                                        :rem 5
246Ø : KB$="Y"                            :rem 145
247Ø PRINT "{DOWN}NEED A NEW DISK";: GOSUB 273Ø: I
     F KB$="N" THEN RETURN                 :rem 58
248Ø PRINT "{DOWN}CHANGE DISK";           :rem 188
249Ø IF TY<>6 THEN PRINT " IN DRIVE #";DR$;:rem 31
25ØØ PRINT "{2 SPACES}-- THEN": GOSUB 265Ø:rem 168
251Ø PRINT# 15,"I"+DR$+CR$;: GOSUB 3Ø1Ø: IF EN<>Ø
     {SPACE}THEN 246Ø                      :rem 26
252Ø RETURN                               :rem 169
253Ø :                                        :rem 4
254Ø REM"--SUB--FIND BLOCKS FREE--"       :rem 236
255Ø :                                        :rem 6
256Ø : DR$=DO$: OPEN 14,8,Ø,"$"+DR$+":": GOSUB 3Ø1
     Ø                                    :rem 23Ø
257Ø IF EN<>Ø THEN CLOSE 14: GOSUB 246Ø: OK=1: RET
     URN                                  :rem 134
258Ø FOR JJ=1 TO 18: GET # 14,X1$,X2$: NEXT JJ: CL
     OSE 14                                :rem 59
259Ø SF=ASC(X1$+ZR$)+ASC(X2$+ZR$)*256: PRINT CR$;S
     F;"BLOCKS FREE";                      :rem 39
26ØØ IF TY<>6 THEN PRINT " ON DRIVE #";DR$;:rem 3Ø
261Ø PRINT CR$;" FOR WORDPRO FILE ";FW$;NP$: OK=Ø:
      RETURN                               :rem 22
262Ø :                                        :rem 4
263Ø REM"--SUB--WAIT FOR OPERATOR--"      :rem 115
264Ø :                                        :rem 6
265Ø : PRINT "PRESS ANY KEY TO CONTINUE"   :rem 62
```

```
266Ø : GET KB$: IF KB$<>"" THEN 266Ø         :rem 2Ø6
267Ø : GET KB$: IF KB$="" THEN 267Ø          :rem 147
268Ø RETURN                                  :rem 176
269Ø :                                        :rem 11
27ØØ REM"--SUB--WAIT FOR YES OR NO--"         :rem 52
271Ø :                                         :rem 4
272Ø : KB$="Y"                               :rem 144
273Ø : PRINT CU$;SPC(S2);"? ";KB$;CU$;SPC(S2+2);
                                             :rem 118
274Ø : INPUT# 1,KB$: PRINT : KB$=LEFT$(KB$,1)
                                             :rem 1Ø1
275Ø IF KB$="Y" OR KB$="N" THEN RETURN        :rem 24
276Ø PRINT "{RVS} Y {OFF} YES OR {RVS} N {OFF} NO
     {SPACE}? ";: GOTO 274Ø                  :rem 12Ø
277Ø :                                        :rem 1Ø
278Ø REM"--SUB--PRESET FORMAT DATA--"        :rem 167
279Ø :                                        :rem 12
28ØØ : FOR JJ=1 TO 2Ø: FB(JJ)=2: NF(JJ)=JJ: FJ$(JJ
     )="L": FC(JJ)=9                          :rem 74
281Ø FH$(JJ)=MID$(STR$(JJ),2): MF$(JJ)="Y": OC(JJ)
     =Ø: OL(JJ)=Ø                             :rem 67
282Ø NEXT JJ: FB(1)=Ø: H1$="": H2$=""          :rem 1
283Ø B1=3: B2=1: B3=1: D1=55: BP=1: PN=1: CC=2: FD
     $="|": RETURN                           :rem 146
284Ø :                                         :rem 8
285Ø REM"--SUB--FORM PAGE HEADING--"          :rem 48
286Ø :                                        :rem 1Ø
287Ø : HA$=H1$+H2$: HB$=HA$: HE=Ø: JJ=1     :rem 157
288Ø : IF MID$(HA$,JJ,2)="<>" THEN HA$=LEFT$(HA$,J
     J-1): GOTO 291Ø                         :rem 115
289Ø IF JJ=>LEN(HA$) THEN HE=1: RETURN       :rem 146
29ØØ JJ=JJ+1: GOTO 288Ø                      :rem 2Ø9
291Ø : HB$=MID$(HB$,JJ+2,74-LEN(HA$)): RETURN
                                             :rem 198
292Ø :                                         :rem 7
293Ø REM"--SUB--CHANGE CASE--"               :rem 176
294Ø :                                         :rem 9
295Ø : CV=PEEK (LC)                          :rem 229
296Ø IF (CV AND 2)=2 THEN POKE LC,(CV AND 253): CA
     $="": RETURN                            :rem 171
297Ø POKE LC,(CV OR 2): CA$="{DOWN}": RETURN:rem 7
298Ø :                                        :rem 13
299Ø REM"--SUB--TEST/PRINT DISK ERROR--"     :rem 165
3ØØØ :                                       :rem 253
3Ø1Ø : ER=Ø: INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN
     =63 THEN RETURN                          :rem 49
3Ø2Ø PRINT Y1$;"{DOWN}{RVS} DISK ERROR {OFF}";YØ$;
                                              :rem 12
3Ø3Ø IF TY<>6 THEN PRINT "{RVS}ON DRIVE #";DR$;"
     {OFF}";                                  :rem 63
```

```
3Ø4Ø PRINT CR$;EN;EM$;ET;ES: ER=1: RETURN :rem 2Ø4
3Ø5Ø :                                         :rem 2
3Ø6Ø REM"--SUB--STRING INPUT--"               :rem 8Ø
3Ø7Ø :                                         :rem 4
3Ø8Ø : PRINT CU$;SPC(S1);"? ";K1$;CU$;SPC(S1+2);:
     {SPACE}GOTO 31ØØ                          :rem 153
3Ø9Ø : PRINT CU$;SPC(S2);"? ";K1$;CU$;SPC(S2+2);
                                               :rem 1Ø1
31ØØ : INPUT# 1,K1$: PRINT : RETURN           :rem 143
311Ø :                                         :rem 255
312Ø REM"--SUB--NUMERIC INPUT--"              :rem 137
313Ø :                                         :rem 1
314Ø : PRINT CU$;SPC(S2);"?";K2;CU$;SPC(S2+2);
                                               :rem 62
315Ø INPUT# 1,KB$: PRINT : K2=VAL(KB$): RETURN
                                               :rem 68
316Ø :                                         :rem 4
317Ø REM"--SUB--LOAD FORMAT FILE--"           :rem 244
318Ø :                                         :rem 6
319Ø : PRINT "{RVS} LOAD PRINT FORMAT FILE {OFF}
     {DOWN}"                                   :rem 35
32ØØ NA$=FF$: LN=16: GOSUB 24ØØ: FF$=NA$      :rem 55
321Ø GOSUB 28ØØ: GOSUB 219Ø: IF ER<>Ø THEN GOSUB 2
     46Ø: OK=2: RETURN                         :rem 12Ø
322Ø GOSUB 334Ø: CC=VAL(DA$(1)): BP=VAL(DA$(2)): P
     N=BP: B1=VAL(DA$(3))                      :rem 2ØØ
323Ø B2=VAL(DA$(4)): B3=VAL(DA$(5)): IF OK=1 THEN
     {SPACE}RETURN                             :rem 1Ø
324Ø GOSUB 334Ø: H1$=DA$(1): IF OK=1 THEN RETURN
                                               :rem 5Ø
325Ø GOSUB 334Ø: H2$=DA$(1): JJ=1: IF OK=1 THEN RE
     TURN                                      :rem 112
326Ø : GOSUB 334Ø: FB(JJ)=VAL(DA$(1)): NF(JJ)=VAL(
     DA$(2)): FJ$(JJ)=DA$(3)                   :rem 31
327Ø FC(JJ)=VAL(DA$(4)): LH(JJ)=VAL(DA$(5)): MF$(J
     J)=DA$(6)                                 :rem 111
328Ø IF OK=1 THEN RETURN                      :rem 115
329Ø GOSUB 334Ø: FH$(JJ)=DA$(1): IF OK=1 THEN RETU
     RN                                        :rem 49
33ØØ JJ=JJ+1: IF JJ<=CC THEN 326Ø             :rem 221
331Ø INPUT# 8,DA$(Ø): SYS SP: D1=VAL(DA$(1)): DM%=
     VAL(DA$(2))                               :rem 188
332Ø CLOSE 8: GOSUB 287Ø: OK=Ø: RETURN        :rem 92
333Ø :                                         :rem 3
334Ø : INPUT# 8,DA$(Ø): SYS SP: IF ST=Ø THEN RETUR
     N                                         :rem 13
335Ø CLOSE 8: PRINT Y1$;"{RVS} BAD FILE DATA {OFF}
     ";YØ$: OK=1: RETURN                       :rem 86
336Ø :                                         :rem 6
337Ø REM"--SUB--DEFINE OR MODIFY PRINT FORMAT--"
                                               :rem 87
```

```
3380 :                                          :rem 8
3390 : PRINT "{DOWN}{RVS} DEFINE THE PRINTING FORM
     AT {OFF}{DOWN}": GOTO 3410              :rem 155
3400 : PRINT "{DOWN}{RVS} MODIFY THE PRINTING FORM
     AT {OFE}{DOWN}"                          :rem 117
3410 : PRINT "#BLANK LINES ABOVE HEADING";: K2=B1:
      GOSUB 3140                               :rem 90
3420 IF K2<0 OR K2>58 THEN 3410                :rem 1
3430 B1=K2: PRINT "{DOWN}{RVS} ENTER PAGE HEADING
     {SPACE}LINE {OFF}"                       :rem 106
3440 PRINT "{DOWN}USE TWO ENTRY LINES TO FORM A"
                                              :rem 185
3450 PRINT "COMPLETE PAGE HEADING LINE."       :rem 89
3460 PRINT "{DOWN}USE '<>' TO SHOW PAGE NUMBER LOC
     ATION"                                   :rem 135
3470 PRINT "{DOWN}DON'T DISTURB THE QUOTE OR THE E
     ND OF"                                   :rem 134
3480 PRINT "LINE MARKER.{DOWN}"               :rem 201
3490 : PRINT " ";H1$;SPC(36-LEN(H1$));"{RVS}";LM$;
     CU$;QT$                                   :rem 12
3500 PRINT "{UP}";: INPUT# 1,H1$: PRINT        :rem 19
3510 IF LEN(H1$)<37 OR LEN(H1$)=37 AND RIGHT$(H1$,
     1)=LM$ THEN 3530                         :rem 148
3520 H1$=LEFT$(H1$,36): PRINT "{RVS} TOO LONG !
     {OFF}": GOTO 3490                        :rem 171
3530 : H1$=LEFT$(H1$,36)                      :rem 177
3540 : PRINT " ";H2$;SPC(36-LEN(H2$));"{RVS}";LM$;
     CU$;QT$                                   :rem 10
3550 PRINT "{UP}";: INPUT# 1,H2$: PRINT        :rem 25
3560 IF LEN(H2$)<37 OR LEN(H2$)=37 AND RIGHT$(H2$,
     1)=LM$ THEN 3580                         :rem 161
3570 H2$=LEFT$(H2$,36): PRINT "{RVS} TOO LONG !
     {OFF}": GOTO 3540                        :rem 174
3580 : H2$=LEFT$(H2$,36)                      :rem 184
3590 GOSUB 2870: IF HE=1 THEN 3630            :rem 225
3600 : PRINT "{DOWN}STARTING PAGE #";: K2=BP: GOSU
     B 3140                                   :rem 212
3610 IF K2<0 OR K2>9000 THEN 3600              :rem 95
3620 BP=K2: PN=BP                             :rem 190
3630 : PRINT "#BLANK LINES BELOW HEADING";: K2=B2:
      GOSUB 3140                              :rem 107
3640 IF K2<0 OR K2+B1>58 THEN 3630            :rem 167
3650 :                                          :rem 8
3660 B2=K2: JL=0: CL=0: REM"--INDIVIDUAL FIELD INF
     ORMATION --"                              :rem 25
3670 : PRINT "{DOWN}# OF PRINT FIELDS";: K2=CC: GO
     SUB 3140                                  :rem 31
3680 IF K2<1 OR K2>20 THEN 3670                 :rem 7
3690 CC=K2: FOR JJ=1 TO CC: MF$(JJ)="Y": NEXT JJ
                                               :rem 27
```

```
3700 FOR JJ=CC TO 20: MF$(JJ)="N": NEXT JJ:rem 191
3710 : JL=JL+1                              :rem 202
3720 : PRINT "{DOWN}{RVS}{5 SPACES}FOR PRINT FIELD
     {4 SPACES}#";JL;"{LEFT}{RVS} {OFF}"  :rem 134
3730 : PRINT "# SPACES AHEAD OF FIELD";: K2=FB(JL)
     : GOSUB 3140                            :rem 38
3740 IF K2<0 OR K2>78 THEN 3730              :rem 13
3750 FB(JL)=K2: CL=CL+FB(JL)                 :rem 39
3760 : PRINT "PRINT DATA FIELD #";: K2=NF(JL): GOS
     UB 3140                                 :rem 53
3770 IF K2<1 OR K2>20 THEN 3760               :rem 7
3780 NF(JL)=K2                                :rem 7
3790 : PRINT "LEFT OR RIGHT JUSTIFIED";: K1$=FJ$(J
     L): GOSUB 3090                          :rem 66
3800 IF K1$<>"L" AND K1$<>"R" THEN 3790     :rem 137
3810 FJ$(JL)=K1$                             :rem 68
3820 : PRINT "#OF COLUMNS IN FIELD";: K2=FC(JL): G
     OSUB 3140                              :rem 205
3830 IF K2<1 OR K2>78 THEN 3820              :rem 14
3840 FC(JL)=K2: CL=CL+FC(JL)                 :rem 41
3850 PRINT "FIELD HEADING": IF FC(JL)<39 THEN 3890
                                            :rem 217
3860 PRINT QT$;LEFT$(FH$(JL),37);QT$;CU$;: INPUT#
     {SPACE}1,KB$: PRINT                     :rem 41
3870 PRINT QT$;MID$(FH$(JL),38);QT$;CU$;: INPUT# 1
     ,FH$(JL): PRINT                        :rem 194
3880 FH$(JL)=KB$+FH$(JL): GOTO 3900          :rem 93
3890 : PRINT QT$;LEFT$(FH$(JL),37);QT$;CU$;: INPUT
     # 1,FH$(JL): PRINT                      :rem 78
3900 : LH(JL)=FC(JL)-LEN(FH$(JL)): IF LH(JL)=>0 TH
     EN 3940                                 :rem 18
3910 PRINT "{DOWN}FIELD HEADING IS {RVS}";-LH(JL);
     "{OFF} CHR'S TOO LONG"                  :rem 30
3920 PRINT "RE-ENTER DATA FOR THIS FIELD{DOWN}"
                                            :rem 143
3930 CL=CL-FB(JL)-FC(JL): GOTO 3720         :rem 162
3940 : PRINT "{DOWN}NOW AT COLUMN{2 SPACES}#";CL
                                             :rem 48
3950 PRINT "{6 SPACES}MORE FIELDS";: KB$=MF$(JL)
                                            :rem 141
3960 GOSUB 2730: MF$(JL)=KB$: IF KB$="Y" THEN 3710
                                             :rem 50
3970 : CC=JL: PRINT "{DOWN}#BLANK LINES ABOVE DATA
     ";: K2=B3: GOSUB 3140                   :rem 53
3980 IF K2<0 OR K2+B1+B2>58 THEN 3970        :rem 84
3990 B3=K2: D2=59-B1-B2-B3: IF D1>D2 THEN D1=D2
                                            :rem 139
4000 : PRINT "DATA LINES/PAGE (MAX";D2;")";: K2=D1
     : GOSUB 3140                            :rem 61
4010 IF K2<1 OR K2>D2 THEN 4000               :rem 3
```

