$19.95

Inside SUPER UTILITY PLUS 3.0 / Published by POWERSOFT

POWERSOFT

POWERFUL SOFTWARE

from Breeze/QSD, Inc.

INSIDE SUPER UTILITY PLUS Series 3.0

INSIDE SUPER UTILITY PLUS 3.0

by

Paul Wiener & Gary Camp

Breeze/QSD, Incorporated
11500 Stemmons Freeway, Suite 125
Dallas Texas 75229
214/484-2976

Acknowledgements & Credits

TRS-80, TRSDOS are registered trademarks of Tandy Corporation
DBLDOS is a trademark of Percom Data, Inc.
DOSPLUS is a trademark of Micro-Systems Software
LDOS is a trademark of Logical Systems, Inc.
MultiDOS is a trademark of Cosmopolitan Electronics
NEWDOS, NEWDOS-80 are trademarks of Apparat, Inc.

First Printing, July 1983

Table of Contents

PREFACE TO THE REVISED EDITION

The TRS-80 world continues to change. New equipment and new software keeps appearing on the scene. Where once there were only one or two disk operating systems available for the TRS-80 computers, there are (at last count) at least 15 of the beasts, not counting the weird variants. And it seems that everyone and his uncle has got to own at least three or four different ones. Where once there was only the venerable Model I, we now have the Model I, the Model III, and the Model 4, along with a number of "work-alikes" that perform closely enough as to make little or no difference. In a way, this is a tribute to the dynamism of this little corner of the personal computing universe. But to those of us trying to support the TRS-80 computers, it's a bloody nightmare. We spend months of effort producing a software package that is as bug-free as we can get it, only to receive a phone call the day after it's released, with a plaintive voice on the other end saying, "It won't work on my TRX-80 with my UNODOS 3.0 operating system .... "

It's enough to drive a man to drink.

Super Utility Plus was originally written with an eye to creating some sort of software interface between the various disk operating systems on the market. At first it was simple, and the previous version of the program was more than up to the task. Thanks to the wizardry of Kim Watt, its author, Super Utility Plus was able to read a variety of diskette formats and operate comfortably with each one. Therein lay its greatest value. One could take disks created by two totally incompatible disk operating systems, each of which refuses to recognize the existence of the other, and transfer files freely between them using the same computer. Super Utility Plus was a godsend.

Let's admit one thing right here and now. Super Utility Plus is not a simple program, both from a programming standpoint and a user standpoint. Those who expect it to be "user-friendly" (whatever that means) are probably going to be disappointed. What is does provide the user, however, is enormous power. He can do things that normally aren't possible under the more beneficent and "user-friendly" environment of a standard disk operating system. But Super Utility assumes that the user knows (or at least, **thinks** he knows) what he's doing. And that may not always be true. That's why the original **Inside Super Utility Plus** was written. **Inside** gave users the information necessary to use Super Utility Plus with some confidence. It made available the experience of one person who had used Super Utility Plus for a good long time and had learned how best to work with it.

In the past year or so, however, the godsend has begun to fray at the edges. New systems and new formats continued to appear on the market, and Super Utility Plus began to look very inadequate. So Kim sat down and rewrote the program, and incorporated support for many of the new items. Double sided disks could now be operated on. The Radio Shack double-density adaptor for the Model I computer could now be freely used. Those weird Model I formats could finally be correctly handled. And so on ...

Shortly after the release of Super Utility Plus 3.0, we contacted Paul Wiener, who wrote the original **Inside Super Utility Plus**, and asked him to rewrite the book to reflect the changes in the new version of the program. It was not an easy task. When Paul wrote the original book, he could call on at least two years of experience with the Super Utility Plus program. Now he had to write about a new and significantly revised version, which had been out less than three months! How he must have hated us! But being the kind of person he is, Paul came through. He enlisted the help of a friend, Gary Camp, and between the two of them they spent long hours dissecting the program, interviewing by phone the people most closely associated with the project (to Ma Bell's enormous glee) and coming up with the necessary revisions to the book which you now hold in your hands. All of this while forsaking the wondrous enticements of California. Both of them deserve a medal for their efforts.

To those of you who waited so patiently for this book, we hope you find it worth the wait. As we have said from time to time, there is no substitute for experience, and that goes for using Super Utility Plus as well. But this book will help show you the way and assist in avoiding the various pitfalls which are so common in trying to use a program as complex and as powerful as Super Utility Plus.

And don't even THINK about the day when there'll be twenty-five disk operating systems available for the TRS-80s. Y'hear?

PowerSOFT Products
Dallas, TX
July, 1983

FOREWORD

Things were not working. I was trying to copy a program for a customer, but it was unreadable to the TRS-80 and no one could find another copy. At the software house where I worked, no one else was doing any better. One of my co-workers needed to edit a program, but the only copy of that program would not load. The person with the other TRS-80 could not even get it to boot. Things seldom run smoothly in the software business, but this situation had gone beyond our endurance. Tempers, including mine, were beginning to flare. Suddenly I heard a voice saying, "Need help?" It was Paul Wiener, the fellow who wrote this book. In his hand was a diskette. Noticing that my face was turning from red to purple, he walked up to me and said, "Problems?"

Before I could answer, he used the disk he was carrying to reformat the disk I was trying to copy. In a few seconds the whole disk had been reformatted without destroying the contents. Immediately the TRS-80 began to turn out copy after flawless copy.

"What was that disk?" I asked.

"Super Utility," he said. Before I could ask any more questions he had walked to the next computer and repaired the hash table of the disk that would not load the needed program. Then he quickly moved to the other side of the counter and in a few seconds repaired the disk that would not boot. In less than a few minutes Paul had put an end to two days of nonproductive computer frustration.

I was impressed. I suggested that they should start including a free cape with each copy of Super Utility. Kim Watt, the author of SUPER UTILITY -- now upgraded to SUPER UTILITY PLUS -- ignored that suggestion. He also ignored my only complaint. SUPER UTILITY was so versatile and had so many options that it was not always easy for people like me to use it. Fortunately I could take all my stupid questions to Paul who would patiently answer them. Unfortunately for most other people, they could not. Now Paul has collected the answers to all my stupid questions, added a lot of other material, and put together this book. Now you too can become a software superhero, but you'll have to provide your own cape.

William D. Allen
Measurement Systems Engineer
Physics Department
Simmons College
02 Feb 1982

Notes on Nomenclature

I've always been a disciple of Alice's caterpillar, making words mean what I mean them to mean by making them go where no similar words have gone before. Less intentionally, I have also been one of our century's leading exponents of creative spelling. My editors and friends, who have to interpret my day to day scribblings and utterings, will tell you to what extremes I've been known to carry these propensities.

Since this book is being published by a sane publisher, I've allowed Microproof to veto some of my more original concatenations of letters. With regard to inventing new words, using old words in strange ways, and unilaterally enacting new laws of grammar, I've abandoned departs of speech and restricted myself to the conventions--as much as possible. For instance, I'll be using the smaller-than and greater-than symbols ("<" and ">") as brackets to indicate keys which you are supposed to press. So when you see a statement like 'Press <1><ENTER>', you'll know you're intended to press the key that says "1" and the key that says "ENTER."

Sometimes I've used the more or less standard shorthand of saying "Enter <Y>" to mean press <Y> and then press <ENTER>. "Press," "type," or 'key <Y>', on the other hand, indicates that you're not supposed to press <ENTER> afterwards.

In some instances, I've allowed myself to stretch things just a little. I've used single quotes instead of double quotes to indicate something that Super Utility puts on the screen. So 'HIGHSPEED=Y' means that when you look at the screen, you will see HIGHSPEED=Y but no quotes.

But I've also used single quotes, rather than doubles, to escape a dilemma--the dilemma being that it's a grammatical requirement to enclose certain punctuation within quotes, even when logic and clarity require placing the punctuation outside the quotes. So I'll tell you to type '1,1,1', rather than "1,1,1." Notice how in the former, the third comma is outside the single quotes, but in the latter, the period is inside the double quotes where it creates the following ambiguity: do I or don't I want you to type the period? You'll notice that I've already used that escape hatch earlier in this preface.

In certain cases, I have gone ahead and coined new terms after all. I've worked up a taxonomy of disk errors. You'll learn about compliant errors and stubborn errors. You'll learn about two kinds of peccadilloes, and about nightmares. The less said about nightmares the better.

"Farkled" is a word you'll be seeing a lot of in this book. It's used by Jesse Bob Overholt in his Alternate Source columns. It's usually used in conjunction with munched disks. It means "messed up."

I sometimes use the term "TRSDOS III" when I get tired of spelling out "Model III TRSDOS."

A last word about words: this book is about Super Utility Plus. Much of the information simply doesn't apply to plain old Super Utility. But the manuscript looked

so cluttered up with "Plus"'s that I pulled all of them out at the last minute. So when you see Super Utility, remember, I mean Super Utility Plus!

A last word: This book is not intended to be a substitute for your Super Utility instruction manual. My purpose is to fill in the background needed to use Super Utility effectively, describe some extended application techniques which aren't covered in the manual, and to clarify a few points which were covered, but have been generally misunderstood.

This book won't be very useful to you if you don't use your Super Utility manual as well. Please keep it handy and refer to it whenever necessary.

Chapter I
TECHNICAL INTRODUCTION

Though it's not strictly necessary, knowing a little about the way information is stored on and retrieved from your disks can help you understand Super Utility Plus and use it more effectively. All TRS-80 disk I/O takes place through a special piece of hardware manufactured by the Western Digital Corporation. It's called the Floppy Disk Controller chip (FDC). In this chapter I will describe some of the FDC's characteristics, functions, and peculiarities. Wherever appropriate, I will indicate which features of Super Utility Plus allow you to observe or control the FDC functions, or otherwise capitalize on their results. Detailed instructions on how to actually use Super Utility Plus to achieve these effects will be found in various other chapters.

Some of the information in this chapter may seem a little technical. If you're not into electronics or machine language, don't worry. You can probably follow this anyway. I don't know the difference between a casistor and a repacitor [sic], but I can still relate to the concepts presented here. Even if you can't or don't want to acquaint yourself with these technicalities, it might be a good idea for you to skim this section anyway. The references to other chapters will help you zero in on the sections of this book which are more pertinent to your interests.