```
4020 D1=K2: DM%=2+B1+B2+B3+D1: RETURN      :rem 235
4030 :                                        :rem 1
4040 REM"======= START OF MAIN PROGRAM =========="
                                             :rem 93
4050 :                                        :rem 3
4060 REM"---INITIALIZATION--"                :rem 251
4070 :                                        :rem 5
4080 : DIM FB(20),NF(20),FJ$(20),FC(20),FH$(20),LH
     (20),OC(20),OL(20)                      :rem 120
4090 DIM DA$(21),MF$(20)                     :rem 154
4100 OPEN 1,0: OPEN 3,3: OPEN 4,4: OPEN 15,8,15
                                             :rem 74
4110 PRINT# 4,"{HOME}";: GOSUB 2800          :rem 236
4120 BL$="{63 SPACES}"                       :rem 250
4130 LC=59468: IF TY=6 THEN LC=53272         :rem 139
4140 CA$="": IF (PEEK (LC) AND 2)<>0 THEN CA$="
     {DOWN}"                                 :rem 41
4150 CR$=CHR$(13): CU$=CR$+"{UP}": QT$=CHR$(34)
                                             :rem 91
4160 LM$=CHR$(160): CK$=CHR$(122): ZR$=CHR$(0): BF
     $=ZR$+CHR$(16)                          :rem 94
4170 DV=3: S1=22: S2=27: FA$="DA": FT%=0: SP=30979
     : SW=30982                              :rem 56
4180 DI$="0": FF$="FM-": FW$="WP-": DO$="1": IF TY
     =6 THEN DO$="0"                         :rem 205
4190 M1=2: M4=2: CH$="YOUR CHOICE -----"     :rem 42
4200 DC$="{RVS} DISK FULL ": IF TY<>6 THEN DC$=DC$
     +"IN DRIVE #"+DO$                       :rem 207
4210 :                                        :rem 1
4220 PRINT "{CLR}{DOWN}-------------- N O T E S --
     ------------"                           :rem 178
4230 IF TY=6 THEN 4280                       :rem 117
4240 PRINT "{DOWN} ALL SOURCE DATA AND PRINT FORMA
     T FILES"                                :rem 12
4250 PRINT " MUST BE IN{3 SPACES}{RVS} DRIVE #0
     {OFF}"                                  :rem 116
4260 PRINT "{DOWN} ANY WORDPRO OUTPUT FILES CREATE
     D WILL"                                 :rem 87
4270 PRINT " BE SAVED ON{2 SPACES}{RVS} DRIVE #1
     {OFF}"                                  :rem 167
4280 : PRINT "{2 DOWN} OUTPUT OPERATIONS CAN BE:"
                                             :rem 131
4290 PRINT "{DOWN}{2 SPACES}FROZEN BY PRESSING ANY
      KEY"                                   :rem 92
4300 PRINT "{2 SPACES}-THEN-";CR$;"{2 SPACES}ABORT
     ED BY PRESSING {RVS} Q {OFF}"           :rem 144
4310 PRINT "{2 SPACES}-OR-";CR$;"{2 SPACES}RESUMED
      BY PRESSING ANY OTHER KEY."            :rem 163
4320 PRINT "{3 DOWN}SET PRINTER TO TOP-OF-FORM AND
     "                                       :rem 105
```

```
4330 :                                              :rem 4
4340 REM"---FORMAT SOURCE MENU--                  :rem 129
4350 :                                              :rem 6
4360 : GOSUB 2650                                  :rem 84
4370 : PRINT "{CLR}{DOWN}{RVS}{5 SPACES}F O R M A
     {SPACE}T{4 SPACES}S O U R C E S{5 SPACES}
     {OFF}"                                        :rem 14
4380 PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CHANGE SC
     REEN/PRINTER CASE"                            :rem 91
4390 PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}LOAD FORM
     AT FILE FROM DISK"                           :rem 240
4400 PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}DEFINE TH
     E PRINTING FORMAT"                            :rem 65
4410 PRINT "{2 DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR
      GO TO MASTER MENU{DOWN}"                    :rem 200
4420 : PRINT CH$;: K2=M1: GOSUB 3140              :rem 111
4430 IF K2=1 THEN GOSUB 2950                      :rem 196
4440 IF K2=2 THEN 4490                             :rem 71
4450 IF K2=3 THEN GOSUB 3390: M2=4: GOTO 4560
                                                   :rem 51
4460 IF K2=9 THEN 5980                             :rem 85
4470 PRINT "{UP}";: GOTO 4420                     :rem 169
4480 :                                             :rem 10
4490 : GOSUB 3190: IF OK=1 THEN M1=2: GOTO 4360
                                                  :rem 133
4500 IF OK=2 THEN M1=2: GOTO 4370                 :rem 190
4510 M2=4: GOTO 4560                              :rem 252
4520 :                                              :rem 5
4530 REM"---SETUP OPTIONS MENU--"                 :rem 199
4540 :                                              :rem 7
4550 : GOSUB 2650                                  :rem 85
4560 : PRINT "{CLR}{DOWN}{RVS}{5 SPACES}S E T U P
     {4 SPACES}O P T I O N S{7 SPACES}{OFF}"
                                                  :rem 223
4570 PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CHANGE SC
     REEN/PRINTER CASE"                            :rem 92
4580 PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}LOAD FORM
     AT FILE FROM DISK"                           :rem 241
4590 PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}MODIFY TH
     E PRINTING FORMAT"                           :rem 104
4600 PRINT "{DOWN}{RVS} 4 {OFF}{2 SPACES}TEST HEAD
     INGS TO SCREEN"                              :rem 106
4610 PRINT "{DOWN}{RVS} 5 {OFF}{2 SPACES}TEST HEAD
     INGS TO PRINTER"                             :rem 208
4620 PRINT "{DOWN}{RVS} 6 {OFF}{2 SPACES}SAVE PRIN
     T FORMAT FILE"                                :rem 45
4630 PRINT "{2 DOWN}{RVS} 7 {OFF}{2 SPACES}OUTPUT
     {SPACE}OPTIONS"                              :rem 184
4640 PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR G
     O TO MASTER MENU{DOWN}"                      :rem 188
```

```
4650 : PRINT CH$;: K2=M2: GOSUB 3140        :rem 117
4660 IF K2=1 THEN M2=4: GOSUB 2950          :rem 243
4670 IF K2=2 THEN 4760                       :rem 76
4680 IF K2=3 THEN M2=4: GOSUB 3400: GOTO 4560
                                             :rem 48
4690 IF K2=4 THEN M2=7: SE%=1: DV=3: GOSUB 1580: G
     OTO 4550                               :rem 228
4700 IF K2=5 THEN M2=7: SE%=1: DV=4: GOSUB 1580: G
     OTO 4560                               :rem 223
4710 IF K2=6 THEN M2=7: GOTO 5060           :rem 171
4720 IF K2=7 THEN M3=3: GOTO 4840           :rem 175
4730 IF K2=9 THEN 5960                       :rem 83
4740 PRINT "{UP}";: GOTO 4650               :rem 174
4750 :                                       :rem 10
4760 : GOSUB 3190: IF OK=1 THEN M2=2: GOTO 4550
                                            :rem 135
4770 IF OK=2 THEN M2=2: GOTO 4560           :rem 201
4780 M2=4: GOTO 4560                          :rem 5
4790 :                                       :rem 14
4800 :                                        :rem 6
4810 REM"---OUTPUT OPTIONS MENU--"           :rem 40
4820 :                                        :rem 8
4830 : GOSUB 2650                            :rem 86
4840 : PRINT "{CLR}{DOWN}{RVS}{5 SPACES}O U T P U
     {SPACE}T{4 SPACES}O P T I O N S{5 SPACES}
     {OFF}"                                  :rem 64
4850 PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CHANGE SC
     REEN/PRINTER CASE"                      :rem 93
4860 PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}SCREEN OU
     TPUT ONLY"                             :rem 125
4870 PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}PRINTER O
     UTPUT ONLY"                            :rem 227
4880 PRINT "{DOWN}{RVS} 4 {OFF}{2 SPACES}WORDPRO F
     ILES ONLY"                             :rem 112
4890 PRINT "{DOWN}{RVS} 5 {OFF}{2 SPACES}WORDPRO A
     ND PRINTER"                            :rem 180
4900 PRINT "{DOWN}{RVS} 6 {OFF}{2 SPACES}WORDPRO A
     ND SCREEN"                              :rem 73
4910 PRINT "{2 DOWN}{RVS} 7 {OFF}{2 SPACES}RETURN
     {SPACE}TO SETUP OPTIONS"               :rem 220
4920 PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR G
     O TO MASTER MENU{DOWN}"                :rem 189
4930 : PRINT CH$;: K2=M3: GOSUB 3140        :rem 119
4940 IF K2=1 THEN GOSUB 2950: GOTO 5020       :rem 4
4950 IF K2=2 THEN M3=2: DV=3: SE%=1: GOTO 5250
                                             :rem 83
4960 IF K2=3 THEN M3=3: DV=4: SE%=1: GOTO 5250
                                             :rem 87
4970 IF K2=4 THEN M3=4: DV=3: SE%=3: GOTO 5250
                                             :rem 91
```

```
4980 IF K2=5 THEN M3=5: DV=4: SE%=2: GOTO 5250
                                              :rem 94
4990 IF K2=6 THEN M3=6: DV=3: SE%=2: GOTO 5250
                                              :rem 96
5000 IF K2=7 THEN M2=3: GOTO 4560            :rem 165
5010 IF K2=9 THEN 5960                        :rem 75
5020 : PRINT "{UP}";: GOTO 4930              :rem 225
5030 :                                         :rem 2
5040 REM"---SAVE THE FORMAT DATA FILE--"     :rem 229
5050 :                                         :rem 4
5060 : PRINT "{DOWN}{RVS} READY TO SAVE PRINT FORM
     AT FILE {OFF}{DOWN}"                     :rem 89
5070 DA$(0)=HA$+HB$: FOR JJ=1 TO CC: DA$(0)=DA$(0)
     +FH$(JJ): NEXT JJ                       :rem 159
5080 : PRINT "DELIMITER TO BE USED";          :rem 13
5090 K1$=FD$: GOSUB 3090: FD$=K1$: GOSUB 2090: IF
     {SPACE}EF<>0 THEN 5080                   :rem 35
5100 SYS SP: IF FT%<2 THEN 5120               :rem 85
5110 PRINT "{RVS} CHARACTER IS USED IN HEADINGS
     {OFF}": GOTO 5080                       :rem 176
5120 : NA$=FF$: LN=16: GOSUB 2400: FF$=NA$:rem 116
5130 DR$=DI$: GOSUB 2280: IF EN<>0 THEN GOSUB 2460
     : M2=6: GOTO 4560                       :rem 230
5140 PRINT# 9,QT$;FD$;"←@0:";FF$;CR$;         :rem 51
5150 PRINT# 9,QT$;CC;FD$;PN;FD$;B1;FD$;B2;FD$;B3;F
     D$;CR$;                                  :rem 12
5160 PRINT# 9,QT$;H1$;FD$;CR$;QT$;H2$;FD$;CR$;: JJ
     =0                                      :rem 143
5170 : JJ=JJ+1: IF JJ>CC THEN 5210           :rem 224
5180 PRINT# 9,QT$;FB(JJ);FD$;NF(JJ);FD$;FJ$(JJ);
                                             :rem 233
5190 PRINT# 9,FD$;FC(JJ);FD$;LH(JJ);FD$;MF$(JJ);FD
     $;CR$;                                  :rem 176
5200 PRINT# 9,QT$;FH$(JJ);FD$;CR$;: GOTO 5170
                                             :rem 207
5210 : PRINT# 9,QT$;D1;FD$;DM%;FD$;CR$;: CLOSE 9:
     {SPACE}GOTO 4560                        :rem 173
5220 :                                         :rem 3
5230 REM"---MAIN PRINT ROUTINE--"            :rem 171
5240 :                                         :rem 5
5250 : TC%=SE%: SE%=1: GOSUB 1460: SE%=TC%: IF DV=
     3 THEN 5290                             :rem 104
5260 PRINT "IS PRINTER AT TOP OF FORM";      :rem 234
5270 GOSUB 2720: IF KB$="Y" THEN 5290        :rem 108
5280 PRINT "SET PRINTER AND THEN": GOSUB 2650
                                              :rem 57
5290 : IF SE%<2 THEN 5380                    :rem 195
5300 PRINT "{DOWN}{RVS} FOR WORDPRO FILES {OFF}":
     {SPACE}PC=0: GOSUB 2030                 :rem 142
5310 NA$=FW$: LN=12: GOSUB 2400: FW$=NA$      :rem 89
```

```
5320 GOSUB 256Ø: IF OK<>Ø THEN 483Ø         :rem 34
533Ø IF SF>24 THEN 536Ø                      :rem 148
534Ø IF TY<>6 THEN PRINT DC$: GOSUB 246Ø: GOTO 484
     Ø                                       :rem 232
535Ø GOTO 593Ø                               :rem 215
536Ø : DR$=DO$: GOSUB 229Ø: IF EN<>Ø THEN GOSUB 24
     6Ø: GOTO 484Ø                            :rem 1
537Ø GOSUB 189Ø: GOSUB 19ØØ                  :rem 165
538Ø : PRINT "{DOWN}{RVS} READY FOR FIRST DATA FIL
     E {OFF}"                                :rem 174
539Ø : PRINT : NA$=FI$: LN=16: GOSUB 24ØØ: FI$=NA$
                                              :rem 74
54ØØ GOSUB 219Ø: IF ER<>Ø THEN GOSUB 246Ø: CLOSE 9
     : SE%=1: GOTO 484Ø                       :rem 43
541Ø IF NB=Ø THEN GOSUB 158Ø                 :rem 211
542Ø NB=Ø: GOSUB 127Ø: CLOSE 8                :rem 52
543Ø IF KB$="Q" THEN PRINT "{RVS} SOURCE FILE CLOS
     ED {OFF}{DOWN}": GOTO 548Ø              :rem 13Ø
544Ø PRINT "{RVS}{2 SPACES}END OF SOURCE FILE
     {OFF}{DOWN}"                            :rem 176
545Ø :                                         :rem 8
546Ø REM"---CONTINUATION OPTIONS MENU--"     :rem 212
547Ø :                                        :rem 1Ø
548Ø : GOSUB 265Ø: PRINT "{CLR}{DOWN}{RVS}
     {3 SPACES}C O N T I N U E{4 SPACES}O P T I O
     {SPACE}N S{3 SPACES}{OFF}"               :rem 6Ø
549Ø PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}CHANGE SC
     REEN/PRINTER CASE"                       :rem 94
55ØØ PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}CONTINUE
     {SPACE}-- NO PAGE BREAK"                 :rem 96
551Ø PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}CONTINUE
     {SPACE}-- AT TOP OF PAGE"               :rem 125
552Ø PRINT "{DOWN}{RVS} 4 {OFF}{2 SPACES}CHANGE PR
     INTING SETUP"                            :rem 4Ø
553Ø PRINT "{2 DOWN}{RVS} 5 {OFF}{2 SPACES}PRINT O
     PERATIONS SUMMARY"                       :rem 88
554Ø PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR G
     O TO MASTER MENU{DOWN}"                 :rem 188
555Ø : PRINT CH$;: K2=M4: GOSUB 314Ø         :rem 119
556Ø IF K2=1 THEN GOSUB 295Ø                 :rem 2Ø1
557Ø IF K2=2 THEN M4=2: NB=1: GOTO 539Ø      :rem 231
558Ø IF K2=3 THEN M4=3: NB=Ø: GOSUB 146Ø: PN=PN+1:
      GOTO 564Ø                              :rem 123
559Ø IF K2=4 THEN M2=3: GOSUB 183Ø: PN=BP: GOTO 45
     6Ø                                      :rem 221
56ØØ IF K2=5 THEN M2=2: GOTO 568Ø            :rem 172
561Ø IF K2=9 THEN 596Ø                        :rem 81
562Ø PRINT "{UP}";: GOTO 555Ø                :rem 172
563Ø :                                         :rem 8
```

```
564Ø : PRINT "NEXT PAGE #";: K2=PN: GOSUB 314Ø: PN
     =K2: GOTO 539Ø                           :rem 126
565Ø :                                         :rem 1Ø
566Ø REM"---PRINT OPERATIONS SUMMARY--"       :rem 153
567Ø :                                         :rem 12
568Ø : GOSUB 183Ø: PN=PN+1: GOSUB 158Ø       :rem 24Ø
569Ø PRINT# DV,CR$;CR$;"NUMBER OF DATA LINES PRINT
     ED =";DL%;CR$;: DL%=Ø                    :rem 55
570Ø PRINT# DV,"BLANK LINES ABOVE HEADING
     {4 SPACES}=";B1;CR$;                     :rem 224
571Ø PRINT# DV,"BLANK LINES BELOW HEADING
     {4 SPACES}=";B2;CR$;                     :rem 238
572Ø PRINT# DV,"BLANK LINES ABOVE DATA{7 SPACES}="
     ;B3;CR$;                                 :rem 14
573Ø PRINT# DV,CR$;"PRINTER FORMAT WAS :";CR$;
                                              :rem 19Ø
574Ø PRINT# DV,CR$;"PRINT{9 SPACES}FILE{5 SPACES}L
     EADING{2 SPACES}FIELD{3 SPACES}";CR$;:rem 178
575Ø PRINT# DV,"FIELD#{2 SPACES}-IS-{2 SPACES}FIEL
     D#{3 SPACES}SPACES{3 SPACES}WIDTH{3 SPACES}JU
     STIFIED";CR$;                            :rem 164
576Ø FOR JJ=1 TO CC                           :rem 228
577Ø TE$=STR$(JJ): PRINT# DV,SPC(1);TE$;SPC(5-LEN(
     TE$));                                   :rem 219
578Ø PRINT# DV,"-------";SPC(2);              :rem 168
579Ø TE$=STR$(NF(JJ)): PRINT# DV,TE$;SPC(9-LEN(TE$
     ));                                      :rem 35
58ØØ TE$=STR$(FB(JJ)): PRINT# DV,TE$;SPC(9-LEN(TE$
     ));                                      :rem 15
581Ø TE$=STR$(FC(JJ)): PRINT# DV,TE$;SPC(1Ø-LEN(TE
     $));                                     :rem 57
582Ø TE$=FJ$(JJ): PRINT# DV,TE$;CR$;: NEXT JJ
                                              :rem 11
583Ø PRINT# DV,CR$;CR$;: IF DV<>3 THEN PL%=PL%+CC+
     13                                       :rem 1Ø8
584Ø FOR JJ=1 TO CC: IF OC(JJ)=Ø THEN 588Ø:rem 148
585Ø PRINT# DV,OC(JJ);" OVERRUNS IN FIELD";JJ;: OC
     (JJ)=Ø                                   :rem 229
586Ø PRINT# DV," -- LONGEST WAS";OL(JJ);"CHARACTER
     S";CR$;                                  :rem 55
587Ø IF DV<>3 THEN PL%=PL%+1                  :rem 244
588Ø : OL(JJ)=Ø: NEXT JJ: GOSUB 146Ø: IF DV=3 THEN
      455Ø                                    :rem 94
589Ø PN=BP: GOTO 456Ø                         :rem 133
59ØØ :                                         :rem 8
591Ø REM"---PROGRAM TERMINATION--"            :rem 58
592Ø :                                         :rem 1Ø
593Ø : PRINT Y1$;DC$;YØ$: PRINT "{DOWN}TRANSFER SO
     URCE DATA FILE TO A"                     :rem 132
```

```
594Ø PRINT "NEW DISKETTE AND RE-RUN THE PROGRAM.
     {DOWN}": GOTO 598Ø                      :rem 247
595Ø : PRINT Y1$;"{RVS} OVER 8Ø CHARACTERS IN DATA
      RECORD {OFF}";YØ$: GOTO 598Ø           :rem 144
596Ø : GOSUB 183Ø                             :rem 9Ø
597Ø :                                        :rem 15
598Ø : CLOSE 3: CLOSE 4: CLOSE 8: CLOSE 9: CLOSE 1
     4                                        :rem 131
599Ø : PRINT "{DOWN}PRESS {RVS} Q {OFF} TO QUIT OR
      --"                                     :rem 86
6ØØØ PRINT "ANY OTHER KEY FOR MASTER MENU":rem 21Ø
6Ø1Ø GOSUB 266Ø: IF KB$<>"Q" THEN 6Ø3Ø     :rem 15Ø
6Ø2Ø PRINT "{RVS} PROGRAM TERMINATED {OFF}";: CLOS
     E 1: CLOSE 15: END                       :rem 133
6Ø3Ø : DR$="Ø:": PS$="DFH BOOT": OPEN 8,8,8,+DR$+P
     S$+",P,R"                                :rem 8
6Ø4Ø GOSUB 3Ø1Ø: CLOSE 8: IF EN=Ø THEN 6Ø6Ø
                                              :rem 186
6Ø5Ø PRINT "{DOWN}TRYING TO LOAD {RVS} ";PS$;"
     {OFF}": GOSUB 246Ø: GOTO 599Ø           :rem 66
6Ø6Ø : CLOSE 1: CLOSE 15: PRINT "{DOWN}{RVS} LOADI
     NG ";PS$;" {OFF}"                        :rem 2
6Ø7Ø POKE 824,248: LOAD PS$,8                 :rem 233
```

DFH MERGE

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:DFH MERGE",8                :rem 1Ø4
1Ø1Ø :                                        :rem 252
1Ø2Ø REM" A DATA FILE HANDLER PROGRAM FOR MERGING
     {SPACE}PRE-SORTED"                       :rem 81
1Ø3Ø REM" SINGLE OR MULTI-FIELD SEQUENTIAL DATA FI
     LES."                                    :rem 33
1Ø4Ø :                                        :rem 255
1Ø7Ø :                                        :rem 2
1Ø8Ø REM"--ADJUST FOR COMPUTER TYPE--"        :rem 8
1Ø9Ø :                                        :rem 4
11ØØ IF PEEK (65534)=72 THEN 114Ø: REM"-- C-64 COM
     PUTER --"                                :rem 2ØØ
111Ø IF PEEK (824)<>248 THEN 118Ø: REM"-- NOT A PR
     OG'D LOAD --"                            :rem 13Ø
112Ø POKE 42,PEEK (2Ø1): POKE 43,PEEK (2Ø2): GOTO
     {SPACE}118Ø                              :rem 2Ø9
113Ø :                                        :rem 255
114Ø : POKE 5328Ø,14: POKE 53281,14: PRINT CHR$(31
     );: REM"--COLORS--"                      :rem 95
115Ø IF PEEK (824)<>248 THEN 118Ø: REM"-- NOT A PR
     OG'D LOAD --"                            :rem 134
```

```
116Ø POKE 45,PEEK (174): POKE 46,PEEK (175)
                                          :rem 176
117Ø :                                      :rem 3
118Ø : CLR : POKE 824,Ø: TY=2: YØ$="": Y1$=""
                                          :rem 111
119Ø IF PEEK (65534)=72 THEN TY=6: YØ$=CHR$(31): Y
     1$=CHR$(158)                         :rem 187
12ØØ GOTO 26ØØ                            :rem 196
121Ø :                                    :rem 254
122Ø REM"======START OF SUBROUTINES======" :rem 79
123Ø :                                      :rem Ø
124Ø REM"--SUB--LOAD RECORDS--"            :rem 25
125Ø :                                      :rem 2
126Ø : INPUT# 8,DA$(DP): TT=ST: CC=CC+LEN(DA$(DP))
                                          :rem 218
127Ø DA$(DP)=DA$(DP)+TG$: RC=RC+1: DP=DP+1  :rem 5
128Ø PRINT "{UP}";RC+DC%(LF): IF TT<>Ø THEN RETURN
                                          :rem 225
129Ø IF CC<CS AND RC<RS% THEN 126Ø         :rem 25
13ØØ RETURN                               :rem 164
131Ø :                                    :rem 255
132Ø REM"--SUB--DATA OUTPUT--"            :rem 241
133Ø :                                      :rem 1
134Ø : PRINT TAB(7);"{DOWN}RECORDS OUT TO {RVS} ";
     NA$;OF$;" {OFF}"                      :rem 48
135Ø : JO=JO+1                            :rem 206
136Ø LE=LEN(DA$(JO))-2: PRINT# 9,QT$;LEFT$(DA$(JO)
     ,LE);CR$;                             :rem 44
137Ø LF=VAL(RIGHT$(DA$(JO),2)): DL%(LF)=DL%(LF)-1
                                          :rem 183
138Ø DA$(JO)="": NC=NC+LE: NR=NR+1: PRINT "{UP}";N
     R                                    :rem 191
139Ø IF DL%(LF)<1 THEN 147Ø                :rem 92
14ØØ :                                    :rem 255
141Ø : IF NC>CO OR NR=>RO THEN CP=2: RETURN
                                          :rem 152
142Ø IF JO<FT% THEN 135Ø                  :rem 226
143Ø PRINT "{DOWN}{RVS} END OF VALID SORTED DATA.
     {2 SPACES}NOW USING {OFF}"           :rem 189
144Ø PRINT "{RVS} FIELD ZERO SORT FOR REST OF RECO
     RDS. {OFF}"                           :rem 36
145Ø SF=Ø: FS%=SF: SYS SS: JO=-1: GOTO 134Ø:rem 38
146Ø :                                      :rem 5
147Ø : IF DD%(LF)>Ø THEN CP=1: RETURN     :rem 221
148Ø FOR JJ=Ø TO CT: IF DL%(JJ)>Ø THEN JJ=CT+3
                                          :rem 181
149Ø NEXT JJ: IF JJ>CT+2 THEN 141Ø         :rem 37
15ØØ CP=8: RETURN                         :rem 232
151Ø :                                      :rem 1
152Ø REM"--SUB--WAIT FOR OPERATOR--"      :rem 112
```

```
153Ø :                                            :rem 3
154Ø : PRINT "PRESS ANY KEY TO CONTINUE{DOWN}"
                                                 :rem 76
155Ø : GET KB$: IF KB$<>"" THEN 155Ø            :rem 2ØØ
156Ø : GET KB$: IF KB$="" THEN 156Ø             :rem 141
157Ø RETURN                                     :rem 173
158Ø :                                            :rem 8
159Ø REM"--SUB--WAIT FOR YES OR NO--"            :rem 58
16ØØ :                                            :rem 1
161Ø : KB$="Y": GOTO 163Ø                       :rem 2Ø2
162Ø : KB$="N"                                  :rem 131
163Ø : PRINT CU$;SPC(S1);"? ";KB$;CU$;SPC(S1+2);
                                                :rem 114
164Ø : INPUT# 1,KB$: PRINT : KB$=LEFT$(KB$,1)
                                                 :rem 99
165Ø IF KB$="Y" OR KB$="N" THEN RETURN           :rem 22
166Ø PRINT "{RVS} Y {OFF} YES OR {RVS} N {OFF} NO
     {SPACE}? ";: GOTO 164Ø                     :rem 116
167Ø :                                            :rem 8
168Ø REM"--SUB--TEST/PRINT DISK ERROR--"        :rem 16Ø
169Ø :                                           :rem 1Ø
17ØØ : EB=1                                     :rem 247
171Ø : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN=63 TH
     EN 173Ø                                    :rem 227
172Ø : IF EB=Ø THEN PRINT Y1$;"{DOWN}{RVS} DISK ER
     ROR {OFF}";YØ$;CR$;EN;EM$;ET;ES             :rem 28
173Ø : EB=Ø: RETURN                              :rem 19
174Ø :                                            :rem 6
175Ø REM"--SUB--STRING INPUT--"                  :rem 84
176Ø :                                            :rem 8
177Ø : PRINT CU$;SPC(S1);"? ";K1$;CU$;SPC(S1+2);
                                                :rem 1Ø2
178Ø INPUT# 1,K1$: PRINT : RETURN                :rem 97
179Ø :                                           :rem 11
18ØØ REM"--SUB--NUMBER INPUT--"                  :rem 66
181Ø :                                            :rem 4
182Ø : PRINT CU$;SPC(S1);"?";K2;CU$;SPC(S1+2);
                                                 :rem 63
183Ø INPUT# 1,KB$: PRINT : K2=VAL(KB$): RETURN
                                                 :rem 71
184Ø :                                            :rem 7
185Ø REM"--SUB--TEST DELIMITER--"               :rem 2Ø5
186Ø :                                            :rem 9
187Ø : EF=Ø: IF FD$<>"" THEN 189Ø                :rem 56
188Ø PRINT "{RVS} ENCLOSE COMMA/COLON/SPACE IN QUO
     TES {OFF}": EF=1: RETURN                   :rem 198
189Ø : IF LEN(FD$)<>1 OR ASC(FD$)<32 THEN EF=1
                                                :rem 1ØØ
19ØØ IF ASC(FD$)>127 AND ASC(FD$)<161 THEN EF=1
                                                :rem 172
```

```
1910 IF FD$="0" OR VAL(FD$)<>0 THEN EF=1    :rem 14
1920 IF EF=1 THEN PRINT Y1$;"{DOWN}{RVS} ILLEGAL D
     ELIMITER{3 SPACES}{OFF}";Y0$;QT$;FD$;QT$
                                             :rem 100
1930 RETURN                                  :rem 173
1940 :                                         :rem 8
1950 REM"--SUB--DISK CHANGE--"               :rem 192
1960 :                                        :rem 10
1970 : PRINT "{DOWN}INSTALL {RVS} SOURCE DISKETTE
     {2 SPACES}#";DN$;" {OFF}": GOTO 1990 :rem 175
1980 : PRINT "{DOWN}INSTALL {RVS} OUTPUT DISKETTE
     {2 SPACES}#";DN$;" {OFF}"               :rem 138
1990 : IF TY<>6 THEN PRINT "{16 SPACES}IN DRIVE
     {2 SPACES}# ";DR$                        :rem 34
2000 PRINT "THEN, ";                          :rem 41
2010 : GOSUB 1540: PRINT# 15,"I";DR$          :rem 69
2020 GOSUB 1710: IF EN=0 THEN RETURN         :rem 229
2030 PRINT "CAN'T INITIALIZE -- TRY AGAIN.": GOTO
     {SPACE}2010                             :rem 242
2040 :                                         :rem 0
2050 REM"--SUB--OPEN SOURCE FILE--"           :rem 10
2060 :                                         :rem 2
2070 : OPEN 8,8,8,DI$+":"+DT$(LF)+",S,R": GOSUB 17
     00:                                     :rem 173
2080 IF EN=0 THEN 2120                        :rem 77
2090 IF EN<>62 THEN GOSUB 1720: PRINT "TRYING TO O
     PEN ";DT$(LF)                           :rem 223
2100 CLOSE 8: DN$=STR$(DD%(LF)): DR$=DI$: GOSUB 19
     70: GOTO 2070                            :rem 69
2110 :                                       :rem 254
2120 : INPUT# 8,X1$: IF FP=1 THEN 2210       :rem 132
2130 IF ST<>0 THEN PRINT Y1$;"{RVS} NO DATA {OFF}"
     ;Y0$: GOTO 2190                          :rem 17
2140 FD$=LEFT$(X1$,1): GOSUB 1870: IF EF<>0 THEN 2
     190                                     :rem 223
2150 :                                         :rem 2
2160 IF FF=0 THEN FF=1: TD$=FD$: PRINT : RETURN
                                              :rem 60
2170 IF TD$=FD$ THEN PRINT : RETURN          :rem 214
2180 PRINT Y1$;"{DOWN}{RVS} DIFFERENT DELIMITER
     {OFF}";Y0$;QT$;FD$;QT$                   :rem 73
2190 : PRINT "{RVS} IN FILE - {OFF}";QT$;DT$(LF);Q
     T$: GOTO 4220                           :rem 124
2200 :                                       :rem 254
2210 : PRINT TAB(7);"SPOOLING-UP IN {RVS} ";DT$(LF
     );" {OFF}": HC%=DC%(LF)/256             :rem 189
2220 POKE SD+3,HC%: POKE SD+2,DC%(LF)-HC%*256: SYS
      SU: RETURN                             :rem 125
2230 :                                         :rem 1
2240 REM"--SUB--LOAD FILE SEGMENT--"          :rem 59
```

```
225Ø :                                               :rem 3
226Ø : GOSUB 2Ø7Ø: CC=Ø: RC=Ø: TG$=MID$(STR$(LF),2
     )                                             :rem 153
227Ø IF LEN(TG$)<2 THEN TG$="Ø"+TG$                 :rem 64
228Ø PRINT TAB(7);"{UP}RECORDS{2 SPACES}FROM
     {2 SPACES}{RVS} ";DT$(LF);" {OFF}"             :rem 11
229Ø GOSUB 126Ø: CLOSE 8: IF TT=Ø THEN 231Ø
                                                   :rem 2Ø9
23ØØ PRINT "{UP}{RVS}";RC+DC%(LF);"{LEFT} {OFF}":
     {SPACE}DD%(LF)=Ø                              :rem 168
231Ø : DC%(LF)=DC%(LF)+RC: DL%(LF)=DL%(LF)+RC: RET
     URN                                           :rem 156
232Ø :                                               :rem 1
233Ø REM"--SUB--TEST #BLOCKS FREE--"                :rem 43
234Ø :                                               :rem 3
235Ø : OPEN 14,8,Ø,"$"+DR$+":": GOSUB 171Ø: IF EN=
     Ø THEN 238Ø                                   :rem 197
236Ø CLOSE 14: PRINT Y1$;"{RVS} CAN'T READ OUTPUT
     {SPACE}DISK DIRECTORY {OFF}";YØ$               :rem 96
237Ø GOSUB 198Ø: GOTO 235Ø                          :rem 91
238Ø : FOR JJ=1 TO 18: GET #14,X1$,X2$: NEXT JJ: C
     LOSE 14                                       :rem 115
239Ø BF%=ASC(X1$+ZR$)+ASC(X2$+ZR$)*256              :rem 75
24ØØ PRINT "{DOWN}";BF%;TAB(6);"BLOCKS FREE FOR ";
     NA$;OF$                                        :rem 81
241Ø ER=Ø: IF BF%>(TB+2*TR)/254/NF+3 THEN RETURN
                                                   :rem 141
242Ø PRINT Y1$;"{RVS} NOT ENOUGH BLOCKS FREE {OFF}
     ";YØ$: ER=1: RETURN                           :rem 254
243Ø :                                               :rem 3
244Ø REM"--SUB--OPEN/SETUP NEW OUTPUT FILE--"
                                                   :rem 215
245Ø :                                               :rem 5
246Ø : RE$=""                                       :rem 66
247Ø : OPEN 9,8,9,RE$+DR$+":"+NA$+OF$+",S,W"
                                                   :rem 227
248Ø GOSUB 171Ø: IF EN<>Ø THEN 25ØØ                 :rem 19
249Ø PRINT# 9,QT$;FD$"←@Ø:";NA$;OF$;CR$;: NR=Ø: NC
     =9+LEN(NA$): RETURN                           :rem 164
25ØØ : CLOSE 9: IF EN<>63 THEN RETURN              :rem 254
251Ø PRINT "{RVS} OUTPUT FILE {OFF}";NA$;OF$;"
     {RVS} EXISTS {OFF}"                            :rem 51
252Ø PRINT "WANT TO REPLACE IT";                    :rem 75
253Ø GOSUB 161Ø: IF KB$="Y" THEN RE$="@": GOTO 247
     Ø                                              :rem 81
254Ø RETURN                                        :rem 171
255Ø :                                               :rem 6
256Ø REM"====== START OF MAIN PROGRAM ====="
                                                   :rem 244
257Ø :                                               :rem 8
```

```
2580 REM"--INITIALIZE--"                          :rem 157
2590 :                                             :rem 10
2600 : RI=650: CI=13000: MR=650: MC=13000   :rem 24
2610 DIM DT$(50): REM"--DATA FILE TITLES--":rem 58
2620 DIM DD%(50): REM"--DATA FILE DISK #--"
                                                  :rem 165
2630 DIM DC%(50): REM"--DATA COUNT FOR SPOOL-UP--"
                                                    :rem 6
2640 DIM DL%(50): REM"--DATA LOADED COUNT--"
                                                  :rem 115
2650 DIM DA$(RI): REM"--DATA STORAGE ARRAY--"
                                                    :rem 0
2660 OPEN 1,0: OPEN 15,8,15                        :rem 85
2670 S1=22: SS=30976: SU=30985: SD=30991     :rem 86
2680 FA$="DA": FT%=0                              :rem 231
2690 CR$=CHR$(13): QT$=CHR$(34): CU$=CR$+"{UP}": Z
     R$=CHR$(0)                                    :rem 43
2700 PRINT "{CLR}{DOWN}{RVS} READY TO MERGE FILES
     {SPACE}{OFF}"                                :rem 221
2710 : FP=0: TN=0: OD=1: DI$="0": DO$="1": IF TY=6
      THEN DO$="0"                                 :rem 74
2720 RV=0: CT=0: TC=0: FOR JJ=0 TO 50: DT$(JJ)="":
      NEXT JJ                                     :rem 239
2730 :                                              :rem 6
2740 : FF=0: PRINT "{DOWN}HOW WERE THE SOURCE FILE
     S SORTED ?{DOWN}"                            :rem 131
2750 : PRINT "SORTED ON FIELD #";                   :rem 9
2760 K2=SF: GOSUB 1820: SF=K2: IF SF<0 OR SF>20 TH
     EN SF=0: GOTO 2750                            :rem 82
2770 : PRINT "{DOWN}{RVS} A {OFF} ASCENDING OR";CR
     $;"{RVS} D {OFF} DESCENDING ORDER";    :rem 228
2780 K1$=SO$: IF K1$="" THEN K1$="A"         :rem 239
2790 GOSUB 1770: SO$=K1$: IF SO$<>"A" AND SO$<>"D"
      THEN 2770                                    :rem 38
2800 :                                              :rem 4
2810 PRINT "{DOWN}{RVS} ENTER NAMES OF UP TO 50 SO
     URCE FILES {OFF}": K1$=""                     :rem 36
2820 : PRINT "{DOWN}SOURCE FILE #";TC+1;     :rem 101
2830 IF RV=1 THEN K1$=DT$(TC)                      :rem 34
2840 GOSUB 1770: DT$(TC)=K1$: IF TC<1 THEN 2890
                                                  :rem 167
2850 ER=0: FOR JJ=0 TO TC-1: IF K1$=DT$(JJ) THEN E
     R=1                                            :rem 8
2860 NEXT JJ                                      :rem 163
2870 IF ER=1 THEN PRINT Y1$;"{RVS} FILENAME ALREAD
     Y USED {OFF}";Y0$: GOTO 2820              :rem 141
2880 :                                             :rem 12
2890 : TC=TC+1                                    :rem 212
2900 IF TC>49 THEN PRINT "{DOWN}{RVS} ONLY 50 FILE
     S ALLOWED {OFF}": GOTO 2940             :rem 181
```

```
291Ø PRINT "ANY MORE FILES";: KB$="Y"       :rem 43
292Ø IF RV=1 AND TC>CT THEN KB$="N"         :rem 96
293Ø GOSUB 163Ø: IF KB$="Y" THEN 282Ø      :rem 1Ø3
294Ø : CT=TC-1: PRINT "{DOWN}ALL FILENAMES OK";: G
     OSUB 161Ø                             :rem 178
295Ø IF KB$="Y" THEN 3Ø2Ø                  :rem 222
296Ø : PRINT "{DOWN}RE-DEFINE THE MERGE";: GOSUB 1
     61Ø                                   :rem 1Ø6
297Ø IF KB$="Y" THEN RV=1: TC=Ø: GOTO 274Ø:rem 175
298Ø GOTO 427Ø                             :rem 217
299Ø :                                      :rem 14
3ØØØ REM"--DEFINE DRIVE USAGE--"            :rem 57
3Ø1Ø :                                      :rem 254
3Ø2Ø : RV=Ø: FC=Ø: FOR JJ=Ø TO CT: DD%(JJ)=Ø: NEXT
      JJ                                   :rem 23Ø
3Ø3Ø PRINT "{DOWN}{RVS} READY TO LOCATE FILES & CH
     ECK SIZES {OFF}{DOWN}"                :rem 178
3Ø4Ø IF TY=6 THEN 313Ø                     :rem 1Ø8
3Ø5Ø : PRINT "SOURCE FILES IN DRIVE";       :rem 99
3Ø6Ø K1$=DI$: GOSUB 177Ø: IF K1$<"Ø" OR K1$>"1" TH
     EN 3Ø5Ø                               :rem 228
3Ø7Ø DI$=K1$                                :rem 88
3Ø8Ø : PRINT "OUTPUT FILES TO DRIVE";       :rem 146
3Ø9Ø K1$=DO$: GOSUB 177Ø: IF K1$<"Ø" OR K1$>"1" TH
     EN 3Ø8Ø                               :rem 24Ø
31ØØ DO$=K1$: TY=2: IF DI$=DO$ THEN TY=1    :rem 1Ø2
311Ø REM"--FIND FILE LOCATIONS AND SIZES--"
                                           :rem 239
312Ø :                                       :rem Ø
313Ø : DK%=1: DR$=DI$                        :rem 5
314Ø : DN$=STR$(DK%): GOSUB 197Ø           :rem 162
315Ø DE=Ø: POKE SD+3,254: FOR JA=Ø TO CT    :rem 55
316Ø OPEN 8,8,8,"Ø:"+DT$(JA)+",S,R": GOSUB 17ØØ
                                           :rem 135
317Ø IF EN=62 THEN 327Ø                    :rem 141
318Ø IF EN<>Ø THEN GOSUB 172Ø: DE=1: JA=CT: GOTO 3
     27Ø                                    :rem 26
319Ø IF DD%(JA)=DK% THEN 327Ø              :rem 2Ø9
32ØØ IF DD%(JA)=Ø THEN INPUT# 8,X1$: SYS SU: GOTO
     {SPACE}324Ø                            :rem 9Ø
321Ø DE=2: PRINT "{2 DOWN}{RVS} FILE {OFF} ";DT$(J
     A)                                    :rem 13Ø
322Ø PRINT Y1$;"{RVS} FOUND ON MORE THAN ONE DISK
     {SPACE}{OFF}{DOWN}";YØ$: JA=CT: GOTO 327Ø
                                           :rem 122
323Ø :                                       :rem 2
324Ø : FR=PEEK (SD)+PEEK (SD+1)*256+1: FB=PEEK (SD
     +4)+PEEK (SD+5)*256-FR                 :rem 43
325Ø DD%(JA)=DK%: FC=FC+1: TR=TR+FR: TB=TB+FB
                                             :rem 5
```

```
326Ø PRINT "FOUND ";DT$(JA);TAB(2Ø);INT((FB+FR*2)/
     254)+1;TAB(25);"BLOCKS"                    :rem 76
327Ø : CLOSE 8: NEXT JA: IF DE=Ø THEN 331Ø:rem 1Ø9
328Ø IF DE=2 THEN 422Ø                          :rem 75
329Ø PRINT "{RVS}{2 SPACES}TRY AGAIN !{9 SPACES}
     {OFF}": GOTO 314Ø                         :rem 254
33ØØ :                                           :rem Ø
331Ø : IF FC>CT THEN PRINT "{RVS} ALL FILES LOCATE
     D {OFF}": GOTO 344Ø                        :rem 24
332Ø PRINT "{DOWN}{RVS} STILL LOOKING FOR: {OFF}"
                                                :rem 1Ø
333Ø FOR JJ=Ø TO CT: IF DD%(JJ)=Ø THEN PRINT DT$(J
     J)                                         :rem 16
334Ø NEXT JJ                                   :rem 157
335Ø IF TY<>1 THEN DK%=DK%+1: GOTO 314Ø        :rem 29
336Ø FOR JJ=Ø TO 5Ø: DD%(JJ)=Ø: NEXT JJ: FC=Ø: TR=
     Ø: TB=Ø                                   :rem 188
337Ø PRINT "{DOWN}{RVS} ALL FILES MUST BE ON THE S
     AME DISK {OFF}"                            :rem 62
338Ø PRINT "{RVS} FOR 'SINGLE DRIVE' OPERATIONS
     {6 SPACES}{OFF}{DOWN}"                    :rem 2Ø1
339Ø PRINT "NEED A NEW DISK";: GOSUB 161Ø: IF KB$=
     "Y" THEN 314Ø                              :rem 26
34ØØ GOTO 296Ø                                 :rem 2Ø9
341Ø :                                           :rem 2
342Ø REM"--MEMORY & DISK SPACE ALLOCATION--"
                                                :rem 33
343Ø :                                           :rem 4
344Ø : PRINT "{DOWN}MERGE INFORMATION SUMMARY:"
                                                 :rem 5
345Ø PRINT FC;TAB(8);"TOTAL DATA FILES"        :rem 13
346Ø PRINT INT(TR);TAB(8);"TOTAL RECORDS" :rem 236
347Ø PRINT INT((TB+TR*2)/254)+1;TAB(8);"TOTAL BLOC
     KS"                                        :rem 45
348Ø CS=CI/FC: RS%=RI/FC: RF%=TB/MC+1          :rem 95
349Ø IF TR/MR+1>RF% THEN RF%=TR/MR+1           :rem 35
35ØØ PRINT "{DOWN}{RVS} READY TO DEFINE OUTPUT FIL
     ES{8 SPACES}{OFF}"                        :rem 117
351Ø : PRINT "{DOWN}";RF%;"OUTPUT FILES ARE SUGGES
     TED."                                     :rem 113
352Ø : PRINT "HOW MANY DO YOU WANT";           :rem 253
353Ø K2=RF%: GOSUB 182Ø: IF K2<1 OR K2>99 THEN 352
     Ø                                          :rem 65
354Ø NF=K2: IF NF=>RF% THEN 358Ø              :rem 174
355Ø PRINT Y1$;"{DOWN}{RVS} WARNING {OFF}";YØ$;" F
     ILE(S) MAY BE TOO LARGE FOR"               :rem 47
356Ø PRINT "THE SORTING AND EDITING PROGRAM.{DOWN}
     "                                         :rem 212
357Ø PRINT "DO YOU WANT TO CONTINUE";: GOSUB 162Ø:
      IF KB$="N" THEN 351Ø                     :rem 113
```

```
3580 : RO=MR: CO=MC: IF NF<>RF% THEN CO=(TB+TR/2)/
     NF                                      :rem 142
3590 IF NF>RF% AND CO>MC THEN CO=MC          :rem 176
3600 IF NF<RF% THEN RO=99999                  :rem 15
3610 :                                         :rem 4
3620 REM"--GET OUTPUT FILE NAME--"           :rem 185
3630 :                                         :rem 6
3640 : PRINT "{DOWN}A SEQUENCE NUMBER WILL BE ADDE
     D TO"                                    :rem 0
3650 PRINT "EACH OUTPUT FILE NAME.{2 SPACES}WHAT N
     AME DO"                                 :rem 248
3660 PRINT "YOU WANT TO USE";                :rem 162
3670 K1$=NA$: GOSUB 1770: NA$=K1$            :rem 179
3680 IF LEN(NA$)<12 THEN 3710                :rem 220
3690 PRINT "{RVS} NAME TOO LONG {OFF}": NA$=LEFT$(
     NA$,11): GOTO 3640                      :rem 213
3700 :                                         :rem 4
3710 : IF TY=2 THEN 3810                     :rem 171
3720 NE=0: FOR JJ=0 TO CT                     :rem 40
3730 IF NA$+"."=LEFT$(DT$(JJ),LEN(NA$)+1) THEN NE=
     1                                       :rem 197
3740 NEXT JJ: IF NE=0 THEN 3810              :rem 101
3750 PRINT Y1$;"{DOWN}{RVS} FILENAME CONFLICT
     {OFF}";Y0$                              :rem 185
3760 PRINT "{DOWN}SOURCE FILENAME CAN'T BE RE-USED
      FOR"                                   :rem 116
3770 PRINT "OUTPUT WITH SINGLE DISK DRIVE{DOWN}":
     {SPACE}GOSUB 1540: GOTO 3640             :rem 11
3780 :                                        :rem 12
3790 REM"--LOAD & SORT INITIAL FILE SEGMENTS--"
                                             :rem 205
3800 :                                         :rem 5
3810 : DP=0: LF=0: PRINT "{DOWN}{RVS} READY TO LOA
     D INITIAL FILE SEGMENTS {OFF}{DOWN}" :rem 217
3820 FOR JJ=0 TO 50: DC%(JJ)=0: DL%(JJ)=0: NEXT JJ
                                              :rem 67
3830 : GOSUB 2260: IF LF<CT THEN LF=LF+1: GOTO 383
     0                                       :rem 173
3840 FP=1: FS%=SF: FO$=SO$: SYS SS           :rem 118
3850 IF FT%<1 THEN PRINT Y1$;"{RVS} NO DATA IN SOR
     T FIELD {OFF}";Y0$: GOTO 4220            :rem 53
3860 :                                        :rem 11
3870 REM"--MERGE PROCESS CONTROL--"           :rem 94
3880 :                                        :rem 13
3890 DN$=STR$(OD): DR$=DO$: JO=-1: IF TY=1 THEN 39
     20                                      :rem 255
3900 GOSUB 1980                               :rem 30
3910 :                                         :rem 7
3920 : TN=TN+1: OF$=".": IF TN<10 THEN OF$=OF$+"0"
                                              :rem 12
```

```
393Ø OF$=OF$+MID$(STR$(TN),2)                    :rem 1Ø2
394Ø :                                            :rem 1Ø
395Ø : GOSUB 235Ø: IF ER=1 THEN 398Ø              :rem 38
396Ø GOSUB 246Ø: IF EN=Ø THEN 4Ø4Ø               :rem 222
397Ø :                                            :rem 13
398Ø : IF TY=1 THEN 4Ø1Ø                         :rem 172
399Ø OD=OD+1: DN$=STR$(OD)                       :rem 194
4ØØØ : GOSUB 198Ø: GOTO 395Ø                     :rem 148
4Ø1Ø : PRINT "{DOWN}TRY OUTPUT AGAIN";: GOSUB 161Ø
     : IF KB$"Y" THEN 4ØØØ                       :rem 248
4Ø2Ø GOTO 422Ø                                   :rem 199
4Ø3Ø :                                            :rem 1
4Ø4Ø : IF TY<>6 THEN 411Ø                        :rem 227
4Ø5Ø CLOSE 9                                     :rem 12Ø
4Ø6Ø : OPEN 9,8,9,DR$+":"+NA$+OF$+",A": REM"--OPEN
      FOR APPEND--"                               :rem 76
4Ø7Ø GOSUB 17ØØ: IF EN=Ø THEN 411Ø               :rem 2Ø9
4Ø8Ø CLOSE 9: IF EN<>62 THEN GOSUB 172Ø           :rem 5Ø
4Ø9Ø GOSUB 198Ø: GOTO 4Ø6Ø                        :rem 92
41ØØ :                                           :rem 255
411Ø : GOSUB 134Ø: GOSUB 171Ø                    :rem 2Ø3
412Ø IF EN<>Ø THEN PRINT Y1$;"{DOWN}{RVS} FILE NOT
      SAVED CORRECTLY {OFF}";YØ$: GOTO 422Ø
                                                 :rem 143
413Ø IF CP=8 THEN 423Ø                            :rem 87
414Ø IF CP=2 THEN CLOSE 9: GOTO 392Ø             :rem 121
415Ø IF TY=6 THEN CLOSE 9                         :rem 87
416Ø FS%=Ø: FO$=SO$: SYS SS: DP=FT%: GOSUB 226Ø
                                                  :rem 25
417Ø FS%=SF: SYS SS: JO=-1: IF TY=6 THEN 4Ø6Ø
                                                 :rem 143
418Ø GOTO 411Ø                                   :rem 2Ø4
419Ø :                                            :rem 8
42ØØ REM"--PROGRAM TERMINATION--"                 :rem 4
421Ø :                                            :rem 1
422Ø : EN=Ø: PRINT "{DOWN}{RVS} PROGRAM OPERATION
     {SPACE}HALTED {OFF}": GOTO 424Ø             :rem 122
423Ø : PRINT "{DOWN}{RVS} MERGE COMPLETED {OFF}"
                                                 :rem 15Ø
424Ø : CLOSE 8: CLOSE 9: CLOSE 14: PRINT "{DOWN}MO
     RE FILES TO MERGE";                         :rem 192
425Ø GOSUB 161Ø: IF KB$="Y" THEN TB=Ø: TR=Ø: GOTO
     {SPACE}271Ø                                  :rem 35
426Ø CLOSE 1                                     :rem 115
427Ø : PRINT "{DOWN}PRESS {RVS} Q {OFF} TO QUIT OR
      --"                                         :rem 76
428Ø PRINT "ANY OTHER KEY FOR MASTER MENU":rem 218
429Ø GOSUB 155Ø: IF KB$<>"Q" THEN 431Ø          :rem 154
43ØØ PRINT "{RVS} PROGRAM TERMINATED{2 SPACES}
     {OFF}": CLOSE 1: CLOSE 15: END               :rem 73
```

```
431Ø : DR$="Ø": PS$="DFH BOOT": OPEN 8,8,8,"Ø:"+PS
     $+",P,R"                                 :rem 15Ø
432Ø GOSUB 171Ø: CLOSE 8: IF EN=Ø THEN 437Ø
                                              :rem 192
433Ø PRINT "{DOWN}TRYING TO LOAD {RVS} ";PS$;"
     {OFF}"                                   :rem 113
434Ø PRINT "INSTALL CORRECT DISK ";            :rem 43
435Ø IF TY<>6 THEN PRINT "IN DRIVE ";DR$; :rem 249
436Ø PRINT CR$;"THEN ---";: GOSUB 2Ø1Ø: GOTO 427Ø
                                              :rem 64
437Ø : CLOSE 1: CLOSE 15: PRINT "{DOWN}{RVS} LOADI
     NG ";PS$;" {OFF}"                          :rem 4
438Ø POKE 824,248: LOAD PS$,8                 :rem 235
```