The FDC has the ability to translate serial (one bit at a time) data to parallel (8 bits, or one byte, at a time) data and parallel data to serial data. This is necessary because the Z-80 handles data in parallel, but data is stored on the disk serially. The FDC also contains what amounts to a dedicated microprocessor which performs certain control functions and computations that the TRS-80's Z-80 isn't fast enough for.

Though the FDC is a wonderful device, there are certain facets of its internal logic that can create difficulties. The only way around them seems to be convoluted programming. In this book, I will not try to explain why the FDC chip does things the way it does. That will be accepted as part of the physical laws of the TRS-80™ universe. But, where pertinent, I will explain what the FDC does so you will understand why Super Utility Plus (and other software) is the way it is. This knowledge will also help you use Super Utility to the fullest advantage.

The division of disk space into tracks, granules, and sectors has been adequately dealt with in numerous popular books. Tandy's TRSDOS & Disk BASIC Reference Manual (that's your good ol' Model I TRSDOS users' manual) has a brief, but intelligible, treatment of the topic in section 2, Mini Disk Operation. There are even a couple of nice diagrams. Page 74 (Model III Disk Operator's Manual) also mentions the subject, but there aren't any diagrams. If you would like more information on the subject of this chapter, see the informal bibliography near the end of this chapter.

Briefly, the surface of a disk is divided up into concentric rings, known as tracks, cylinders, lumps, or even records. The oldest TRS-80 disk drives could only read and write to 35 such tracks. Most newer drives can handle at least 40, because the read/write head can move farther in toward the center of the disk. Both the 35 and 40 track drives use a 48 Tracks-per-inch (TPI) format.

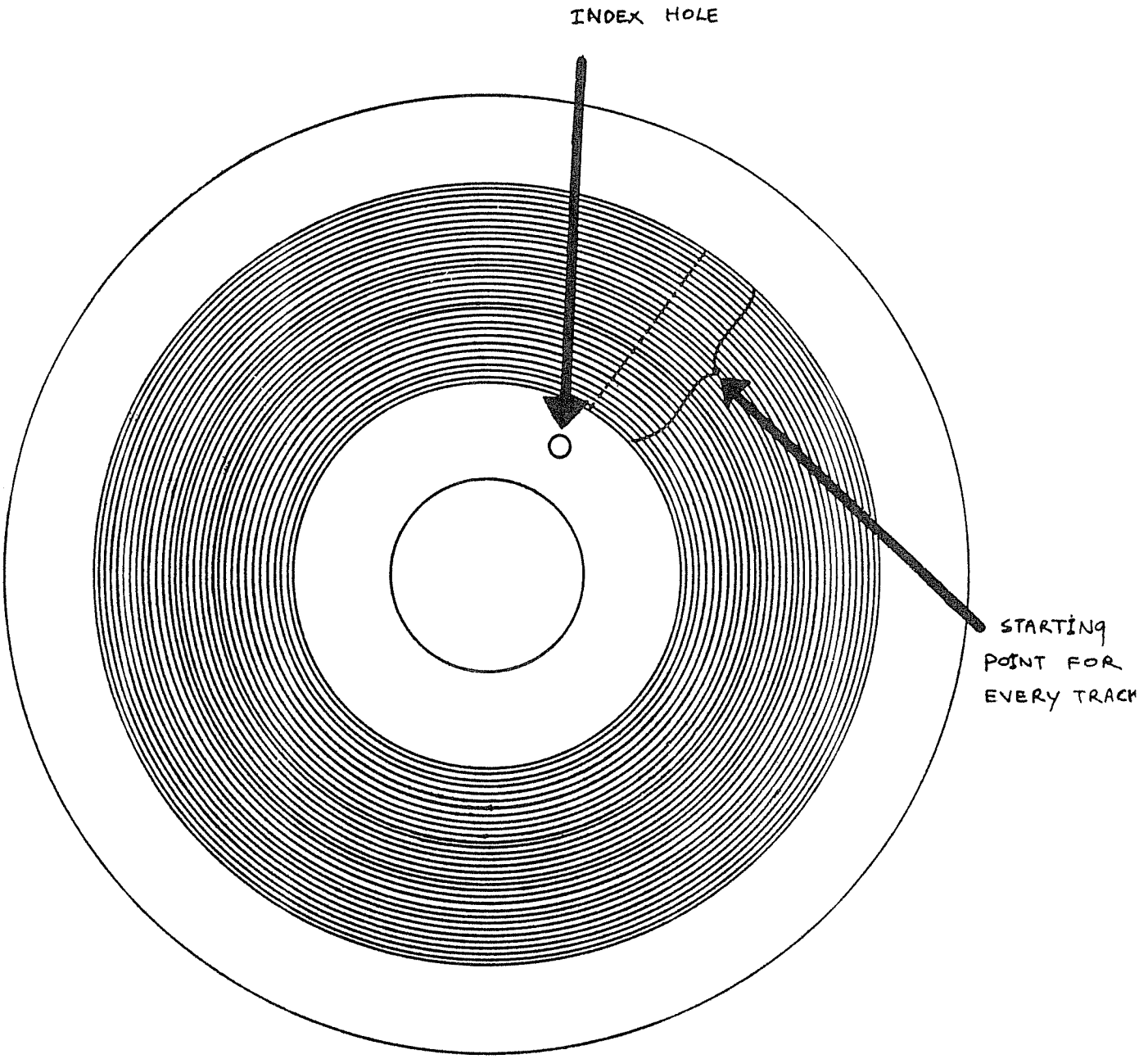


FIGURE 1

Eighty-track drives are now becoming popular, but they access tracks only half as wide as those used by 35- and 40- track drives. Thus, disks used by 80-track drives may be said to have double track-density, since they contain twice as many tracks per inch as do disks used by standard drives. This should not be confused with the more conventional type of double-density, which puts more data in each track. (Editor's note: Some disk drive manufacturers have taken to using the term "quad-density" which refers to a combination of both "double track-density" and the conventional "double-density"; hence a "quad-density" drive is usually an 80-track drive capable of double-density)

Since a track written by an 80-track drive is only half as wide as one written by a forty-track drive, 40-track drives can't read diskettes which were written by "80-trackers". However, 80-track drives can read disks created by 40-track drives. All the information is present and detectable. However, when the 80-track drive tries to step to another track, it won't move the head far enough to reach it. Therefore, when an 80-track drive is dealing with a 40-track disk, the system software must move the head twice, instead of once, to reach the next track. Super Utility's Double-step feature accomplishes this. You may select it from the CONFIGURATION mode.

Using Double-step, 80-track drives may read data written by 40-track drives, and they can do so reliably. However, it is risky to attempt to write to 40-track diskettes in 80-track drives. If you use Super Utility's Double-step feature to write, you will probably succeed in creating data which can be read back--only while the target disk is still in the 80-track drive! If you intend to use the disk in a 40-track drive again, you may be heading for trouble. See the MISCELLANEOUS Chapter for a more detailed account of this topic.

Caution: As stated earlier, standard 35- and 40- track drives use a 48 Track-per-inch (TPI) format. Eighty-track drives normally use a 96 TPI format. This is why a Double-step feature works. One TPI is exactly double the other TPI, so moving the head twice gets to the same place. However some drives (especially some old Micropolis 77-trackers) use a 100-TPI format. Disks written on such drives will be incompatible with normal drives, with or without the Double-step feature.

The TRS-80 uses soft sectored diskettes, which means that there is only one index hole on the disk. This hole is detected by an optical sensor within the disk drive. It defines the starting point of each track for the system (see figure 1).

Tracks are divided into arbitrary units called grans. In what I will refer to as the TRS-80 Single-density Standard (T1S), each track has two grans. Grans are further divided into sectors. In T1S, each gran has 5 sectors. Thus, each track contains 10 sectors (see figure 2). In T1S, each sector contains 256 bytes of user accessible information, plus some special data normally used only by the system. Super Utility allows you to monitor and alter all this information--both user and system.

Other arrangements are possible. Non-standard, or "protected" disks are created by varying this type of disk organization. Data organization on a disk may be altered by several other factors which will be described later. Even some "standard" disks may not conform to the picture just painted. For instance, some double-density disks (i.e, DOSPLUS, LDOS) have six sectors per gran and three grans per track while others (TRSDOS III) have six grans of three sectors each per track. NEWDOS-80 and

DBLDOS use lumps instead of grans. Lumps are explained in the MISCELLANEOUS Chapter.

As I insinuated earlier, a disk contains a certain amount of information other than "user data." The bulk of this data is placed on the disk during the formatting process. Mostly, it consists of special gaps and identification (ID) "fields" which identify the various tracks and sectors for the FDC.

For a moment, consider what a remarkable task the Floppy Disk Controller performs as it reads information from a disk. As mentioned earlier, the data is recorded on the disk as a series of tiny magnetic pulses. As the disk spins, over 100,000 of these dots fly by the disk drive's read/write head every second. To interpret the data properly, the FDC must maintain perfect synchronization with the flow of pulses . Without perfect sync, bits could be lost. The loss of a single bit would cause all further data to be out of place, or "shifted", in memory. Such shifted data would be useless, as the computer would interpret it incorrectly. The FDC can usually recognize this situation, however, and will signal a "parity error" if it occurs. One parity error in a text file is not serious (you can spot it and correct it); the same error in a machine language file can be fatal. In a BASIC file, it's a tossup. You may be able to recognize it and correct it, or you may not.