DFH SWAP

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:DFH SWAP",8                  :rem 51
1Ø1Ø :                                        :rem 252
1Ø2Ø REM" A DATA FILE HANDLER PROGRAM TO RE-STRUCT
     URE"                                     :rem 18Ø
1Ø3Ø REM" MULTI-FIELD SEQUENTIAL DATA FILES."
                                              :rem 19Ø
1Ø4Ø :                                        :rem 255
1Ø7Ø :                                          :rem 2
1Ø8Ø REM"---SET TOP OF BASIC IF REQUIRED--"
                                              :rem 152
1Ø9Ø :                                          :rem 4
11ØØ IF PEEK (65534)=72 THEN 114Ø: REM"-- C-64 COM
     PUTER --"                                :rem 2ØØ
111Ø IF PEEK (824)<>248 THEN 118Ø: REM"-- NOT A PR
     OG'D LOAD --"                            :rem 13Ø
112Ø POKE 42,PEEK (2Ø1): POKE 43,PEEK (2Ø2): GOTO
     {SPACE}118Ø                              :rem 2Ø9
113Ø :                                        :rem 255
114Ø : POKE 5328Ø,14: POKE 53281,14: PRINT CHR$(31
     );: REM"-- COLORS --"                     :rem 95
115Ø IF PEEK (824)<>248 THEN 118Ø: REM"-- NOT A PR
     OG'D LOAD --"                            :rem 134
116Ø POKE 45,PEEK (174): POKE 46,PEEK (175)
                                              :rem 176
117Ø :                                          :rem 3
118Ø : CLR : POKE 824,Ø: TY=2: YØ$="": Y1$=""
                                              :rem 111
119Ø IF PEEK (65534)=72 THEN TY=6: YØ$=CHR$(31): Y
     1$=CHR$(158)                             :rem 187
12ØØ GOTO 233Ø                                :rem 196
121Ø :                                        :rem 254
```