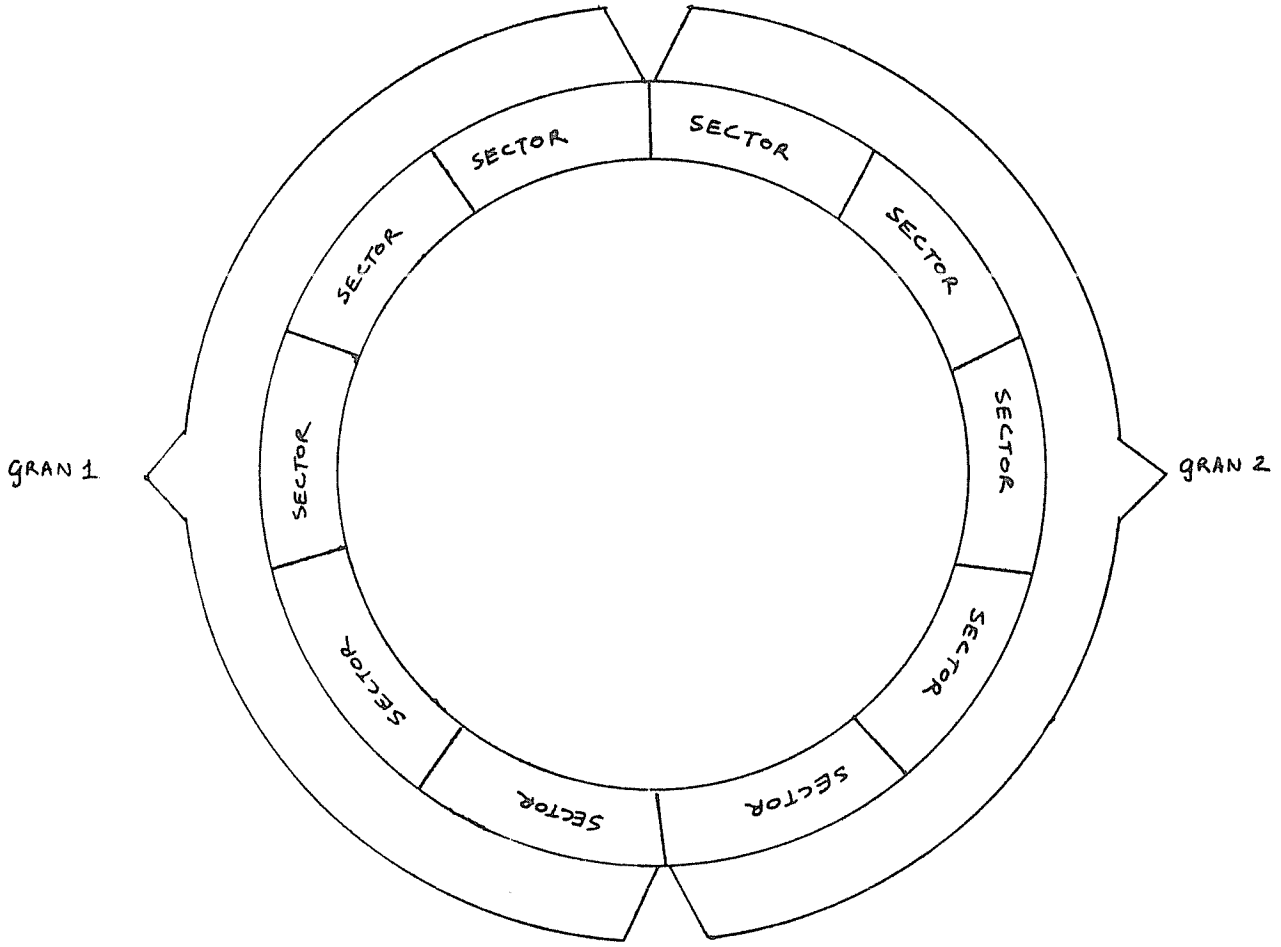
To help achieve correct synchronization, the FDC uses its internal clock. In the TRS-80, the FDC clock runs at 1 mhz. With the help of its clock, the FDC puts a series of evenly spaced pulses (called clock bits) on the disk. When writing data to the disk, a one bit is represented by placing a data pulse between two successive clock bits. A zero is represented by leaving the interval between two clock bits blank.

When the system reads back the information, the FDC simply locks into the flow of clock bits. This lets the FDC look at the slot between each successive pair of clock bits and interpret blank slots as zeros, and slots with pulses as ones. Actually, it's not quite that simple, but almost. An FM (Frequency Modulation) method is used, since the frequency of the bit flow will be higher where there is a data bit between clock bits than where there isn't.

You might think that all this would assure perfect synchronization. Unfortunately, it doesn't, largely because of small variations in the speed of spinning disks. Such variations can nullify the high precision of the FDC's clock, so that in effect, the FDC which wrote the data was timed differently from the FDC which tries to read in the data, even if it's one and the same chip!

Another cause of trouble is that the inner tracks are physically shorter (though logically equal in length) the outer tracks. In other words, on the inner tracks, the same number of clock and data bits must be jammed into a shorter band. This can make it more difficult for the FDC to stay in sync and separate the clock bits from the data bits.

As you've probably heard, data separators are available to cure these problems. An important part of their read circuitry is a phase locked loop widget which dynamically adjusts the clock rate of the FDC to match the flow of clock bits on the target disk. Such external data separators are quite effective. Yet, even with one (and especially without one), it is still possible for errors to occur. Consequently, certain other measures are taken to insure accuracy.



2 GRANS PER TRACK

5 SECTORS PER GRAN

10 SECTORS PER TRACK

FIGURE 2

Every sector on the disk has a special ID header. This header is preceded by a "gap" which separates it from the previous sector. The sequence of gap and ID header makes it easy for the FDC to lock into the bit flow for that sector. Obviously, the longer the sector, the more likely the FDC is to drift out of sync during a read. Thus, it is important to keep the sectors sufficiently short to ensure positive sync.

The people who devised the original TRS-80 system software apparently felt that dividing a track into 10 sectors, resulted in short enough sectors (256 bytes a piece) to minimize errors. Almost all other TRS-80 system software programmers have followed suit.

Super Utility allows you to create special disk formats. If you wish, you can create disks with fewer (or more than) than 10 sectors per track. You may even fill an entire track with only one giant sector. If you do so, however, you should be aware that you're risking a higher probability of error. You can create these unusual tracks with Super Utility's SPECIAL FORMAT, or BUILD FORMAT TRACK options.

As you have seen, a major purpose of dividing each track up into sectors is to give the FDC frequent opportunities to resynchronize. In order for this scheme to work, each sector must start in a way which the FDC can easily detect and lock onto. Here's how that is arranged.

Every sector contains two subfields which I shall refer to as the header subfield and the data subfield (see figure 3). The header subfield (sometimes called the sector ID block or ID pack) is a seven byte segment that starts the sector (Editor's note: this is not the same as the "Pack ID" referred to by TRSDOS 2.3's BACKUP utility when you attempt to do a backup to a previously formatted disk with a different name. That "Pack ID" refers to the diskette's name/master password combo). It consists of a one byte address mark followed by four one byte designators and a two byte CRC. The address mark is always FE hex. The header address mark is also called an ID address mark. Each of the four designators contains pertinent information about that sector--namely the track number, head number, sector number, and sector length.

Notice that there is no separate identifier for the track as a whole--the track number is specified by the first designator in every one of its sectors. The head designator is useful with double-sided diskettes, or hard disks with more than one track per "cylinder." On all "normal" TRS-80 diskettes, the head designator should contain zero (0). If a disk were double-sided, that is, formatted on both sides, the head designator might read one (1) to indicate a sector on the back side of the disk (Some DOS'es, however do not use the head designator). And while we're on the subject, it should be noted that the new Super Utility Plus 3.0 does offer support for double-sided drives -- but not for all the DOSes that it supports. Double-sided drives created by DOSPLUS, LDOS and MULTIDOS can be operated on.