```
1220 REM"=========== START OF SUBROUTINES ========
     =="                                      :rem 116
1230 :                                          :rem Ø
1240 REM"--SUB--INPUT, RE-STRUCTURE, AND OUTPUT DA
     TA LINES--"                              :rem 189
125Ø :                                          :rem 2
126Ø : RC=Ø                                     :rem 5
127Ø : PRINT TAB(7);"{DOWN}DATA RECORDS CONVERTED"
                                               :rem 86
128Ø : INPUT# 8,OD$(Ø): TT=ST: PR$=QT$: SYS SP: FO
     R JJ=1 TO NF                              :rem 86
129Ø ON FC(JJ) GOTO 131Ø,134Ø,133Ø,135Ø,137Ø
                                              :rem 168
13ØØ :                                        :rem 254
131Ø : IF FT%=>OP(JJ) THEN PR$=PR$+OD$(OP(JJ))
                                              :rem 251
132Ø GOTO 139Ø                                :rem 2Ø4
133Ø : IF FT%=>OP(JJ) THEN PR$=PR$+OD$(OP(JJ))
                                              :rem 253
134Ø : PR$=PR$+AD$(JJ): GOTO 139Ø             :rem 196
135Ø : PR$=PR$+AD$(JJ): IF FT%=>OP(JJ) THEN PR$=PR
     $+OD$(OP(JJ))                            :rem 187
136Ø GOTO 139Ø                                :rem 2Ø8
137Ø : IF FT%=>OP(JJ) THEN PR$=PR$+OD$(OP(JJ))
                                                :rem 1
138Ø IF FT%=>CP(JJ) THEN PR$=PR$+OD$(CP(JJ))
                                              :rem 176
139Ø : PR$=PR$+DE$: NEXT JJ: PRINT# 9,PR$;CR$;: RC
     =RC+1: PRINT "{UP}";RC                    :rem 98
14ØØ LP=LEN(PR$)-1                            :rem 242
141Ø IF LP>75 THEN PRINT Y1$;"{RVS} LONG RECORD
     {OFF}";YØ$;LP;"CHARACTERS"                :rem 53
142Ø IF TT<>Ø THEN RETURN                     :rem 183
143Ø IF LP>75 THEN 127Ø                       :rem 15Ø
144Ø GOTO 128Ø                                :rem 2Ø5
145Ø :                                          :rem 4
146Ø REM"--SUB--GET OLD DATA FIELD NUMBER--"
                                              :rem 241
147Ø :                                          :rem 6
148Ø : PRINT "{DOWN}WHICH OLD DATA FIELD";: K2=OP(
     NF): GOSUB 22ØØ                          :rem 234
149Ø IF K2<1 OR K2>2Ø THEN PRINT Y1$;"{RVS} ILLEGA
     L # {OFF}";YØ$: GOTO 148Ø                :rem 215
15ØØ OP(NF)=K2: RETURN                         :rem 3Ø
151Ø :                                          :rem 1
152Ø REM"--SUB--GET NEW FIXED DATA ENTRY--"
                                              :rem 2Ø6
153Ø :                                          :rem 3
154Ø : PRINT "{DOWN}ENTER NEW FIXED DATA";CR$;QT$;
     AD$(NF);CU$;                             :rem 2Ø3
```

```
155Ø INPUT# 1,AD$(NF): PRINT : RETURN       :rem 74
156Ø :                                       :rem 6
157Ø REM"--SUB--TEST DELIMITER--"           :rem 2Ø4
158Ø :                                       :rem 8
159Ø : EF=Ø: IF FD$<>"" THEN 161Ø           :rem 45
16ØØ PRINT "{RVS} ENCLOSE COMMA/COLON/SPACE IN QUO
     TES {OFF}": EF=1: RETURN               :rem 188
161Ø : IF LEN(FD$)<>1 OR ASC(FD$)<32 THEN EF=1
                                            :rem 9Ø
162Ø IF ASC(FD$)>127 AND ASC(FD$)<161 THEN EF=1
                                            :rem 171
163Ø IF FD$="Ø" OR VAL(FD$)<>Ø THEN EF=1    :rem 13
164Ø IF EF=1 THEN PRINT Y1$;"{RVS} ILLEGAL DELIMIT
     ER {OFF} ";YØ$;FD$                     :rem 74
165Ø RETURN                                 :rem 172
166Ø :                                       :rem 7
167Ø REM"--SUB--WAIT FOR OPERATOR--"        :rem 118
168Ø :                                       :rem 9
169Ø : PRINT "{DOWN}PRESS ANY KEY TO CONTINUE"
                                             :rem 82
17ØØ : GET KB$: IF KB$<>"" THEN 17ØØ        :rem 194
171Ø : GET KB$: IF KB$="" THEN 171Ø         :rem 135
172Ø RETURN                                 :rem 17Ø
173Ø :                                       :rem 5
174Ø REM"--SUB--WAIT FOR YES OR NO ANSWER--":rem 7
175Ø :                                       :rem 7
176Ø : KB$="Y": GOTO 178Ø                   :rem 214
177Ø : KB$="N"                              :rem 137
178Ø : PRINT CU$;SPC(S1);"? ";KB$;CU$;SPC(S1+2);
                                            :rem 12Ø
179Ø : INPUT# 1,KB$: PRINT : KB$=LEFT$(KB$,1)
                                            :rem 1Ø5
18ØØ IF KB$="Y" OR KB$="N" THEN RETURN      :rem 19
181Ø PRINT "{RVS} Y {OFF} YES OR {RVS} N {OFF} NO"
     ;CU$;SPC(S1)"? ";: GOTO 179Ø           :rem 168
182Ø :                                       :rem 5
183Ø REM"--SUB--DISK CHANGE / INITIALIZATION--"
                                             :rem 2Ø
184Ø :                                       :rem 7
185Ø : KB$="N": GOTO 187Ø                   :rem 2Ø3
186Ø : KB$="Y"                              :rem 148
187Ø : PRINT "{DOWN}NEED A NEW DISK";: GOSUB 178Ø
                                             :rem 35
188Ø IF KB$="N" THEN ER=1: RETURN           :rem 46
189Ø : IF TY=6 THEN DR$="Ø": GOTO 192Ø      :rem 148
19ØØ PRINT "WHICH DRIVE";: K1$=DR$: GOSUB 215Ø: DR
     $=K1$                                  :rem 231
191Ø IF DR$<"Ø" OR DR$>"1" THEN 189Ø        :rem 211
192Ø : PRINT "{DOWN}INSTALL NEW DISK -- THEN{UP}":
      GOSUB 169Ø                            :rem 184
```

```
1930 PRINT# 15,"I";DR$: GOSUB 1980: IF EN=Ø THEN R
     ETURN                                    :rem 245
1940 PRINT "CAN'T INITIALIZE -- TRY AGAIN.": GOTO
     {SPACE}1890                              :rem 1Ø
1950 :                                         :rem 9
1960 REM"--SUB--TEST / PRINT DISK ERROR--":rem 161
1970 :                                        :rem 11
1980 : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN=63 TH
     EN RETURN                                 :rem 1
1990 : PRINT Y1$;"{DOWN}{RVS} DISK ERROR {OFF}";YØ
     $;CR$;EN;EM$;ET;ES: RETURN              :rem 141
2ØØØ :                                       :rem 252
2Ø1Ø REM"--SUB--TEST #BLOCKS FREE--"          :rem 38
2Ø2Ø :                                       :rem 254
2Ø3Ø : ER=Ø: OPEN 14,8,Ø,"$"+DO$+":": GOSUB 198Ø:
     {SPACE}IF EN=Ø THEN 2Ø7Ø                  :rem Ø
2Ø4Ø CLOSE 14: PRINT Y1$;"{RVS} CAN'T READ OUTPUT
     {SPACE}DISK DIRECTORY {OFF}";YØ$         :rem 91
2Ø5Ø GOSUB 186Ø: IF ER=1 THEN RETURN         :rem 243
2Ø6Ø GOTO 2Ø3Ø                               :rem 198
2Ø7Ø : FOR JJ=1 TO 18: GET #14,X1$,X2$: NEXT JJ: C
     LOSE 14                                 :rem 111
2Ø8Ø BF%=ASC(X1$+ZR$)+ASC(X2$+ZR$)*256        :rem 71
2Ø9Ø PRINT "{DOWN}";BF%;TAB(6);"BLOCKS FREE"
                                             :rem 141
21ØØ IF BF%>(MC+2*MR)/254+2 THEN RETURN      :rem 122
211Ø ER=1: PRINT Y1$;"{RVS} NOT ENOUGH BLOCKS FREE
      {OFF}";YØ$: RETURN                     :rem 25Ø
212Ø :                                       :rem 255
213Ø REM"--SUB--STRING INPUT--"               :rem 77
214Ø :                                         :rem 1
215Ø : PRINT CU$;SPC(S1);"? ";K1$;CU$;SPC(S1+2);
                                              :rem 95
216Ø INPUT# 1,K1$: PRINT : RETURN             :rem 9Ø
217Ø :                                         :rem 4
218Ø REM"--SUB--NUMERIC INPUT--"             :rem 142
219Ø :                                         :rem 6
22ØØ : PRINT CU$;SPC(S1);"?";K2;CU$;SPC(S1+2);
                                              :rem 56
221Ø INPUT# 1,KB$: PRINT : K2=VAL(KB$): RETURN
                                              :rem 64
222Ø :                                         :rem Ø
223Ø REM"--SUB--CHANGE CASE--"               :rem 169
224Ø :                                         :rem 2
225Ø : CV=PEEK(LC)                           :rem 222
226Ø IF (CV AND 2)=2 THEN POKE LC,(CV AND 253): RE
     TURN                                     :rem 65
227Ø POKE LC,(CV OR 2): RETURN               :rem 14Ø
228Ø :                                         :rem 6
229Ø REM"========= START OF MAIN PROGRAM =========
     "                                       :rem 159
```

```
2300 :                                      :rem 255
2310 REM"---INITIALIZE--"                   :rem 193
2320 :                                        :rem 1
2330 : MR=700: MC=14000: REM"--MAX RECORDS & CHR'S
     --"                                    :rem 252
2340 DIM OD$(21): REM"--PARTITIONING ARRAY-"
                                             :rem 14
2350 DIM FC(21): REM"--MENU CHOICE SELECTION--"
                                            :rem 125
2360 DIM OP(21): REM"--FIRST OLD FIELD NUMBER--"
                                            :rem 162
2370 DIM CP(21): REM"--SECOND OLD FIELD NUMBER--"
                                            :rem 203
2380 DIM AD$(21): REM"--NEW FIXED DATA--" :rem 142
2390 DIM CO$(21): REM"--MORE FIELDS PROMPTING--"
                                            :rem 210
2400 :                                        :rem 0
2410 LC=59468: IF TY=6 THEN LC=53272        :rem 138
2420 CR$=CHR$(13): QT$=CHR$(34): ZR$=CHR$(0): FA$=
     "OD": FT%=0: DR$="0"                    :rem 58
2430 CU$=CR$+"{UP}": S1=22: SP=30979: OPEN 1,0: OP
     EN 15,8,15                             :rem 193
2440 SC$="N": MF$="N": DI$="0": DO$="1": IF TY=6 T
     HEN DO$="0"                            :rem 210
2450 FOR JJ=1 TO 20: FC(JJ)=1: OP(JJ)=JJ: CP(JJ)=J
     J+1: CO$(JJ)="Y"                       :rem 236
2460 NEXT JJ: CP(20)=1                      :rem 141
2470 PRINT "{CLR}{DOWN}{RVS} READY TO RE-STRUCTURE
      DATA RECORDS {OFF}{DOWN}": GOTO 2560 :rem 16
2480 :                                        :rem 8
2490 REM"---OPEN, TEST, AND CLOSE INPUT FILE--"
                                            :rem 155
2500 :                                        :rem 1
2510 : CLOSE 8: PRINT "{DOWN}PRESS {RVS} E {OFF} T
     O EXIT, OR --"                          :rem 70
2520 PRINT "ANY OTHER KEY TO RE-DEFINE": GOSUB 170
     0                                      :rem 129
2530 IF KB$="E" THEN 3620                   :rem 202
2540 : PRINT "{2 DOWN}{RVS} RE-DEFINE THE CONVERSI
     ON {OFF}"                              :rem 242
2550 : GOSUB 1850                            :rem 84
2560 : PRINT "{DOWN}CHANGE DISPLAY CASE";: GOSUB 1
     770                                    :rem 133
2570 IF KB$="Y" THEN GOSUB 2250              :rem 96
2580 IF TY=6 THEN DI$="0": GOTO 2630         :rem 77
2590 : PRINT "{DOWN}SOURCE FILE IN DRIVE"; :rem 41
2600 K1$=DI$: GOSUB 2150: IF K1$<"0" OR K1$>"1" TH
     EN 2590                                :rem 228
2610 DI$=K1$                                 :rem 87
2620 :                                        :rem 4
```

```
263Ø : PRINT "SOURCE FILE NAME";: K1$=FI$: GOSUB 2
     15Ø: FI$=K1$                                :rem 57
264Ø OPEN 8,8,8,DI$+":"+FI$+",S,R": GOSUB 198Ø: IF
      EN<>Ø THEN 251Ø                           :rem 87
265Ø INPUT# 8,X1$: IF ST=Ø THEN 267Ø        :rem 1Ø8
266Ø PRINT Y1$;"{RVS} NO DATA RECORDS IN THE FILE
     {SPACE}{OFF}";YØ$: GOTO 251Ø           :rem 176
267Ø : DL$=LEFT$(X1$,1): FD$=DL$: GOSUB 159Ø: IF E
     F=1 THEN 251Ø                          :rem 191
268Ø INPUT# 8,FL$: CLOSE 8: PRINT            :rem 76
269Ø CF=Ø: FOR JJ=1 TO LEN(FL$): IF MID$(FL$,JJ,1)
     =DL$ THEN CF=CF+1                       :rem 41
27ØØ NEXT JJ                                :rem 156
271Ø :                                        :rem 4
272Ø REM"---DEFINE, CHECK, AND CLOSE OUTPUT FILE--
     "                                      :rem 143
273Ø :                                        :rem 6
274Ø : IF TY=6 THEN K1$="Ø": GOTO 277Ø      :rem 121
275Ø PRINT "OUTPUT FILE TO DRIVE";            :rem 8
276Ø K1$=DO$: GOSUB 215Ø: IF K1$<"Ø" OR K1$>"1" TH
     EN 274Ø                                :rem 238
277Ø : DO$=K1$: IF FO$="" THEN FO$=FI$      :rem 121
278Ø :                                       :rem 11
279Ø : PRINT "OUTPUT FILE NAME";: K1$=FO$: GOSUB 2
     15Ø: FO$=K1$                           :rem 1Ø8
28ØØ IF LEN(FO$)<14 THEN RE$="": GOTO 283Ø:rem 142
281Ø PRINT Y1$;"{RVS} NAME TOO LONG {OFF}";YØ$: FO
     $=LEFT$(FO$,13): GOTO 279Ø             :rem 178
282Ø :                                        :rem 6
283Ø : GOSUB 2Ø3Ø: IF ER=1 THEN ER=Ø: GOTO 254Ø
                                            :rem 139
284Ø OPEN 9,8,9,RE$+DO$+":"+FO$+",S,R"      :rem 196
285Ø INPUT# 15,EN,EM$,ET,ES: CLOSE 9: IF EN=62 THE
     N 298Ø                                 :rem 254
286Ø IF EN<>Ø THEN GOSUB 199Ø: GOTO 251Ø     :rem 89
287Ø IF FO$<>FI$ THEN 294Ø                   :rem 68
288Ø PRINT Y1$;"{DOWN}{RVS} CAUTION {OFF}";YØ$;" I
     F YOU USE THIS PROGRAM TO"              :rem 39
289Ø PRINT "ADD FIXED DATA IT IS POSSIBLE TO CREAT
     E"                                     :rem 232
29ØØ PRINT "RECORDS OVER 75 CHARACTERS LONG WHICH"
                                            :rem 217
291Ø PRINT "CAN'T BE HANDLED BY 'DFH' PROGRAMS."
                                            :rem 181
292Ø PRINT "{DOWN}REPLACING YOUR SOURCE FILE COULD
      CAUSE"                                :rem 1Ø8
293Ø PRINT "EFFECTIVE LOSS OF ACCESS TO THAT DATA.
     "                                      :rem 212
294Ø : PRINT "{DOWN}REPLACE EXISTING FILE";
                                            :rem 173
```