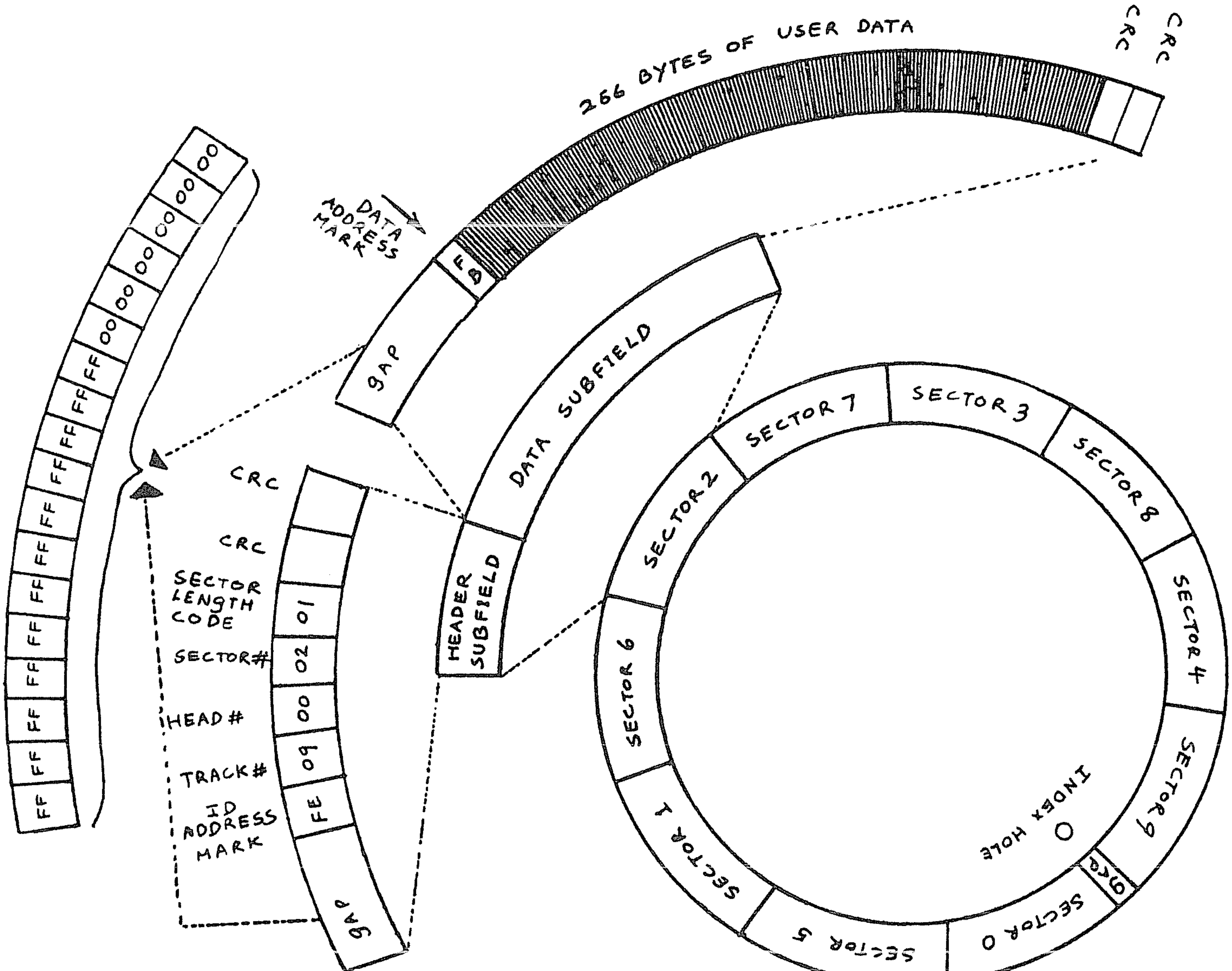


FIGURE 3

The sector-number designator is self explanatory. The sector length designator contains a code that indicates the number of data bytes in the sector. With standard TRS-80 disks, it will always be a 1, indicating 256 bytes of user data. This is what is sometimes referred to as "IBM format." In the IBM scheme of things, a length byte of 0 indicates that there are 128 bytes of data in the sector, while a length of 1 indicates 256 data bytes. In the "non-IBM" convention, the length byte multiplied by 16 gives the actual number of data bytes in the following sector; however, a length byte of 0 indicates that there are 4096 bytes to follow. The header field is put on the disk when the disk is formatted, and is never altered by the system--unless the disk is reformatted.

The second, or data, subfield comprises the main body of the sector. It contains the actual user data. The data subfield is rewritten every time data is saved to that sector.

Each of the two subfields is preceded by a "gap", which typically consists of 12 FF hex bytes followed by six zero bytes. I use the word "typically" advisedly, because there is a great deal of variation from this norm, both with respect to the number of bytes in the gap and the values of those bytes.

The data subfield, like the header subfield, starts with a one byte "address mark." When these address marks are written to disk, the frequency of the accompanying clock signal is changed by dropping some of the pulses. This makes the address marks unique. Therefore, the FDC can easily and quickly find address marks, and will not be fooled by sequences of data marks which mimic address marks (the data can't fake the change in the clock pulses).

For header subfields, the address mark used is FE hex. The address mark used for the data subfield is often referred to as the Data Address Mark (or DAM). The actual byte used varies, depending upon whether a Model I or Model III is being used, whether double or single density is employed, what operating system is in use, and whether the sector in question is a normal one or part of the disk directory. To simplify this discussion, we'll assume a garden variety situation. This means a Model I running TRSDOS in single density. For other situations, see sigure 4. The DAM for shis "garden variety situation" is FB hex for standard sectors, or FA hex for directory sectors. The directory is given a different DAM to help the disk operating system in locating it.

Both the header and data subfields end with a two byte Cyclic Redundancy Check (CRC). CRC's are used to detect errors which occur despite all precautions. A CRC performs much the same function as a parity check or checksum, but is much more reliable. Essentially, it is a 16 bit number derived by applying a polynomial evaluation to every bit in the associated subfield. In other words, the CRC at the end of any subfield is derived by an operation on all the other bits in that subfield. This computation is one of the super-fast ones that the Z-80 leaves to the dedicated FDC microprocessor.

Figure 4
Directory DAM Chart

DOS	Model I	Model III
TRSDOS 2.3	FA	**
TRSDOS 2.7	F8	**
TRSDOS 1.3	**	FB
LDOS 5.1x	F8	F8
DOSPLUS	F8	F8
MultiDOS	F8	F8
NEWDOS/80 Sgl. Den.	FA	**
NEWDOS/80 Dbl. Den.	F8	F8

When the FDC reads back a sector, it recomputes the CRC bytes and compares them to those that were written on the disk during the last formatting or data-save operation. Any disagreement causes the FDC to flag an error (some disk operating systems respond to this by displaying "parity error" messages although, strictly speaking, what has occurred is not a parity error but a CRC error -- Ed.).

A standard disk track consists of ten sectors, separated by gaps, as described. In addition, there is an extra gap between the track's starting point, as defined by the disk's index hole, and the beginning the first sector. This gap is similar to the others, but normally a little longer.

Super Utility's ZAP package contains options to READ ID ADDRESS MARKS and ALTER DATA ADDRESS MARKS. These give you good control over the address marks on your disks. READ ID ADDRESS MARKS also can display CRC values as read from disk, and tell you whether the currently calculated CRC's agree with those read. Also, Super Utility's CONFIGURE SYSTEM option ensures that Super Utility's own use of address marks will be consistent with that of the operating system of any target disk.

As I said, each sector header subfield contains a track and sector number. It is possible to give any sector a "false" ID, simply by putting non-standard information in its header. Thus, a sector may be the first of ten physical sectors on the first physical track on a 40 track disk, and yet have a header that says it is the 108th sector on the 99th track.

In fact, DOS never labels physically adjacent sectors consecutively. For better disk I/O, it uses the following pattern: 0,5,1,6,2,7,3,8,4,9. As you can see, this is composed of an interleaving of the two following patterns: 0,1,2,3,4, and 5,6,7,8,9. Since such an arrangement is normal, it is rarely referred to as false labeling.

Tracks whose numbers don't reflect their actual physical position on the disk, however, are another matter. Tracks may not only be mislabeled, but each one of a track's sectors may be given a different track number! This is why in some displays, Super Utility shows two track numbers. In the ZAP module, they're labeled TRACK

and TRUE. TRUE shows the track number as read from the disk; TRACK shows the number of the track according to its physical location. In Read ID Address Marks, SOURCE shows the actual physical track number, while TRACK shows the track number as read from the disk.

As you can see, the designers of the FDC gave a lot of thought to error prevention and detection. In the DISK ERRORS Chapter, we will see what kinds of disk errors can occur, and how to detect and correct them.

INFORMAL BIBLIOGRAPHY

TRS-80 Disk and Other Mysteries, by Harvard Pennington, published by IJG, seems to be considered the classic in the field. I understand an updated version is being prepared which will have information on double density, Newdos 80, and other new mysteries. So if you're thinking of buying TRS-80 Disk and Other Mysteries, it may be worth your while to wait for the update.

If you can get by with a straightforward presentation of the subject, you might look into book 2 in the Mystery series, **Microsoft BASIC and Decoded and Other Mysteries**, by James Farvour. It's an information, rather than explanation, oriented book. Its ten or so pages on disk I/O present a concise recap of much of the information in Harv's book, with some new data to boot.

Other good books to have handy are **Disk Interfacing Guide** by Bill Barden and **Pathways Through the ROM**. The latter is anthologized and edited by George Blank, and contains material by Robert Richardson, Roger Fuller, John Phillips, George Blank, John Hartford (not, as far as I know, the singer), and the Western Digital Corporation. It includes Supermap (a comprehensive TRS-80 Model 1 memory map), the technical manual for the 1771 (the Model 1's FDC), and an essay on DOS containing practical information which I haven't seen published anywhere else.

An excellent description of Model I TRSDOS formatting is presented in John Brule's **Some Disk Considerations**, in **The Alternate Source**, Volume II, number 3. Also, the **LDOS Quarterly**, Volume 1, number 1, contains an enlightening article on DAM's by Roy Soltoff.

The latest book in the IJG "Other Mysteries" series, **Machine Language Disk I/O and Other Mysteries** by Michael A. Wagner, is a good manual on how to perform disk I/O at the lowest (machine language level). This book gives you a closer view of what I just discussed here, with details on how to actually implement the various items.

Chapter II
CONFIGURATION

You get into the configuration module by entering <9> from the main menu. As soon as you do so, the screen will be filled with a display called the configuration table. Configuration may seem complicated to some new users, but it is actually very simple. It's a way of telling Super Utility about factors which can vary from use to use.

Some factors may be different from disk to disk, and others from computer system to computer system. Those which vary from disk to disk include the DOS contained on the disk, the number of formatted tracks, sides and the density (single or double) of the disk's data.

Factors which may differ from system to system include the characteristics of your printer (if you have one), the absence or presence of a high speed modification, the number of disk drives you have, and the speed characteristics of each.

Super Utility needs to know whether your printer needs a line feed after each carriage return, and if it's capable of printing lower case letters and/or TRS-80 graphics. If you printer handles graphics, Super Utility needs to know whether it's an Epson MX-80 or not (MX-80's may require the computer to adjust the graphics characters before sending them to the printer).

If you have a high speed mod, Super Utility wants to know three things. 1) whether you want high speed turned on or off; 2) how to turn it on; and 3) how to turn it off.

Super Utility also wants to know if you have a Doubler installed in your Model I or not, and whether it is a Radio Shack unit or someone else's.

If you have an 80 track disk-drive, and want to use it with disks that were formatted on a 35 or 40 track drive, Super Utility needs to know about that, too. Naturally, this information is entered via the configuration module. You can also use the configuration module to inform Super Utility that you want a drive software write-protected. This affords similar protection to that obtained by putting a write protect tab on a disk. Once you've told Super Utility to write protect a drive, it will refuse all requests to write to that drive until you cancel the write protect status.

The new version of Super Utility Plus assumes certain things in the configuration table that may not be immediately obvious. Part of the reason for this was simple lack of space on the screen, so you have to understand the configuration process thoroughly to set the tables up correctly. Figure 5 depicts a typical CONFIGURATION table. The configuration shown is similar, but not identical, to the one in the Super Utility manual. It's possible to change most of the data in the table, simply by typing in different data. We'll get to that shortly. But first, lets analyze the information on line one of this sample:

=> DUAL=N GRAPHICS=N LOCASE=N LINEFEEDS=N DOUBLER=R SPEED=N

The "=>" at the beginning of the line means that this is the "current" line--in other words, this is the line which may now be altered by typing in new data. Note that there are 6 items of information, separated by commas, on this line:

1 DUAL=N
2 GRAPHICS=N
3 LOCASE=N
4 LINEFEEDS=N
5 DOUBLER=R
6 SPEED=N

In addition, you can enter two more items which are not prompted for: the code to turn on your high speed clock, if you have any, and the code to turn it off, in that order. Let's call these items 7 and 8. Thus, items 6,7, and 8 have to do with controlling any high speed modification which may have been installed in your computer. Let's consider the meaning of each.

```
#  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  #
.  **  SUPER-UTILITY +  ##  VERSION 3.1A  ##  BY: KIM WATT  ##                .
.       (C)(P) 1983 BREEZE/QSD, INC.  --  DALLAS, TEXAS                       .
#  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  #
.  __   CONFIGURATION  __                                                     .
=>DUAL=N GRAPHICS=N LOCASE=N LINEFEEDS=N DOUBLER=R SPEED=N                    .
.  +:0 T1S'  PTKS= 35 RTKS= 35 DIR= 17 STP=3 RDLY=4 WDLY=4 WP=N              .
.           DD=S DD=S LSD=0 HSD=09 LSD=0 HSD=09 S/G=5 G/T=2 DD=S              .
.  +:1 T3D'  PTKS= 40 RTKS= 40 DIR= 17 STP=3 RDLY=4 WDLY=4 WP=N              .
.           DD=D DD=D LSD=1 HSD=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I              .
.  +:2 T3D'  PTKS= 40 RTKS= 40 DIR= 17 STP=3 RDLY=4 WDLY=4 WP=N              .
.           DD=D DD=D LSD=1 HSD=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I              .
.  +:3 T3D'  PTKS= 40 RTKS= 40 DIR= 17 STP=3 RDLY=4 WDLY=4 WP=N              .
.           DD=D DD=D LSD=1 HSD=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I              .
.  ? #_____________________________________________________                   .
#  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  #
```

Figure 5 - The Configuration Display

I guess I'd better preface my discussion of speed configuration with this comment: If you have a normal TRS-80, with no high speed clock or similar modification, it doesn't matter much how you configure items 6, 7, and 8. But you should keep item six set to 'SPEED=N', that is, "NO" high speed clock. Also if you have a computer that does not use software (programmable by you) speed control, such as the LNW, you should still set SPEED=Y to make sure Super Utility adjusts its timing loops to compensate for the faster I/O speed (Note: DON'T do this on the Lobo MAX-80. The MAX-80 version of Super Utility is already adjusted for the faster CPU speed of the MAX-80, so leave the SPEED setting at N). But you can still get into trouble. One example is the way my LNW goes to reverse video when the SPEED=Y is set -- because, the code sent . . . is the same as the code used for video control by the

LNW. Therefore, it's advisable to send 00 for both on and off codes. That is, NOP (00) instructions are substituted for items seven and eight (explained below). Only one NOP is needed. The speed setting can then reflect the true speed, as set on the computer. For my LNW, it's controlled by the way I set a switch on the back. You must determine the operating speed of your computer (from your documentation) and set **SPEED** accordingly.

'SPEED=Y' indicates that your computer has a high speed mod and you want Super Utility to keep high speed turned on. Here's why Super Utility may need to know you're running high speed. If your computer is in high gear during disk I/O, the timing may be thrown off enough to cause errors. Some "intelligent" high speed mods automatically restore normal speed during disk I/O and then go back into high as soon as the I/O is finished. But some do not. If yours doesn't, Super Utility can adjust its timing loops to secure reliable I/O. It will also adjust timing loops controlling cursor flash and key repeat. High speed mods usually have software commands which you can use to change speeds. Super Utility wants to know what those commands are for your system. Once it knows them, you can turn the high speed on and off from the key board by configuring 'SPEED=Y' or 'SPEED=N'. If you're not sure what the on and off commands are for your system, refer to your speed mod documentation . Most of the popular high speed mods are controlled by output to port 254 decimal. In the current example, the high speed in our hypothetical computer is turned on by outputting one (1) to port 254. It's turned off by outputting zero (0) to port 254. If you were controlling the speed from BASIC, you'd enter OUT 254,1 to turn on high speed, and OUT 254,0 to turn it off. Super Utility, of course, uses Z-80 machine language rather than of BASIC. In Z-80 code, there are many ways to send a value to a port. The sequence 3E01D3FE is one of them. Here is a disassembly of the sequence.

```
3E01      LD      A,1
D3FE      OUT     (0FEH),A
```

In the first line 3E says to load the A register. 01 is the the value with which it's to be loaded.

In the second line, D3 indicates that the value in the A register is to be sent out to a port. The FE means that the port to be used is FE (hex) or 254 (decimal). So executing the machine instructions 3E01D3FE results in a one being sent to port 254. Now, let's take another look at '3E01D3FE'. It consists of four hexadecimal bytes. 3E 01 D3 FE is the sequence of machine code which we just looked at. It sends a one to port 254. In this new version of Super Utility Plus, you simply enter these HEX codes after your reply to the "SPEED" prompt. You'll notice that there are no prompts for the ON and OFF codes, unlike the older versions. They'll be accepted anyway. You may enter up to 8 hexadecimal values for clock control. However, you no longer need to pad them out with 00's in case they're less than 8. The program now does that for you.

Next, let's look at '3E00D3FE', the sequence required to turn the high speed clock OFF. Notice that the OFF sequence is exactly the same as the ON sequence, except that the 01 byte after the 3E has been replaced with a 00 byte. This specifies that high speed is to be turned off by outputting a zero to port 254.

If your high speed mod requires different instruction sequences to turn it on and off, you must determine what they are. If you have no knowledge of machine language, this may seem intimidating, but in most cases it's easy. Port output is almost always used. The only varying factor is the number of the port and the values to be sent to that port. If your high speed mod requires a value other than one (01) to turn it on, simply substitute that value for the 01 in '3E01.....'. If a byte other than zero (0) turns off your high speed, just replace the 00 in '3E00.....' with the byte that slows down your system. Always remember to use a two digit hexadecimal byte.

If you need to use a port other than FEH (254 decimal), substitute the port number for the 'FE's in '3E01D3FE....' and '3E00D3FE....'. Again, remember to use two digit hexadecimal bytes to represent the port numbers.

If you have a high speed mod whose on and off instructions are not similar to the ones in this example, consult the high speed mod documentation to learn just what machine code sequence is needed to do the job. Remember, Super Utility Plus allows you up to eight bytes for each instruction sequence. As you have seen, that's more than enough for what's typically required.

By the way, some people have found that on their systems, Super Utility works fine at high speed, even without adjusting timing loops to compensate for the speed-up. So it may be worth your while to experiment and find out what works for you. We'll get back to this in a few paragraphs. Now that we've got most of the high-speed/low-speed stuff out of the way, let's go back and cover what we skipped:

'DUAL=N'. Super Utility allows you to do a screen print at any time by pressing <SHIFT><CLEAR>. But some times, when using Super Utility you may want a printed record of all the information Super Utility presents to the screen. In such cases, turning on the DUAL mode is more convenient than frequent screen printing. The effect of DUAL is similar to the Model III's DUAL command, or LDOS's LINK *PR TO *DO statement. It causes all significant screen output to go to the printer as well. In Super Utility's configuration table, 'DUAL=Y' means DUAL is turned on (simultaneous screen and printer output) and 'DUAL=N' means that dual is turned off.

Unlike the previous versions of Super Utility Plus, this new version automatically updates the configuration table depending on the disk that it finds in a drive. Thus the old SAVE parameter is no longer available, or needed. For instance, suppose drive 0 is configured for a single density TRSDOS disk. But you decide to use drive 0 to do some zapping on a double density LDOS disk. You want to read track 5, sector 0. You tell Super Utility to do so, using the proper override command (e.g, 0L3D=40). A minute later, you want to read track 10, sector 1 of the same LDOS disk. You no longer have to repeat the override command, because the first time you did so, Super Utility Plus altered the CONFIGURATION table to reflect an LDOS disk instead of a TRSDOS disk. To clarify this further, suppose you originally set the drive zero line of the configuration table so that it starts with something like this:

+0TS' 35

(The TS indicates single-density Model I TRSDOS). Later you use the override command 0LD=40 to read a double density LDOS disk. After reading the LDOS disk, if

you go back to the configuration mode, you will see that the line for drive zero now starts as follows:

+0LD' 40,

The TS has changed to LD to reflect that a double density LDOS disk was the last disk read in drive zero. You may now perform further operations on the LDOS disk without re-entering the override command. However, you will HAVE to use another override command if you want to return to reading your single-density TRSDOS disk.

Sometimes, when you go to the configuration display, it may have changed, even though you haven't used any override commands. This can happen because Super Utility's automatic density recognition is also capable of updating the configuration information. Also, certain operations which involve a disk's directory are capable of passing a new track-count and DOS type back to the configuration module.

Sometimes, as a result of an automatic configuration change, the DOS specifier in the configuration table will be replaced with an asterisk ("*"). "*" means "Unknown DOS." In other words, an asterisk in the configuration table indicates that Super Utility is no longer certain of the disk's resident operating system.

To change an entry in the configuration table, the "=>" pointer must be pointing at the line you want to change. When you first enter the configuration module, the arrow will be pointing at the first line--the one we just discussed. To alter the line, simply retype it.

At the bottom, of the configuration table is a question mark ("?") followed by a dotted line. As you type the new configuration entry into Super Utility, your keystrokes will be echoed on the dotted line, so you can be be sure to spot any typing error. Correct errors by backspacing and retyping. Notice that you only type the variables. In other words, you don't type "SPEED=N", only "N,". Here's a line that would change each entry in the first line of our sample:

N,Y,Y,N,X,N,O=3E02D301,F=3E03D301

This will result in the following update of the configuration table:

DUAL=N, GRAPHICS=Y, LOCASE=Y, LINEFEEDS=N, DOUBLER=X, SPEED=N

This says that the DUAL mode is turned off, lowercase printing enabled, and graphics printing enabled. It also says that the computer is equipped with a Brand X (non-Radio Shack) doubler, and the high speed mod (if any) should be kept off. The high speed on and off command sequences have been altered as well, but they do not show on the configuration line.