```
2950 GOSUB 176Ø: IF KB$="N" THEN 251Ø       :rem 94
296Ø RE$="@"                                :rem 77
297Ø :                                      :rem 12
298Ø : PRINT "{DOWN}DELIMITER TO BE USED";: K1$=DL
     $: GOSUB 215Ø: DE$=K1$                 :rem 53
299Ø FD$=DE$: GOSUB 159Ø: IF EF=1 THEN 298Ø
                                           :rem 185
3ØØØ :                                     :rem 253
3Ø1Ø FD$=DL$: IF RD=1 OR FP=Ø THEN FP=1: GOTO 3Ø4Ø
                                            :rem 65
3Ø2Ø PRINT "{DOWN}SAME CONVERSION";: KB$=SC$: GOSU
     B 178Ø: SC$=KB$                        :rem 92
3Ø3Ø IF SC$="Y" THEN 339Ø                  :rem 231
3Ø4Ø : NF=1: GOTO 313Ø                      :rem 61
3Ø5Ø :                                       :rem 2
3Ø6Ø REM"---SET UP NEW FILE STRUCTURE--"    :rem 62
3Ø7Ø :                                       :rem 4
3Ø8Ø : IF NF>19 THEN PRINT "{DOWN}{RVS} 2Ø FIELDS
     {SPACE}MAX. {OFF}": GOTO 339Ø         :rem 244
3Ø9Ø PRINT "{DOWN}MORE FIELDS";            :rem 211
31ØØ KB$=CO$(NF): GOSUB 178Ø: CO$(NF)=KB$: IF KB$=
     "N" THEN 339Ø                         :rem 225
311Ø NF=NF+1                               :rem 134
312Ø :                                       :rem Ø
313Ø : PRINT "{CLR}FIRST RECORD (";CF;"FIELDS) IN"
                                           :rem 142
314Ø PRINT "SOURCE FILE {RVS} ";FI$;" {OFF} IS:";C
     R$;FL$                                 :rem 86
315Ø PRINT "{DOWN}NEW DATA FIELD {RVS} #";NF;"
     {LEFT} {OFF} TO CONTAIN:"             :rem 174
316Ø PRINT "{RVS} 1 {OFF}{2 SPACES}DATA FROM AN OL
     D DATA FIELD"                         :rem 17Ø
317Ø PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}NEW FIXED
      DATA"                                :rem 247
318Ø PRINT "{DOWN}{RVS} 3 {OFF}{2 SPACES}OLD DATA
     {SPACE}FIELD + NEW FIXED DATA"        :rem 129
319Ø PRINT "{DOWN}{RVS} 4 {OFF}{2 SPACES}NEW FIXED
      DATA + OLD DATA FIELD"               :rem 131
32ØØ PRINT "{DOWN}{RVS} 5 {OFF}{2 SPACES}DATA FROM
      TWO OLD DATA FIELDS"                 :rem 12Ø
321Ø PRINT "--------------------------------------
     -"                                    :rem 114
322Ø PRINT "{RVS} 9 {OFF}{2 SPACES}RE-DEFINE OR EX
     IT"                                   :rem 191
323Ø : RD=Ø: PRINT "YOUR CHOICE -----";     :rem 38
324Ø K2=FC(NF): GOSUB 22ØØ: IF K2=9 THEN RD=1: GOT
     O 251Ø                                :rem 153
325Ø FC(NF)=K2: IF FC(NF)=1 THEN GOSUB 148Ø: GOTO
     {SPACE}3Ø8Ø                            :rem 81
```

```
3260 IF FC(NF)=2 THEN GOSUB 1540: GOTO 3080
                                              :rem 238
3270 IF FC(NF)=3 THEN GOSUB 1480: GOSUB 1540: GOTO
      3080                                    :rem 119
3280 IF FC(NF)=4 THEN GOSUB 1540: GOSUB 1480: GOTO
      3080                                    :rem 121
3290 IF FC(NF)=5 THEN 3320                     :rem 52
3300 PRINT "{2 UP}": GOTO 3230                :rem 244
3310 :                                          :rem 1
3320 : PRINT "{DOWN}FIRST OLD DATA FIELD";: K2=OP(
     NF): GOSUB 2200                          :rem 250
3330 IF K2<1 OR K2>20 THEN PRINT Y1$;"{RVS} ILLEGA
     L # {OFF}";Y0$: GOTO 3320                :rem 205
3340 OP(NF)=K2                                  :rem 8
3350 : PRINT "{DOWN}SECOND OLD DATA FIELD";: K2=CP
     (NF): GOSUB 2200                          :rem 37
3360 IF K2<1 OR K2>20 THEN PRINT Y1$;"{RVS} ILLEGA
     L # {OFF}";Y0$: GOTO 3350                :rem 211
3370 CP(NF)=K2: GOTO 3080                      :rem 61
3380 :                                          :rem 8
3390 : PRINT "{2 DOWN}{RVS} READY TO CONVERT FILE:
      {OFF}{DOWN}"                             :rem 68
3400 PRINT "{RVS} ";FI$;" {OFF} -TO- {RVS} ";FO$;"
      {OFF}"                                  :rem 189
3410 PRINT "{DOWN}PRESS {RVS} R {OFF} TO RE-DEFINE
      OR{UP}": GOSUB 1690                     :rem 251
3420 IF KB$="R" THEN 2540                     :rem 214
3430 IF KB$="Q" THEN 3610                     :rem 213
3440 :                                          :rem 5
3450 REM"---OPEN INPUT AND OUTPUT FILES AND DO CON
     VERSION--"                                :rem 58
3460 :                                          :rem 7
3470 OPEN 8,8,8,DI$+":"+FI$+",S,R": GOSUB 1980: IF
      EN<>0 THEN 2510                          :rem 89
3480 INPUT# 8,X1$                             :rem 147
3490 OPEN 9,8,9,RE$+DO$+":"+FO$+",S,W": GOSUB 1980
                                               :rem 87
3500 IF EN<>0 THEN CLOSE 9: GOTO 2510         :rem 173
3510 :                                          :rem 3
3520 PRINT# 9,QT$;DE$;"←@0:";FO$;CR$;          :rem 59
3530 GOSUB 1260: GOSUB 1980: CLOSE 8: CLOSE 9: IF
     {SPACE}EN=0 THEN 3550                     :rem 54
3540 PRINT Y1$;"{RVS} CONVERSION NOT SUCCESSFUL
     {OFF}";Y0$: GOTO 2510                     :rem 68
3550 : PRINT CR$;"{RVS} CONVERSION COMPLETE {OFF}"
                                              :rem 207
3560 PRINT "ANY MORE FILES";: KB$=MF$: GOSUB 1780:
      MF$=KB$                                 :rem 176
3570 IF MF$="Y" THEN 2550                     :rem 234
3580 :                                         :rem 10
```

```
359Ø REM"---PROGRAM TERMINATION--"           :rem 6Ø
36ØØ :                                        :rem 3
361Ø : CLOSE 8: CLOSE 9                     :rem 155
362Ø : PRINT "{DOWN}PRESS {RVS} Q {OFF} TO QUIT, O
     R --"                                  :rem 118
363Ø PRINT "ANY OTHER KEY FOR MASTER MENU":rem 216
364Ø GOSUB 17ØØ: IF KB$<>"Q" THEN 366Ø      :rem 156
365Ø PRINT "{RVS} PROGRAM TERMINATED{2 SPACES}
     {OFF}": CLOSE 1: CLOSE 15: END          :rem 8Ø
366Ø : PS$="DFH BOOT": OPEN 8,8,8,"Ø:"+PS$+",P,R"
                                            :rem 248
367Ø GOSUB 198Ø: CLOSE 8: IF EN=Ø THEN 369Ø
                                            :rem 212
368Ø PRINT "{DOWN}TRYING TO LOAD {RVS} ";PS$;"
     {OFF}": GOSUB 186Ø: GOTO 362Ø           :rem 63
369Ø : CLOSE 1: CLOSE 15: PRINT "{DOWN}{RVS} LOADI
     NG ";PS$;" {OFF}"                        :rem 8
37ØØ POKE 824,248: LOAD PS$,8               :rem 23Ø
```

DFH SPLIT

For mistake-proof program entry, be sure to use "The Automatic Proofreader," Chapter 9.

```
1ØØØ REM SAVE "@Ø:DFH SPLIT",8              :rem 132
1Ø1Ø :                                      :rem 252
1Ø2Ø REM" A DATA FILE HANDLER PROGRAM TO SPLIT OR
     {SPACE}EXTRACT RECORDS FROM"           :rem 173
1Ø3Ø REM" MULTI-FIELD SEQUENTIAL DATA FILES."
                                            :rem 19Ø
1Ø4Ø :                                      :rem 255
1Ø8Ø :                                        :rem 3
1Ø9Ø REM"--SET TOP OF BASIC IF REQUIRED--":rem 1Ø8
11ØØ :                                      :rem 252
111Ø IF PEEK (65534)=72 THEN 115Ø: REM"--C-64 COMP
     UTER--"                                :rem 2Ø2
112Ø IF PEEK (824)<>248 THEN 119Ø: REM"--NOT A PRO
     G'D LOAD--"                            :rem 132
113Ø POKE 42,PEEK(2Ø1): POKE 43,PEEK (2Ø2):rem 149
114Ø :                                        :rem Ø
115Ø : POKE 5328Ø,14: POKE 53281,14: PRINT CHR$(31
     );: REM"--COLORS--"                     :rem 96
116Ø IF PEEK (824)<>248 THEN 119Ø: REM"--NOT A PRO
     G'D LOAD--"                            :rem 136
117Ø POKE 45,PEEK(174): POKE 46,PEEK(175) :rem 177
118Ø :                                        :rem 4
119Ø : CLR : POKE 824,Ø: TY=2: YØ$="": Y1$=""
                                            :rem 112
12ØØ IF PEEK (65534)=72 THEN TY=6: YØ$=CHR$(31): Y
     1$=CHR$(158)                           :rem 179
121Ø :                                      :rem 254
```

```
122Ø PRINT "{CLR}{RVS} READY TO SPLIT FILES OR EXT
     RACT DATA {OFF}{DOWN}": GOTO 297Ø      :rem 16
123Ø REM"========== START OF SUBROUTINES =========
     ==="                                  :rem 178
124Ø :                                       :rem 1
125Ø REM"--SUB--INPUT NEXT RECORD--"       :rem 118
126Ø :                                       :rem 3
127Ø : INPUT# 8,RC$(Ø)                      :rem 85
128Ø : INPUT# 8,RC$(Ø): TT=ST: CE=CE+1: SYS SP: IF
      FT%<CF THEN RC$(CF)=""               :rem 1Ø7
129Ø RETURN                                :rem 172
13ØØ :                                     :rem 254
131Ø REM"--SUB--MATCH AND STORE--"         :rem 178
132Ø :                                       :rem Ø
133Ø : IF ME=2 THEN TT=1: ME=Ø: RETURN      :rem 41
134Ø GOSUB 128Ø: LS=BB: IF PD$="A" THEN LS=LEN(RC$
     (CF))-LE+1                            :rem 216
135Ø FOR JJ=BB TO LS: IF RM$=MID$(RC$(CF),JJ,LE) T
     HEN FS=1: JJ=LS                        :rem 79
136Ø NEXT JJ: IF FS=1 THEN FS=Ø: KS=KS+1: GOTO 14Ø
     Ø                                     :rem 233
137Ø IF SE$="E" THEN 142Ø                  :rem 21Ø
138Ø IF TT<>Ø THEN ME=2: TT=Ø               :rem 44
139Ø RETURN                                :rem 173
14ØØ : IF M1=2 OR CO=2 THEN 144Ø            :rem 21
141Ø : CS=CS+1: BC=BC+LEN(RC$(Ø)): IF RS$="N" THEN
      PR$(CS)=RC$(Ø)                       :rem 143
142Ø : IF SE$="E" THEN KE=KE+1: PRINT "{2 UP}";KE;
     CR$;KS                                :rem 191
143Ø IF CS>=MR OR BC>MB THEN PRINT MF$: ME=1: RETU
     RN                                    :rem 193
144Ø : IF TT<>Ø THEN RETURN                :rem 243
145Ø GOTO 133Ø                             :rem 2Ø2
146Ø :                                       :rem 5
147Ø REM"--SUB--SPOOL-UP--"                 :rem 75
148Ø :                                       :rem 7
149Ø : GOSUB 216Ø: GOSUB 24ØØ: IF EN<>Ø THEN 149Ø
                                           :rem 212
15ØØ INPUT# 8,RC$(Ø): IF CE<2 THEN 153Ø    :rem 2Ø6
151Ø PRINT "{DOWN}SPOOLING UP IN "QT$;OF$;QT$: RP%
     =(CE-1)/256                            :rem 82
152Ø POKE HB,RP%: POKE LB,CE-1-RP%*256: SYS SU
                                            :rem 25
153Ø : GOSUB 128Ø: CE=CE-1: RETURN          :rem 77
154Ø :                                       :rem 4
155Ø REM"--SUB--SAVE TO DISK--"            :rem 232
156Ø :                                       :rem 6
157Ø : IF D1$="Y" THEN 16ØØ                :rem 255
158Ø : CLOSE 8                             :rem 182
159Ø : GOSUB 217Ø                           :rem 83
```

```
1600 : OPEN 14,8,0,"$"+OD$+":": GOSUB 1940: IF EN=
     0 THEN 1630                             :rem 193
1610 CLOSE 14: PRINT "{RVS}CAN'T READ DIRECTORY ON
      DRIVE # ";OD$;" {OFF}"                  :rem 27
1620 GOTO 1590                               :rem 209
1630 : FOR JJ=1 TO 18: GET #14,X1$,X2$: NEXT JJ: C
     LOSE 14                                 :rem 112
1640 BF=ASC(X1$+ZR$)+ASC(X2$+ZR$)*256         :rem 35
1650 PRINT "{DOWN}";BF;" BLOCKS FREE": IF BF>(BC+C
     S*2)/254+2 THEN 1710                    :rem 113
1660 PRINT Y1$;"{DOWN}{RVS} NOT ENOUGH BLOCKS FREE
      {OFF}";Y0$                             :rem 187
1670 : IF TY=2 OR D1$="N" THEN 1590          :rem 186
1680 D1$="N": PRINT "{DOWN}REMAINDER OF OUTPUT FIL
     ES WILL"                                 :rem 27
1690 PRINT "BE ON A SEPERATE DISKETTE !": GOTO 158
     0                                        :rem 30
1700 :                                         :rem 2
1710 : SF$=YN$+SC$+",S,W"                     :rem 30
1720 : OPEN 9,8,9,RE$+OD$+":"+SF$: GOSUB 1940: IF
     {SPACE}EN=0 THEN 1860                   :rem 225
1730 CLOSE 9: IF EN<>63 THEN GOSUB 2820: GOTO 1590
                                             :rem 118
1740 PRINT "{DOWN}{RVS} FILE {OFF} ";QT$;YN$;SC$;Q
     T$;" {RVS} EXISTS {OFF}"                :rem 120
1750 PRINT "{DOWN}WANT TO REPLACE IT";        :rem 96
1760 GOSUB 2640: IF KB$="Y" THEN RE$="@": GOTO 172
     0                                        :rem 86
1770 PRINT "{DOWN}OPTIONS AVAILABLE:{DOWN}"
                                             :rem 169
1780 IF SQ$="I" THEN PRINT "{RVS} R {OFF} RENAME O
     UTPUT FILE"                             :rem 176
1790 PRINT "{RVS} C {OFF} CHANGE DISKETTES"
                                             :rem 223
1800 : PRINT "{RVS} Q {OFF} QUIT";            :rem 71
1810 K1$=OP$: GOSUB 2720: OP$=K1$: IF OP$="Q" THEN
      4590                                    :rem 40
1820 IF OP$="C" THEN 1670                    :rem 222
1830 IF OP$="R" AND SQ$="I" THEN GOSUB 2480: GOTO
     {SPACE}1710                              :rem 15
1840 PRINT "{UP}";: GOTO 1800                :rem 166
1850 :                                         :rem 8
1860 : PRINT# 9,FD$"←@";OD$;":";YN$;SC$       :rem 85
1870 PRINT TAB(6);"{DOWN}RECORDS OUT TO ";QT$;YN$+
     SC$;QT$: FOR JJ=1 TO CS                  :rem 71
1880 PRINT "{UP}{RVS}";JJ;"{LEFT} {OFF}": PRINT# 9
     ,PR$(JJ): PR$(JJ)="": NEXT JJ            :rem 47
1890 GOSUB 1940 : CLOSE 9: IF EN=0 THEN BC=0: CS=0
     : RE$="": RETURN                        :rem 192
```