Let's see what the new high speed on/off sequences say. The "on" command is the same as the previous one, except that it uses port # one instead of 254, and sends a two instead of a one. The "off" command likewise uses port one. It sends a three to that port.

Now suppose you want Super Utility to start using DUAL mode. Instead of re-typing the whole command line, you can go to the configuration module and enter a

single "Y". Super Utility will display 'DUAL=Y' at the beginning of the configuration table, and leave the rest of the line unaltered. In other words, when changing the configuration table, you need only type from the beginning of the line up through the part you're going to change. The rest of the line may be omitted.

To change any other line of the configuration table, press <ENTER> until the "=>" character is pointing at the line you want to alter. Then retype the line and press <ENTER>. If you entered the data correctly, the line will be updated and the "=>" will point at the next line. If your attempted input was illegal, the "=>" will remain where it was and the line will not be updated past the point of the error.

To go back to a higher line on the page, press <BREAK>, which takes you back to the top of the table.

If you have a high speed modification, now you may want to try the experiment I suggested a few paragraphs back--namely getting Super Utility to run at high speed, without adjusting its timing loops. To do so, lie a little! Re-enter line one of the configuration table. But for the 'OFF' sequence, type in the set of instructions that actually turns your high speed on. Also, tell Super Utility to turn off high speed by setting the configuration table to 'SPEED=N'.

Super Utility will think it has turned off your high speed clock, so it will not use its special timing loops. But in reality, it will have turned high speed on. Experiment (on a non-crucial disk, of course!) with various kinds of disk I/O. Also, test the timing of the key stroke auto-repeat in the Zap and Memory Modify modes. If all goes well, you may want to make this the normal mode of operation.

'GRAPHICS=Y' indicates that it's capable of printing TRS-80 graphics. If your printer doesn't have that capability, change the "Y" to an "N" and Super Utility will send dots to your printer, instead of graphics characters. If you have a printer like the EPSON MX-80, where the graphics codes are displaced from their normal positions, you may enter "M" here and Super Utility will automatically perform the needed adjustment. Incidentally, you should keep two things in mind: (a) the EPSON printer must be in its non-TRS-80 mode, and (b) it SHOULD NOT have the GRAFTRAX™ chips installed. GRAFTRAX™ takes away the TRS-80 block graphic set, so that an EPSON so equipped is no longer capable of printing them. In such a case your printer becomes a "non-EPSON" as far as Super Utility is concerned.

Use "Y" only if your MX-80 has its own TRS-80 configuration switch set to the TRS-80 position. Super Utility will then "adjust" graphics characters before sending them to your printer. If your MX-80 doesn't have its TRS-80 switch set to the TRS-80 position, the Epson responds like a "normal" printer, and Super Utility doesn't need to make any special adjustment.

'LOCASE=Y' means that your printer can print lower case letters. If you change this to "N," Super Utility will convert all lower case to upper before sending it to the printer.

'LINEFEEDS=N' indicates that your printer doesn't need a linefeed character sent after each carriage return. Most printers which can be used with a TRS-80 don't. If yours does, but Super Utility is configured to 'LINEFEEDS=N', then your printer will tend not to linefeed when it should, printing over the same part of the paper

repeatedly. On the other hand, if your printer does not require the extra linefeed, and you have Super Utility configured to 'LINEFEEDS=Y', your printer will tend to skip lines and double space when it shouldn't. So if you have any doubt as to the linefeed requirements of your printer, just do a few experimental screen prints.

The rest of the configuration table refers to any disk drives which may be on your system. You can have up to a maximum of four drives. If you have fewer, there will still be four entries in the table. Soon you will see how to adjust the configuration table to let SU know how many (and which) drives are actually in the system. Let's look at the sample entry for drive zero:

```
+:0 T1S' PTKS= 40 RTKS=40 DIR=17 STP=2 RDLY=4 WDLY=4 WP=N
    D0=S DD=S LS0=0 HS0=09 LSD=0 HSD=09 S/G=5 G/T=2 DD=S
```

It is composed of two lines. The first line, with one exception, contains information which you may alter directly by retyping the line. The second line contains "implied" information. It is derived from the data in the first line. The second line may be changed only by re-entering the first.

The first character on line one may be either a plus, minus, or an equal sign ("+", "-", or "="). "+" indicates that the device exists and is "logged into the system" (in other words, Super Utility knows it is there). "-" indicates that, at least as far as Super Utility is concerned, the drive is not in the system. "=" indicates that the drive is in the system, and is in the "double-step" mode. The double-step mode is for an 80 track drive which will be used to read a diskette formatted on a 35 or 40 track drive. Drives so configured will step twice, instead of once, for every track-step.

There are restrictions to be observed when using the double-step mode. In general, it's O.K. to read in this mode, but inadvisable to write. For fuller details, see the MISCELLANEOUS Chapter.

The second and third characters on the first disk line are a colon followed by a digit from zero to three. This simply identifies the drive number.

The 5th to 9th characters (the 4th is a blank), are used to describe the disk's operating system, SU+ calls these bytes a DOS specifier. The 5th is the kind of DOS system used (like T for TRSDOS). See the table below for the characters used for each DOS supported. The 6th character is sometimes referred to as the Model I or III specifier. Another way to look at it is as the specifier for the density of the first track. That's because the first track on the Model I TRS-80 is almost always single-density. The 1 in character position 6 then means SU will expect to find a single-density first track. If it was a 3, it would expect a double-density first track, because the Model III diskettes are almost always double-density on the first track. Special cases will sometimes arise and we will deal with them as they come up.

The 7th character is the density of all tracks **except** the first track. Use S for single-density or D for double-density.

In the 8th position, we usually find the single-side, double-side disk drive character. I say usually the 8th, because there is sometimes another special specifier (used mainly for NEWDOS 80). It is the relative sector specifier, "R" and is needed if you plan to read files and directories on diskettes made with the NEWDOS 80 system.

That "R" would push the side specification character into the ninth position. The side specification should be a single quote (apostrophe) for single-sided and a double quote for double-sided drives. It tells SU how many heads to look for on this particular drive.

If you plan to use the Model 4, use the Model III specifier when in the Model III mode -- which is whenever you're using Model III diskettes. When using the Model 4's TRSDOS 6.0, use the L3D specifier, because TRSDOS 6.0 is actually LDOS. The first track is double-density (the "3") and the rest of the disk is also double-density, therefore "D".

Your Super Utility Plus owner's manual has a table showing which letter declares what information. There are a few DOS's not mentioned in the table. Use "TS" for Newdos 2.1 (or Newdos +), and for single density Newdos-80. Also, if you have any NEWDOS 2.1 disks which have been modified with Doublezap to make them double density, use "N1D" for them.

You will have noticed that the DOS specifiers in this new version of Super Utility are considerably different from the previous versions. Here is the table of DOS specifiers for Version 3.0 of Super Utility Plus:

SYSTEM	Model I	Model III
TRSDOS		
Single Density	T, T1, TS, T1S	* INVALID *
Double Density	T1D	T3, TD, T3D
LDOS		
Single Density	L, L1, LS, L1S	L, LS, L3S
Double Density	L1D (SOLE)	L3, LD, L3D
DOSPLUS		
Single Density	D, D1, DS, D1S	D, DS, D3S
Double Density	D1D (system disks)	D3, DD, D3D
MULTIDOS		
Single Density	M, MS, M1, M1S	M, MS, M3S
Double Density	M1D (system disks)	M3, MD, M3D
NEWDOS/80 V.2		
Single Density	N, N1, NS, N1S	N, NS, N3S
Double Density	N1D (Tk. 0 reversed)	N3, ND, N3D
DBLDOS		
Double Density	B, B1, BD, B1D	* INVALID *

There are several things to note about this table. The very first thing is that several of these specifiers are identical for Model I and Model III disks. For example, to specify a single density MultiDOS disk, you can use MS regardless of whether the computer was a Model I or a Model III.

Conversely, there are some specifiers which are very machine specific. T1D, for example, pertains to Model I double-density TRSDOS only (TRSDOS 2.7DD). Similarly, L1D refers to a double-density Model I LDOS disk which has had the SOLE modification applied.

Secondly, some specifiers under "Model III" can refer to Model I disks as well. An example of this is a double-density Model I DOSPLUS data disk, which is identical to a Model III DOSPLUS disk. In this case you would use DD, even if you were reading it on a Model I. Other disks like this are LDOS double-density Model I data disks, and a double-density Model I MULTIDOS data disk.

Thirdly, there are other characters which you can append to the DOS specifiers to further define their structure. An apostrophe (e.g., L1D') indicates that the diskette is formatted on one side only. A double-quote (e.g., DD") indicates a double-sided disk. NOTE: Double sided support is available only for LDOS, DOSPLUS and the newest release of MultiDOS. This is not a matter of DOS chauvinism, but of simple lack of memory. These three DOS'es can be handled with the same identical set of routines, but others, like NEWDOS/80, require a totally different set of routines for themselves, and the Super Utility program cannot accommodate them without sacrificing badly-needed buffer space.

For NEWDOS/80 double density disks, and MultiDOS P-density disks (Model I only), you must append the letter "R" after the DOS specifier. This indicates that the track scheme on the disk follows the DISK RELATIVE SECTORS scheme, in which relative tracks may overlap physical tracks.

O.K. Let's focus on the first eight characters for the first drive in our sample configuration table:

+:0 T1S'

The "+:0" says that there is a drive zero (0) in the system and that it's not set to "double-step." The "TS'" says that the operating system on the disk is Model I TRSDOS, or a member of the Model I TRSDOS family (NEWDOS 2.1, or ULTRADOS, the revered ancestor of MultiDOS).

The item on drive zero's first line is PTKS= followed by a two digit decimal number which tells you how many physical tracks are present on the diskette in drive zero. In our example, the diskette is formatted to 40 tracks. Then comes "RTKS= " followed by a two digit number. It shows the relative track count for the diskette. The number of relative tracks will normally be equal to the number of physical tracks. The exception occurs with NEWDOS-80, DBLDOS, and MULTIDOS "P-density" disks which use Disk Relative Sectors (DRS). I will present more information on DRS in the MISCELLANEOUS Chapter. Since the disk in our example is Model I TRSDOS, the number or relative tracks equals the number of physical tracks. Hence "RTKS= 40."

"DIR= nn" shows the number of the disk's directory track. On our sample disk, the directory is on track 17. When working with NEWDOS-80 or DBLDOS disks, it will show the relative, rather than the real, track number.

"STP= ", "RDLY= " and "WDLY= " pertain to hardware timing constraints of the disk drive. The stepping time refers to the time it takes the read/write head to move from one track to the next. Some drives must be allowed 40 ms (milliseconds) per step. Others can get by with only 6 ms. Most drives can step within at most 20 ms. Super Utility gives you a choice of four stepping speeds per drive: 0=6 ms, 1=12 ms, 2=20 ms, and 3=40 ms. Our sample drive zero is set for "2," or 20 ms.

Please make note of two facts:

1) If a drive is configured to a single density DOS, it will not step faster than once every 12 ms, even if it's configured to the 6 ms speed. This is due to TRS-80 hardware design, and is not a bug in Super Utility.

2) It will never do any harm to configure a drive for too slow a step-rate, except that disk operations will lag a bit. However, configuring too high a speed can result in extremely unreliable I/O. If you're using a fast step-rate and getting frequent "inexplicable" I/O errors, try slowing the step-rate down to 40 ms.

"RDLY" refers to the period of time it takes from the moment a drive turns on to the moment it stabilizes at full speed and Super Utility attempts to read from the disk. Conversely, "WDLY" refers to the period of time from the moment the drive turns on to the moment Super Utility attempts to WRITE to the disk. One second is an adequate allowance for most drives. Some come up to speed even more quickly. The new version of Super Utility allows you to adjust these settings by QUARTER-SECONDS. As with the stepping-rate, you're safe with a speed that's too slow, but not with one that's too fast. So if you have problems, try slowing down.

The delay is entered and displayed in quarter seconds. Some older versions of Super Utility allowed 1/2 or 1 second delay intervals only.

The sample drive zero is set to 4-quarter seconds (or one full second) delay.

The final item on line one is 'WP=N.' "WP" stands for software write-protect. If you set it to "Y" for "Yes," Super Utility will act as if the disk in that drive had its write protect notch covered, and refuse to write to it.

As stated earlier, the second line of drive data contains information implied by the first line. In our sample, the first item is 'D0=S.' This means that the density of TRACK 0 is single. 'S' stands for "Single" and it specifies that track zero is in single density. This is meaningful because track zero doesn't always have to be the same density as the rest of the disk. This is especially true for double density Model I disks. The Model I requires track zero, sector zero of any bootable disk to be in (DOSPLUS, for example,) have a single density track zero. This makes double density DOSPLUS disks bootable on a Model I.

Other Model I double density DOS's (like LDOS) require track zero to be the same density as the other tracks on the disk. This has the advantage of being consistent, and leaving a little more room on double density disks; but if you want to run LDOS with a double density diskette in drive zero, you have to boot in a single density LDOS disk and then swap.

Recently, however, MISOSYS in Alexandria, VA came up with a modification system called SOLE which places a single density track 0 on a double-density LDOS Model I system disk. Previous versions of Super Utility did not recognize this form. This version, however, knows enough to look for a SOLE-type disk if the DOS specifier L1D' is given.

NEWDOS-80 allows you more or less full flexibility. You may have a track zero of either density, regardless of the density of the other tracks. But Super Utility doesn't support all such possible NEWDOS-80 configurations. For discussions of which are supported, see the MISCELLANEOUS Chapter of this book. At any rate, the diskette in our sample drive zero contains Model I TRSDOS, so track zero, like all the other tracks, is single density.

The next bit of information is 'DD=S.' 'DD' stands for "Density of Disk." This indicates what the density of the REST of the disk (track 0 excepted) is.

Following this is 'LS0=0' and 'HS0=09'. LS0 stands for "Lowest sector on Track 0." HS0, conversely, means "Highest sector on track 0. In every DOS except Model III TRSDOS, the first sector on each track is zero. Model III TRSDOS doesn't use a sector zero, so the first sector of each of its tracks is one. And because of this, the highest sector has to be 18 instead of 17 like everyone else's double density track. In our example, since we are looking at a single-density disk, the highest sector is naturally 9, since sector numbers go from 0 to 9.

Next we have 'LSD=0' and 'HSD=09.' These two items tell you the lowest and highest sector number on the rest of the disk. Unless a disk is one of the mixed-density types, these two values will be the same as LS0 and HS0.

The next item is 'S/G=5.' This tells you the number of sectors per granule there are on the disk. Remember that "granule" is a totally arbitrary term, and its size varies depending on whether the disk is formatted in single or double density, and whether the DOS is TRSDOS 1.3 or something else. TRSDOS for the Model III uses 3 sectors per granule, half the size of everyone else's granule.

Related to this is the next item: 'G/T=2'. G/T stands for "Granules per track." Since the number of sectors on a track are pretty much fixed (10 for single density and 18 for double) the number of granules per track will vary depending on granule size. It will also vary depending on whether the standard track scheme or the Disk Relative Sectors scheme is being used.

Finally, we have 'DD=S'. DD stands for Data Address Mark of Directory. See the Technical Introduction for more information on DAM's. The DAM's being referred to here are the non-standard DAM's to be found on the disk. Usually, non-standard DAM's are used only for directory sectors. The only known exception is Model III TRSDOS which reverses things. It uses standard DAM's in the directory and Read Protected DAM's everywhere else.

'DD=S' indicates that the directory DAM's are of the Read Protected variety and every other track uses standard DAMS. This is the Standard Scheme, hence the "S." This is the scheme used by every DOS except TRSDOS 1.3. TRSDOS 1.3 will show DD=I, to indicate "Inverted DAM scheme."

Now let's take a quick look at the other three drives in our sample configuration table. The two lines for drive one read as follows:

```
+:1 T3D' PTKS= 80 RTKS= 80 DIR= 40 STP=0 RDLY=2 WDLY=2 WP=N
    D0=D DD=D LS0=1 HS0=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I
```

The information on line one indicates that we have, in drive one, a Model III TRSDOS family disk with 80 physical tracks and 80 relative tracks. The directory is on track 40. The drive read/write head can step from track to track within 6 ms, and the drive should come up to speed within half a second before I/O is attempted.

Line two describes a double density disk with logical grans. Logical Grans are those that start with a 1 for the first gran and two for the second gran and so on. I guess grans that start with the first sector as sector zero are "illogical grans." Track zero is in double density; the starting sector of each track is sector one, and the directory Data Address Marks use the Inverted scheme.

Now for drive two:

```
=:2 L1S' PTKS= 35 RTKS= 35 DIR= 17 STP=0 RDLY=2 WDLY=2  WP=Y
    D0=S DD=S LS0=0 HS0=09 LSD=0 HSD=09 S/G=5 G/T=2 DD=S
```

The "=" at the beginning of line one indicates that drive two is an 80 track drive (narrow track width) containing a diskette which was formatted on a 35 or 40 track drive (full track width). The rest of line one shows that the diskette was formatted with LDOS, has 35 physical tracks and 35 relative tracks. Its directory is on track 17. The drive can complete a track-step within 6 ms and comes up to speed within half a second. It is software write protected.

Line two indicates that the diskette is all single density and has 5-sector grans. Track zero is in single density, the starting sector of each track is zero, and the directory uses the Standard DAM scheme.

And finally, drive three:

```
-:3 N1D' PTKS= 35 RTKS= 63 DIR= 17 STP=3 RDLY=4 WDLY=4  WP=N
    D0=S DD=D LS0=0 HS0=09 LSD=0 HSD=09 S/G=5 G/T=2 DD=S
```

The minus ("-") at the beginning of line one indicates that there is no drive three in the system, so the rest of the drive data is of little consequence. However, if the drive described were logged onto the system, line one indicates that it would be configured to read a NEWDOS-80 diskette with 35 real, or physical, tracks and 63 relative tracks. The directory is shown to be on relative track 17. The step speed of the drive is 40 ms and the drive requires a full second delay for both reads and writes.

Line two describes a double density diskette with illogical grans. Track zero is single density. The first sector of each track is sector zero, and the directory has Read protected Data Address Marks.

As I said earlier, some Super Utility users seem to have trouble setting up the configuration table for their own drives. The difficulty nearly always is the result of misunderstanding how the data is entered. Here are three simple rules to follow. They all apply when you're changing the the lines in the configuration table which pertain to the drives:

Rule #1: The first thing you must enter for a drive is either a DOS specifier (see the table above) or "+", "-", or "=".

People often forget this when they want to change the number of physical tracks on a drive. Let's consider an example. Suppose our configuration table shows drive one as follows:

```
+:1 T3D' PTKS= 40 RTKS= 40 DIR= 17 STP=0 RDLY=2 WDLY=2  WP=N
    D0=D DD=D LS0=1 HS0=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I
```

To change the number of tracks from 40 to 35, the first thing to do is to get the modify arrow down to where it's pointing at the "+". So hit <ENTER> until you see the following:

```
=> +:1 T3D' PTKS= 40 RTKS= 40 DIR= 17 STP=0 RDLY=2 WDLY=2 WP=N
    D0=D DD=D LS0=1 HS0=18 LSD=1 HSD=18 S/G=3 G/T=6 DD=I
```

Now here's where people have gone wrong. They type "35" and press <ENTER>. Super Utility rejects this input and waits for them to try again. The correct thing to enter is "T3D' 35". "+T3D' 35" would also work, but the "+" is unnecessary. If you want, you may also put a comma between the "T3D'" and the "35"--e.g. "T3D',35" or "+T3D',35". But again, this is unnecessary.

Rule #2: The second line of information for each drive is implied information. You can change it only by changing line one.

Rule #3: Line one has two track-counts. The first is for physical tracks, the second for relative tracks. Line one's second track-count ('RTKS= ') is implied, like the data on line two. It can be changed only by altering track-count #1 or the DOS specifier.