```
19ØØ PRINT Y1$;"{DOWN}{RVS} FILE NOT SAVED PROPERL
     Y {OFF}";YØ$: GOTO 459Ø                 :rem 1Ø3
191Ø :                                         :rem 5
192Ø REM"--SUB--TEST/PRINT DISK ERROR--"    :rem 157
193Ø :                                         :rem 7
194Ø : INPUT# 15,EN,EM$,ET,ES: IF EN=Ø OR EN=63 TH
     EN RETURN                               :rem 253
195Ø PRINT Y1$;"{DOWN}{RVS} DISK ERROR {OFF}";YØ$;
     CR$;EN;EM$;ET;ES: RETURN                 :rem 79
196Ø :                                        :rem 1Ø
197Ø REM"--SUB--TEST DELIMITER--"            :rem 2Ø8
198Ø :                                        :rem 12
199Ø : INPUT# 8, RC$(Ø): TT=ST: FD$=LEFT$(RC$(Ø),1
     )                                       :rem 128
2ØØØ EF=Ø: IF FD$<>"" THEN 2Ø2Ø              :rem 226
2Ø1Ø PRINT "{RVS}ENCLOSE COMMA, COLON OR SPACE IN
     {SPACE}QUOTES": EF=1: RETURN           :rem 149
2Ø2Ø : IF LEN(FD$)<>1 OR ASC(FD$)<32 THEN EF=1
                                              :rem 86
2Ø3Ø IF ASC(FD$)>127 AND ASC(FD$)<161 THEN EF=1
                                             :rem 167
2Ø4Ø IF FD$="Ø" OR VAL(FD$)<>Ø THEN EF=1      :rem 9
2Ø5Ø IF EF=1 THEN PRINT Y1$;"{DOWN}{RVS} ILLEGAL D
     ELIMITER {OFF}{2 SPACES}";YØ$;FD$        :rem 87
2Ø6Ø RETURN                                  :rem 168
2Ø7Ø :                                         :rem 3
2Ø8Ø REM"--SUB--INCREMENT SEQ. NAME--"      :rem 199
2Ø9Ø :                                         :rem 5
21ØØ : SN=SN+1: SC$="": IF SN>99 THEN 212Ø :rem 91
211Ø SC$="Ø": IF SN<1Ø THEN SC$=SC$+"Ø"     :rem 181
212Ø : SC$="."+SC$+MID$(STR$(SN),2): RETURN:rem 78
213Ø :                                         :rem Ø
214Ø REM"--SUB--DISK CHANGE/INITILIZATION--"
                                             :rem 2Ø6
215Ø :                                         :rem 2
216Ø : DR$=SD$: DT$="{RVS} SOURCE {OFF}": GOTO 218
     Ø                                       :rem 223
217Ø : DR$=OD$: DT$="{RVS} OUTPUT {OFF}"    :rem 19Ø
218Ø : PRINT "{DOWN}INSTALL ";DT$;" DISKETTE"
                                             :rem 2Ø9
219Ø IF TY<>6 THEN PRINT TAB(17);"IN DRIVE # ";DR$
                                             :rem 172
22ØØ PRINT "THEN, ";                          :rem 43
221Ø : GOSUB 256Ø: PRINT# 15,"I";DR$: GOSUB 194Ø:
     {SPACE}IF EN=Ø THEN RETURN             :rem 17Ø
222Ø PRINT "CAN'T INITIALIZE -- TRY AGAIN.": GOTO
     {SPACE}218Ø                             :rem 251
223Ø :                                         :rem 1
224Ø REM"--SUB--PREVIEW TO PRINTER--"       :rem 2Ø9
225Ø :                                         :rem 3
```

```
226Ø : PRINT# 4,CR$;CR$;"SPLIT PREVIEW OF FILE: ";
     OF$;CR$;PF$;CF                           :rem 126
227Ø PRINT# 4,CR$;"FILE";SPC(3);"NO. OF";SPC(4);"N
     O. OF";SPC(3);                           :rem 28
228Ø PRINT# 4,"DATA";CR$;SPC(7);"RECORDS";SPC(3);"
     BYTES";SPC(4);                           :rem 244
229Ø PRINT# 4,"STRING";CR$: RETURN              :rem 6
23ØØ :                                        :rem 255
231Ø : PRINT# 4,CA$;SC$;SPC(8-LEN(STR$(CS)));CS;SP
     C(9-LEN(STR$(BC)));BC;                   :rem 127
232Ø PRINT# 4,SPC(3);QT$;LEFT$(RM$,5Ø);       :rem 175
233Ø IF LEN(RM$)>5Ø THEN PRINT# 4,CR$;CA$;SPC(26);
     MID$(RM$,51);: PL=PL+1                   :rem 199
234Ø PRINT# 4,QT$;CR$;: PL=PL+1: IF PL<6Ø THEN RET
     URN                                      :rem 242
235Ø : FOR JJ=1 TO 66-PL: PRINT# 4,CR$;: NEXT JJ:
     {SPACE}PL=Ø: RETURN                      :rem 1Ø9
236Ø :                                          :rem 5
237Ø REM"--SUB--OPEN INPUT FILE--"            :rem 2Ø6
238Ø :                                          :rem 7
239Ø : GOSUB 246Ø                              :rem 84
24ØØ : OPEN 8,8,8,SD$+":"+TF$+",S"            :rem 153
241Ø GOSUB 194Ø: IF EN<>Ø THEN CLOSE 8        :rem 248
242Ø RETURN                                   :rem 168
243Ø :                                          :rem 3
244Ø REM"--SUB--GET FILENAMES--"               :rem 94
245Ø :                                          :rem 5
246Ø : PRINT "SOURCE FILE NAME";: K1$=TF$: GOSUB 2
     72Ø: TF$=K1$: RETURN                     :rem 1Ø9
247Ø :                                          :rem 7
248Ø : PRINT "{DOWN}OUTPUT FILE NAME";: K1$=YN$: G
     OSUB 272Ø: YN$=K1$                       :rem 16Ø
249Ø IF YN$=OF$ THEN PRINT NE$: GOTO 248Ø :rem 211
25ØØ IF LEN(YN$)<13 THEN RETURN                 :rem Ø
251Ø PRINT "{DOWN}{RVS} FILE NAME TOO LONG {OFF}":
      YN$=LEFT$(YN$,12): GOTO 248Ø             :rem 46
252Ø :                                          :rem 3
253Ø REM"--SUB--WAIT FOR OPERATOR--"          :rem 114
254Ø :                                          :rem 5
255Ø : PRINT "{DOWN}";                         :rem 35
256Ø : PRINT "PRESS ANY KEY TO CONTINUE"       :rem 62
257Ø : GET KB$: IF KB$<>"" THEN 257Ø          :rem 2Ø6
258Ø : GET KB$: IF KB$="" THEN 258Ø           :rem 147
259Ø RETURN                                   :rem 176
26ØØ :                                          :rem 2
261Ø REM"--SUB--WAIT FOR YES OR NO ANSWER--":rem 4
262Ø :                                          :rem 4
263Ø : KB$="Y": GOTO 265Ø                     :rem 2Ø8
264Ø : KB$="N"                                :rem 134
```

```
265Ø : PRINT CU$;SPC(S1);"? ";KB$;CU$;SPC(S1+2);
                                           :rem 117
266Ø : INPUT# 1,KB$: PRINT : KB$=LEFT$(KB$,1)
                                           :rem 1Ø2
267Ø IF KB$="Y" OR KB$="N" THEN RETURN     :rem 25
268Ø PRINT "{RVS} Y {OFF} YES OR {RVS} N {OFF} NO
     {SPACE}? ";: GOTO 266Ø                :rem 122
269Ø :                                     :rem 11
27ØØ REM"--SUB--STRING INPUT--"            :rem 8Ø
271Ø :                                      :rem 4
272Ø : PRINT CU$;SPC(S1);"? ";K1$;CU$;SPC(S1+2);
                                            :rem 98
273Ø INPUT# 1,K1$: PRINT : RETURN           :rem 93
274Ø :                                      :rem 7
275Ø REM"--SUB--NUMERIC INPUT--"           :rem 145
276Ø :                                      :rem 9
277Ø : PRINT CU$;SPC(S1);"?";K2;CU$;SPC(S1+2);
                                            :rem 68
278Ø INPUT# 1,KB$: PRINT : K2=VAL(KB$): RETURN
                                            :rem 76
279Ø :                                      :rem 12
28ØØ REM"--SUB--QUIT OR CONTINUE--"         :rem 51
281Ø :                                      :rem 5
282Ø : PRINT "{DOWN}PRESS {RVS} E {OFF} TO EXIT, O
     R --"                                  :rem 98
283Ø PRINT "ANY OTHER KEY TO CONTINUE": GOSUB 257Ø
                                           :rem 129
284Ø IF KB$="E" THEN 459Ø                  :rem 213
285Ø RETURN                                :rem 175
286Ø :                                      :rem 1Ø
287Ø REM"--SUB--CHANGE CASE--"             :rem 179
288Ø :                                      :rem 12
289Ø : CV=PEEK(LC)                         :rem 232
29ØØ IF (CV AND 2)=2 THEN POKE LC,(CV AND 253): CA
     $="": RETURN                          :rem 165
291Ø POKE LC,(CV OR 2): CA$="{DOWN}": RETURN:rem 1
292Ø :                                      :rem 7
293Ø REM"====== START OF MAIN PROGRAM ======"
                                            :rem 5Ø
294Ø :                                      :rem 9
295Ø REM"--INITIALIZE--"                   :rem 158
296Ø :                                      :rem 11
297Ø : SP=3Ø979: LB=3Ø993: HB=3Ø994: SU=3Ø985: MR=
     65Ø: MB=13ØØØ                         :rem 226
298Ø DIM RC$(2Ø),PR$(MR): LC=59468: IF TY=6 THEN L
     C=53272                               :rem 248
299Ø CA$="": IF (PEEK (LC) AND 2)<>Ø THEN CA$="
     {DOWN}"                                :rem 52
3ØØØ CR$=CHR$(13): QT$=CHR$(34): ZR$=CHR$(Ø): CU$=
     CR$+"{UP}"                             :rem 29
```

```
3Ø1Ø FA$="RC": FT%=Ø: OPEN 1,Ø: OPEN 15,8,15
                                          :rem 172
3Ø2Ø MF$="{DOWN}{RVS} MEMORY FULL. MUST OUTPUT FIL
     E {OFF}"                             :rem 7Ø
3Ø3Ø PF$="BASED ON CONTENTS OF FIELD:"    :rem 158
3Ø4Ø NE$=Y1$+"{RVS} CONFLICT WITH SOURCE FILENAME
     {SPACE}{OFF}"+YØ$                    :rem 244
3Ø5Ø SD$="Ø": OD$="1": OP$="C": SE$="S": CF=1: S1=
     23: ME=Ø: PD$="B"                    :rem 225
3Ø6Ø AN$="E": BB=1: LE=2: SQ$="S": D1$="Y": AU$="O
     ": GOTO 3Ø8Ø                         :rem 45
3Ø7Ø : CLOSE 4: PRINT "{CLR}{DOWN}{RVS}{4 SPACES}R
     EDEFINE JOB SETUP{4 SPACES}{OFF}"    :rem 175
3Ø8Ø : M1=1: BC=Ø: CS=Ø: SS$="": SL$="": SN=Ø: CE=
     Ø: SC$="": PL=8                      :rem 148
3Ø9Ø KS=Ø: KE=Ø                           :rem 14
31ØØ : PRINT "{DOWN}{RVS} S {OFF} SPLIT OR";CR$;"
     {RVS} E {OFF} EXTRACT";: K1$=SE$: GOSUB 272Ø
                                          :rem 14
311Ø IF K1$<>"E" AND K1$<>"S" THEN PRINT "{3 UP}";
     : GOTO 31ØØ                          :rem 16Ø
312Ø IF K1$<>SE$ AND K1$="E" THEN RM$=""  :rem 215
313Ø SE$=K1$: IF SE$="S" THEN PRINT "{DOWN}PREVIEW
      TO PRINTER";: GOTO 315Ø             :rem 42
314Ø PRINT "{DOWN}PREVIEW # OF EXTRACTS";::rem 1Ø3
315Ø : GOSUB 264Ø: RS$=KB$: IF TY=6 THEN OD$="Ø":
     {SPACE}GOTO 318Ø                     :rem 255
316Ø : PRINT "{DOWN}SOURCE FILES ON DRIVE";: K1$=S
     D$: GOSUB 272Ø: SD$=K1$              :rem 165
317Ø IF SD$<"Ø" OR SD$>"1" THEN PRINT "{2 UP}";: G
     OTO 316Ø                             :rem 11Ø
318Ø : GOSUB 239Ø: IF EN<>Ø THEN GOSUB 282Ø: GOSUB
      216Ø: GOTO 3Ø7Ø                     :rem 149
319Ø OF$=TF$: GOSUB 199Ø: IF EF=1 THEN 459Ø
                                          :rem 2Ø9
32ØØ IF TT<>Ø THEN PRINT "{DOWN}{RVS} NO DATA IN F
     ILE {OFF}": GOTO 459Ø                :rem 14
321Ø GOSUB 128Ø: CE=Ø: CLOSE 8            :rem 4Ø
322Ø IF RS$="Y" THEN AU$="A": OPEN 4,4: GOTO 33ØØ
                                          :rem 221
323Ø IF TY<>6 THEN 326Ø                   :rem 174
324Ø PRINT "{DOWN}SOURCE AND OUTPUT";CR$;"FILES ON
      SAME";                              :rem 36
325Ø PRINT " DISKETTE";: KB$=D1$: GOSUB 265Ø: D1$=
     KB$: GOTO 33ØØ                       :rem 117
326Ø : PRINT "{DOWN}OUTPUT FILES TO DRIVE";: K1$=O
     D$: GOSUB 272Ø: OD$=K1$              :rem 196
327Ø IF OD$<"Ø" OR OD$>"1" THEN PRINT "{2 UP}";: G
     OTO 326Ø                             :rem 1Ø4
328Ø IF SD$=OD$ THEN TY=1                 :rem 85
```

```
329Ø :                                          :rem 8
33ØØ : PRINT "{2 DOWN}FIRST RECORD IN {RVS} ";OF$;
     " {OFF} IS:{DOWN}";CR$;RC$(Ø)             :rem 56
331Ø PRINT "{DOWN}CHANGE PRINT CASE";: GOSUB 264Ø:
      IF KB$="Y" THEN GOSUB 289Ø             :rem 143
332Ø IF SE$="E" THEN SQ$="I": GOTO 419Ø    :rem 219
333Ø : PRINT "{DOWN}SPLIT ON WHICH FIELD";: K2=CF:
      GOSUB 277Ø: CF=K2                     :rem 164
334Ø IF CF<1 OR CF>2Ø THEN PRINT "{2 UP}";: GOTO 3
     33Ø                                    :rem 178
335Ø PRINT "{DOWN}FIELD";CF;" = ";QT$;RC$(CF);QT$
                                            :rem 1Ø3
336Ø PRINT "{2 DOWN}SPLIT AT CHANGES IN:";CR$;"
     {DOWN}{RVS} E {OFF} ENTIRE FIELD, OR" :rem 15
337Ø : PRINT "{RVS} S {OFF} SELECTED POSITIONS";:
     {SPACE}K1$=AN$: GOSUB 272Ø             :rem 1Ø6
338Ø IF AN$<>K1$ AND K1$="S" THEN BB=1: LE=2
                                            :rem 2Ø4
339Ø AN$=K1$: IF AN$="E" THEN BB=1: LE=8Ø: GOTO 34
     4Ø                                     :rem 112
34ØØ IF AN$<>"S" THEN PRINT "{UP}";: GOTO 337Ø
                                            :rem 38
341Ø PRINT "{DOWN}START POSITION";: K2=BB: GOSUB 2
     77Ø: BB=K2                             :rem 98
342Ø PRINT "# OF CHARACTERS";: K2=LE: GOSUB 277Ø:
     {SPACE}LE=K2                            :rem 1
343Ø PRINT "{DOWN}SELECTED FROM FIELD";CF;"{LEFT}:
      ";QT$;MID$(RC$(CF),BB,LE);QT$         :rem 57
344Ø : IF RS$="Y" THEN 357Ø                 :rem 53
345Ø :                                       :rem 6
346Ø PRINT "{2 DOWN}SPLITTING/SAVING PROCESS TO BE
     :"                                     :rem 248
347Ø : PRINT "{DOWN}{RVS} A {OFF} AUTOMATIC, OR";C
     R$;"{RVS} O {OFF} OPERATOR'S CHOICE";:rem 117
348Ø K1$=AU$: GOSUB 272Ø: AU$=K1$: IF AU$="A" THEN
      SQ$="S": GOTO 354Ø                    :rem 11
349Ø IF AU$<>"O" THEN PRINT "{3 UP}";: GOTO 347Ø
                                            :rem 85
35ØØ PRINT "{2 DOWN}SELECT OUTPUT FILENAMES:";CR$;
     "{DOWN}{RVS} I {OFF} INDIVIDUALLY OR":rem 218
351Ø : PRINT "{RVS} S {OFF} SEQUENTIALLY";: K1$=SQ
     $: GOSUB 272Ø: SQ$=K1$                 :rem 233
352Ø IF SQ$="I" THEN 357Ø                   :rem 233
353Ø IF SQ$<>"S" THEN PRINT "{UP}";: GOTO 351Ø
                                            :rem 59
354Ø : PRINT "{DOWN}A 3-DIGIT SEQUENCE NUMBER WILL
     "                                      :rem 84
355Ø PRINT "BE ADDED TO THE NAME YOU ENTER": GOSUB
      248Ø                                  :rem 31
```

```
3560 IF YN$+"."=LEFT$(OF$,LEN(YN$)+1) THEN PRINT N
     E$: GOTO 354Ø                               :rem 144
357Ø : PRINT "{DOWN}SPLIT DEFINED OK";: GOSUB 263Ø
     : IF KB$<>"Y" THEN 3Ø7Ø                     :rem 74
358Ø : GOSUB 24ØØ: IF EN<>Ø THEN GOSUB 216Ø: GOTO
     {SPACE}358Ø                                 :rem 17
359Ø GOSUB 127Ø: IF AU$="A" THEN RM$=MID$(RC$(CF),
     BB,LE): GOTO 4Ø9Ø                           :rem 3Ø
36ØØ :                                            :rem 3
361Ø REM"--MENU--"                               :rem 219
362Ø :                                            :rem 5
363Ø : IF SE$="E" THEN PRINT "{CLR}EXTRACTING FROM
      FILE";: GOTO 365Ø                          :rem 115
364Ø : PRINT "{CLR}SPLITTING FILE";              :rem 132
365Ø : PRINT " {RVS} ";OF$;" {OFF}";CR$;PF$;CF
                                                 :rem 157
366Ø IF RS$="Y" THEN BC=Ø: CS=Ø: PRINT "{2 DOWN}":
      GOTO 368Ø                                  :rem 2Ø8
367Ø PRINT "{DOWN}";CS;"RECORDS (";BC;"BYTES) IN M
     EMORY"                                      :rem 155
368Ø : IF CS=Ø OR AU$="A" OR SE$="E" THEN PRINT "
     {3 DOWN}": GOTO 37ØØ                        :rem 77
369Ø PRINT "{DOWN}";QT$;SS$;QT$;CR$;"--THRU--";CR$
     ;QT$;SL$;QT$                                :rem 37
37ØØ : PRINT "------------------------------------
     "                                           :rem 41
371Ø IF TT<>Ø THEN CLOSE 8: PRINT "{DOWN}{RVS} END
      OF FILE {OFF} ";OF$: GOTO 375Ø             :rem 11
372Ø PRINT "NEXT RECORD GROUP IS:";CR$;QT$;RC$(CF)
     ;QT$                                        :rem 2ØØ
373Ø PRINT "{DOWN}{RVS} 1 {OFF}{2 SPACES}ADD NEXT
     {SPACE}RECORD GROUP TO MEMORY"              :rem 84
374Ø PRINT "{DOWN}{RVS} 2 {OFF}{2 SPACES}DISREGARD
      NEXT RECORD GROUP"                         :rem 166
375Ø : IF CS>Ø THEN PRINT "{DOWN}{RVS} 3 {OFF}
     {2 SPACES}SAVE RECORDS IN MEMORY TO DISK"
                                                  :rem 3
376Ø IF TT<>Ø THEN PRINT "{DOWN}{RVS} 4 {OFF}
     {2 SPACES}CONTINUE TO NEXT SOURCE FILE"
                                                 :rem 21Ø
377Ø PRINT "{2 DOWN}{RVS} 8 {OFF}{2 SPACES}DEFINE
     {SPACE}NEW JOB SETUP"                       :rem 161
378Ø PRINT "{DOWN}{RVS} 9 {OFF}{2 SPACES}QUIT OR G
     O TO MASTER MENU"                           :rem 175
379Ø IF TT<>Ø AND CS>Ø THEN M1=3: GOTO 382Ø
                                                 :rem 232
38ØØ IF TT<>Ø THEN M1=4: GOTO 382Ø               :rem 1Ø
381Ø IF M1=4 THEN M1=1                           :rem 1Ø1
382Ø : PRINT "{DOWN}YOUR CHOICE -----";: K2=M1: GO
     SUB 277Ø: M1=K2                             :rem 1Ø9
```

```
383Ø IF M1=1 AND TT=Ø THEN CO=1: GOTO 394Ø:rem 163
384Ø IF M1=2 AND TT=Ø THEN CO=2: GOTO 398Ø:rem 17Ø
385Ø IF M1=3 AND CS>Ø THEN CO=3: GOTO 4ØØØ:rem 14Ø
386Ø IF M1=4 AND TT<>Ø THEN CE=Ø: GOTO 446Ø
                                          :rem 217
387Ø IF RS$="Y" AND SN>Ø THEN GOSUB 235Ø   :rem 95
388Ø IF M1=8 THEN CLOSE 8: GOTO 3Ø7Ø      :rem 111
389Ø IF M1=9 THEN 459Ø                     :rem 88
39ØØ PRINT "{2 UP}";: GOTO 382Ø            :rem 58
391Ø :                                      :rem 7
392Ø REM"--OPERATOR CHOICE--"             :rem 193
393Ø :                                      :rem 9
394Ø : RM$=MID$(RC$(CF),BB,LE): SL$=RC$(CF): IF CS
     <1 THEN SS$=SL$                      :rem 235
395Ø GOSUB 141Ø                            :rem 23
396Ø : IF ME=1 THEN GOSUB 128Ø: GOTO 4ØØØ  :rem 76
397Ø GOTO 363Ø                            :rem 216
398Ø : RM$=MID$(RC$(CF),BB,LE)             :rem 93
399Ø : GOSUB 133Ø: GOTO 363Ø              :rem 149
4ØØØ : IF SQ$="I" THEN GOSUB 248Ø: GOTO 4Ø2Ø
                                          :rem 213
4Ø1Ø IF AU$="O" THEN GOSUB 21ØØ            :rem 8Ø
4Ø2Ø : GOSUB 157Ø: IF D1$="Y" OR TT<>Ø THEN GOSUB
     {SPACE}255Ø                          :rem 247
4Ø3Ø SS$="": SL$="": M1=1: IF D1$="N" AND TT=Ø THE
     N GOSUB 149Ø                          :rem 76
4Ø4Ø IF ME=1 THEN ME=Ø: GOTO 394Ø         :rem 2ØØ
4Ø5Ø GOTO 363Ø                            :rem 2Ø6
4Ø6Ø :                                      :rem 4
4Ø7Ø REM"--AUTOMATIC MODE--"              :rem 115
4Ø8Ø :                                      :rem 6
4Ø9Ø : IF RS$="Y" THEN GOSUB 226Ø         :rem 178
41ØØ : GOSUB 141Ø: GOSUB 21ØØ: IF RS$="Y" THEN GOS
     UB 231Ø: BC=Ø: CS=Ø                   :rem 12
411Ø : IF TT<>Ø THEN CE=Ø: GOTO 363Ø       :rem 68
412Ø IF ME=1 THEN GOSUB 128Ø: ME=Ø          :rem 9
413Ø RM$=MID$(RC$(CF),BB,LE): IF RS$="Y" THEN 41ØØ
                                          :rem 119
414Ø GOSUB 157Ø: IF D1$="N" THEN GOSUB 149Ø: ME=Ø
                                          :rem 253
415Ø GOTO 41ØØ                            :rem 2ØØ
416Ø :                                      :rem 5
417Ø REM"--EXTRACT MODE--"                :rem 232
418Ø :                                      :rem 7
419Ø : PRINT "{2 DOWN}WHAT DATA ARE YOU LOOKING FO
     R";CR$;"AND WHERE IS IT ";             :rem 77
42ØØ PRINT "LOCATED:";CR$;"{DOWN}WHAT DATA STRING"
     ;: K1$=RM$: GOSUB 272Ø                :rem 16
421Ø RM$=K1$: LE=LEN(RM$): PRINT "{DOWN}IN WHICH F
     IELD";: K2=CF: GOSUB 277Ø             :rem 46
```

```
4220 CF=K2: PRINT "{2 DOWN}FIELD";CF;" = ";QT$;RC$
     (CF);QT$                                   :rem 242
4230 PRINT "{DOWN}SEARCH FOR STRING AT:"   :rem 238
4240 PRINT "{DOWN}{RVS} B {OFF} BEGINNING OF FIELD
     ";CR$;"{RVS} S {OFF} SPECIFIED POSITION"
                                                :rem 135
4250 : PRINT "{RVS} A {OFF} ANYWHERE IN FIELD";: K
     1$=PD$: GOSUB 2720: PD$=K1$                :rem 119
4260 IF PD$="A" OR PD$="B" THEN BB=1: GOTO 4290
                                                 :rem 84
4270 IF PD$<>"S" THEN PRINT "{UP}";: GOTO 4250
                                                 :rem 47
4280 PRINT "{DOWN}WHAT POSITION";: K2=BB: GOSUB 27
     70: BB=K2                                   :rem 14
4290 : PRINT "{DOWN}EXTRACT DEFINED OK";: GOSUB 26
     30                                          :rem 79
4300 IF KB$="N" THEN 3070                       :rem 207
4310 : GOSUB 2400: IF EN<>0 THEN GOSUB 2160: GOTO
     {SPACE}4310                                  :rem 1
4320 INPUT# 8,RC$(0)                             :rem 26
4330 : PRINT "{2 DOWN}EXTRACTING FROM FILE {RVS} "
     ;OF$;" {OFF}";CR$;PF$;CF                     :rem 8
4340 PRINT TAB(7);"RECORDS EXAMINED";CR$;TAB(7);"R
     ECORDS EXTRACTED"                           :rem 86
4350 IF LE=0 THEN LE=1                          :rem 135
4360 GOSUB 1330: IF TT<>0 THEN 4390              :rem 46
4370 IF RS$="N" THEN GOSUB 2480: GOSUB 1570
                                                :rem 249
4380 ME=0: GOTO 4330                             :rem 11
4390 : IF ME=0 THEN GOSUB 2550: GOTO 3630   :rem 82
4400 IF RS$="Y" THEN GOSUB 2550: GOTO 4420:rem 178
4410 GOSUB 2480: GOSUB 1570                     :rem 158
4420 : ME=0: GOTO 3630                           :rem 66
4430 :                                            :rem 5
4440 REM"--CONTINUATION OPTIONS--"              :rem 111
4450 :                                            :rem 7
4460 : GOSUB 2460: GOSUB 2160: GOSUB 2400: IF EN<>
     0 THEN 4480                                 :rem 92
4470 OF$=TF$: GOSUB 1990: IF EF=0 THEN 4490
                                                :rem 209
4480 : GOSUB 2550: GOTO 3630                    :rem 149
4490 : IF SE$="E" THEN 4330                      :rem 21
4500 IF AU$="O" THEN 4530                       :rem 221
4510 IF CS>0 THEN GOSUB 1330: GOTO 4110    :rem 12
4520 GOSUB 1280: RM$=MID$(RC$(CF),BB,LE): GOTO 410
     0                                          :rem 215
4530 : IF CO=1 THEN GOSUB 1330: GOTO 3960   :rem 80
4540 IF CO=2 THEN 3990                           :rem 97
4550 GOSUB 1280: GOTO 3640                       :rem 89
4560 :                                            :rem 9
```

```
4570 REM"--PROGRAM TERMINATION--"           :rem 14
4580 :                                      :rem 11
4590 : CLOSE 9: CLOSE 8: CLOSE 4           :rem 135
4600 : PRINT "{DOWN}PRESS {RVS} Q {OFF} TO QUIT OR
     --"                                    :rem 73
4610 PRINT "ANY OTHER KEY FOR MASTER MENU":rem 215
4620 GOSUB 2570: IF KB$<>"Q" THEN 4640     :rem 160
4630 PRINT "{RVS} PROGRAM TERMINATED {OFF}";: CLOS
     E 1: CLOSE 15: END                    :rem 138
4640 : PS$="DFH BOOT": OPEN 8,8,8,"0:"+PS$+",P,R"
                                           :rem 247
4650 GOSUB 1940: CLOSE 8: IF EN=0 THEN 4700
                                           :rem 200
4660 PRINT "{DOWN}TRYING TO LOAD {RVS} ";PS$;"
     {OFF}"                                :rem 119
4670 PRINT "{DOWN}INSTALL CORRECT DISK ";   :rem 66
4680 IF TY<>6 THEN PRINT "IN DRIVE # ";SD$;:rem 35
4690 PRINT CR$;"THEN, ";: GOSUB 2210: GOTO 4600
                                           :rem 234
4700 : CLOSE 1: CLOSE 15: PRINT "{DOWN}{RVS}LOADIN
     G ";PS$;" {OFF}"                        :rem 1
4710 POKE 824,248: LOAD PS$,8              :rem 232
```

Index

To order your copy of the DFH Disk call our toll-free US order line: 1-800-334-0868 (in NC call 919-275-9809) or send your prepaid order to:

DFH Disk
COMPUTE! Publications
P.O. Box 5406
Greensboro, NC 27403

All orders must be prepaid (check, charge, or money order). NC residents add 4.5% sales tax.

Send _____ copies of the DFH Disk at $12.95 per copy.

Subtotal $____________

Shipping & Handling: $2.00/disk* $____________

Sales tax (if applicable) $____________

Total payment enclosed $____________

*Outside US and Canada, add $3.00 per disk for shipping and handling. All payments must be in US funds.

□ Payment enclosed
Charge □ Visa □ MasterCard □ American Express

Acct. No. ________________________________ Exp. Date _______

Name ____

Address ____

City ____________________________ State ________ Zip _______

Please allow 4-5 weeks for delivery.

COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**.

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

Quantity	Title	Price*	Total
______	COMPUTE!'s Commodore Collection, Volume 1	**$12.95**	______
______	Commodore Peripherals: A User's Guide	**$ 9.95**	______
______	Creating Arcade Games on the Commodore 64	**$14.95**	______
______	Machine Language Routines for the Commodore 64	**$14.95**	______
______	Mapping the Commodore 64	**$14.95**	______
______	COMPUTE!'s First Book of VIC	**$12.95**	______
______	COMPUTE!'s Second Book of VIC	**$12.95**	______
______	COMPUTE!'s Third Book of VIC	**$12.95**	______
______	COMPUTE!'s First Book of VIC Games	**$12.95**	______
______	COMPUTE!'s Second Book of VIC Games	**$12.95**	______
______	Creating Arcade Games on the VIC	**$12.95**	______
______	Programming the VIC	**$24.95**	______
______	VIC Games for Kids	**$12.95**	______
______	Mapping the VIC	**$14.95**	______
______	The VIC and 64 Tool Kit: BASIC	**$16.95**	______
______	Machine Language for Beginners	**$14.95**	______
______	The Second Book of Machine Language	**$14.95**	______
______	Computing Together: A Parents & Teachers Guide to Computing with Young Children	**$12.95**	______
______	BASIC Programs for Small Computers	**$12.95**	______

*Add $2.00 per book for shipping and handling.
Outside US add $5.00 air mail or $2.00 surface mail.

Shipping & handling: $2.00/book ________

Total payment ________

All orders must be prepaid (check, charge, or money order).
All payments must be in US funds.
NC residents add 4.5% sales tax.

□ Payment enclosed.

Charge □ Visa □ MasterCard □ American Express

Acct. No.________________ Exp. Date________

Name________________________________

Address______________________________

City________________ State ________ Zip________

*Allow 4–5 weeks for delivery.
Prices and availability subject to change.
Current catalog available upon request.

A Powerful Data Base Manager

With their great speed, computers are perfect tools for keeping records. There's no need for file cabinets, file folders, and all the paper that goes with them. Businesses have long recognized the advantages of electronic data management.

Now you can record and manipulate any information on your Commodore 64 or PET. *COMPUTE!'s Data File Handler for the Commodore 64* contains a series of integrated programs that create a sophisticated file-handling system. The Data File Handler is clearly explained and easy to use.

This powerful system includes four machine language routines and a machine language sequential-file editor. The editor not only allows you to edit any sequential file, but also includes more than a dozen disk commands. There are explanations on how to use the machine language routines in your own BASIC programs, as well as easy-to-follow examples.

The Data File Handler is a versatile tool that can be used for almost any data storage and retrieval need. Some of the features included in this management system are:

- Single or multifield entries
- Up to 700 records or 14,000 characters allowed
- Twenty fields for entry and searches
- Merge up to 50 presorted files from as many as 50 disks
- Sorting on any field, at speeds of up to 650 records in five seconds
- Create and save multiple print formats
- Easy editing, adding, and deleting
- Change field order, concatenate fields, create new fields, or even delete fields
- Split files on any user-defined field
- Easily extract selected records for use in other files

All the programs you need are listed in this book. "The Automatic Proofreader," an error-checking program, makes program entry easy. You'll find this integrated system extraordinarily powerful and easy to use. It's the perfect package for data management.

ISBN 0-942386-86-8

DATA FILE HANDLER
for the Commodore 64

COMPUTE!
Books