Let's go back to the previous example. We want to change

```
+:1 T3D' PTKS= 40 RTKS= 40 DIR= 17 STP=0 RDLY=2 WDLY=2 WP=N
```

to

```
+:1 T3D' PTKS= 35 RTKS= 35 DIR= 17 STP=0 RDLY=2 WDLY=2 WP=N
```

Here's a WRONG way to enter the information:

```
TD 35,35
```

People sometimes make the mistake of entering "35" twice, since they want both 40's in the target line to be replaced by 35's. But the first 35 entered will change both 40's to 35's. If you enter "35" twice, Super Utility will think you want it to use the second 35 for the drive's directory track. The correct way to enter the change is

```
T3D 35 or TD,35.
```

Here's an input line that will change every item on line one in this example.:

=N1DR 35,20,3,4,4,Y

<page>The "=" sets the drive to double-step. The N1D specifies a double density NEWDOS-80 operating system with a single density track zero. The "35" specifies a 35 track diskette. Since NEWDOS-80 uses Disk Relative Sectors, the combination of N1D with "R" and "35" sets the relative track-count to 63. The 20 tells Super Utility that the directory is on track 20. Since this is a DOS that uses data relative sectors, the "20" indicates relative track 20, not physical track 20. The "3" sets the drive to a 40 ms stepping speed. The two "4's" set the drive for 4 half-seconds (one full second) delay on both reads and writes. And the "Y" turns on software write-protect for drive one.

The new information in the configuration table will look like this:

+:1 N1DR PTKS=35 RTKS= 63 Dir= 20 STP=3 RDLY=4 WDLY=4 WP=Y
D0=S DD=D LS0=0 HS0=9 LSD=0 HSD=9 S/G=5 G/T=2 DD=S

Notice that line two has also changed. It now indicates a diskette with illogical grans, a starting sector of zero, and a single density track 0.

Using the configuration module to reconfigure Super Utility is called "soft configuring." Every time you boot Super Utility the original configuration will be restored and you will have to reconfigure Super Utility for your system. You may prefer to have Super Utility boot up preconfigured to your specifications. In the old version, you could accomplish this by zapping certain sectors of your Super Utility disk. This is no longer necessary in Version 3.0.

After you have finished configuring your drives, you will see a new prompt at the bottom of your screen, and it will say, "Save Configuration ?" If you reply "Y," you will be asked to place your Super Utility Disk in drive 0. When you have done so, press <ENTER> and the new configuration information will be written to the disk.

Note that this method of configuration does not allow you to change things that you could formerly change in the older versions of the program, such as the disk name, the format pattern, etc. However, it isn't really necessary to change these anyway, and this new save feature is much easier to use -- especially if you're new to Super Utility Plus!

After you've saved your configuration, it will always be established whenever you boot up the system. At present, there's no way to override the saved configuration and restore the defaults, so make sure the configuration you save is one that you'll be using most often. Of course, you can always change the configuration table to the way you want after Super Utility is loaded, but a little advance planning can save you time.

One last word about the high-speed clock control sequences. Super Utility uses port 254 in conjunction with its clock speed control. Therefore, any port 254 initialization which you do may be countered by Super Utility when it initializes the clock. You should use the "ON" and "OFF" sequences in the configuration to achieve the net effect that you want.

I'll consider two examples:1) You have a high speed mod and a reverse video mod, each of which uses port 254.

2) You have no high speed mod, but you do have a reverse video mod which uses port 254.

In example #1, you have a high speed mod. The least significant bit of any value output to port 254 controls the speed. In other words, OUT 254,1 turns on high speed and OUT 254,0 restores normal speed. This is a fairly standard way of doing things. Now, suppose you also have a reverse video mod which is controlled by the second least significant bit of any value output via port 254. In other words, OUT 254,2 would turn on reverse video, and OUT 254,0 switches back to the normal display mode. As a final supposition, let's assume that you would like Super Utility to come up in both reverse video and high speed. To turn both on simultaneously, a value with both the least significant and second least significant bits set must be sent to port 254. OUT 254,3 would do the trick. So all you have to do is configure Super Utility to 'Speed =Y' and the 'ON' sequence to 3E03D3FE.

This disassembles as

```
LD      A,03
OUT     (254),A
```

The net effect is to send a "3" to port 254.

It would probably also be a good idea to configure your 'OFF' sequence to 3E01D3FE. This disassembles as

```
LD      A,01
OUT     (254),A
```

The net effect of this code is to send a one to port 254. The advantage, in this case, of turning off the high speed clock with a one rather than a zero, is that the one won't turn off the reverse video as well, whereas a zero would. Example #2 assumes that you have no highspeed mod, but you do have a reverse video mod which is turned on and off with the instructions OUT (254),1 and OUT (254),0. You want Super Utility to turn your reverse video on, and keep it turned on.

The easiest thing to do is to use the configuration to tell Super Utility that you don't have a high speed mod. Also tell Super Utility that 'OFF' will be 3E01D3FE' or in effect, OUT (254),1. This will keep your reverse video turned on.

Saving the configuration table is one of the TWO times that you need to have your Super Utility Plus disk in the drive. The other time is when the program is booted. There are no other times when you need to have Super Utility in a disk drive, since the program is totally self-contained. You should get into the habit of removing your Super Utility disk from the drive as soon as the program is booted up, or as soon as your configuration has been saved. PowerSOFT receive many calls from people who inadvertently format over their Super Utility disk because they left it in drive 0 and forgot it was there. This kind of mistake can cost you money and lost time. So make sure that after your Super Utility program is booted up, you take the disk out

of drive 0 and tuck it away in a safe place. This will also make your disk last longer -- disks do wear out, after all.

The new Super Utility comes with three separate boots. Remember, your copy of Super Utility will load and run on both the Model I and Model III. Because you may need to use a mixed-density disk on the Model I to boot up in double density, there is one special boot which will let you do just this. The other two consist of a Model I boot and a Model III boot. That way, no matter which machine you're working on at any given time, you may repair disks belonging to either model and you also have the added flexibility of using a boot that can correctly load a file from a double density track on a Model I.

PowerSOFT felt that if you work on both models, you may have a different set of disks containing different operating systems with each model. Therefore, you might need a separate set of repair-boots for each.

Whenever you use Super Utility's Repair BOOT Sector option, Super Utility will now write the appropriate boot to the target disk, depending on the configuration setup for that drive. The Model I boot gets put on track zero, sector zero, where the Model I would expect to find it. The Model III boot is placed on track zero, sector one, where the Model III expects to find its boot.

The other DOSes sometimes have different boot-loaders and it would be a good idea to copy the boot sector from a good disk (one that boots) of the same DOS and the same configuration. Use the Copy Sectors utility. Be sure to get it in the right spot (the first sector of the first track). The only problem might be on NEWDOS80 where there are two BOOT sectors (those are on double-density disks with single-density first tracks). See the discussion of NEWDOS80 in the Miscellaneous chapter.

Chapter III
WHAT TO DO WITH A MYSTERY DISK

When Super Utility won't behave, the reason is almost always the same. The program is not configured correctly for the disk(s) being worked on. If you don't understand Super Utility's configuration module, the previous chapter of this section of this book zooms in on the topic.

If you understand the configuration process, you may still feel caught up in a kind of Catch-22. Super Utility won't read your disk unless you first configure correctly. You can't configure until you know some facts about the disk--such as the computer it's supposed to run on (Model I or III), what DOS it contains, whether it's single or double density, how many tracks it has, where the directory is located, and whether it was created on a 40 or 80 track drive. You can't get this information until Super Utility lets you look at the disk. So where do you start?

If you have no idea of what a disk contains, density-wise, DOS-wise, and every other -wise, you have two options. Version 3.0 of Super Utility has an automatic DOS recognition feature which you can invoke. From the ZAP menu, select Display Disk Sectors and enter the drive number preceded by an exclamation point (for example, !0). Super Utility will then try to identify the disk both density-wise and DOS-wise. This feature is available in Display Disk Sectors and any other procedure which accesses a disk's directory. It will take a minute or two, and it's not always 100% reliable, but if it succeeds, then the configuration table will be automatically updated, and your worries are over.

If the automatic DOS recognition system fails, then you have to fall back on more manual methods. Use Zap's Read ID Address Marks command (option 11 from the Zap menu). This can give you a good deal of information about the disk. Here's the procedure.

Go to the configuration module and reconfigure the drive you're going to use as follows: configure the drive to be in the system. Do not set it to double-step, and make sure it's configured to the highest possible number of tracks the drive is physically capable of supporting. For instance, suppose you're going to use drive one and it's an 80 track drive. You look at the configuration table and press <ENTER> until get the "=>" gets down to the line for drive one. Then enter

+TS,80

In this case, the "TS" is a dummy parameter, it doesn't mean we think the DOS is TRSDOS. You could just as well enter any other DOS specifier.

Make sure the configuration table has been updated correctly. Then press <SHIFT><BREAK> to return to the main menu. Press <ENTER> to go to to the Zap utility. Then enter "11" to select the "Read ID Address Marks" function. Finally, enter "1" to choose drive one. Super Utility will now try to read the DAM's for drive one, track zero.

There are two major possibilities:

Possibility #1. Super Utility completely or partially succeeds at reading track zero's DAM's. If it succeeds completely, the screen will be filled with a scrolling display under all the headings except the last three (if there is information under the last three headings as well, hold down the <X> key until that data disappears--see note 1).

If Super Utility has partial success at reading track zero's ID address marks, you will see information scrolling up the screen, interspersed with error messages.

Possibility #2. Super Utility is completely unable to read track zero's ID address marks. Error messages appear on every line of the display.

If Super Utility can read track zero, make note of its density. The density is indicated by either an "S" for "Single," or a "D" for "Double," in the the column under the "u" of the word "Source."

If Super Utility can't read track zero, it could mean any of several things. Assuming your system is in good repair, you might be trying to read an unformatted disk. If your system doesn't have a double density board, you might be trying to read a disk whose track zero was formatted in double density. Or you might be trying to read a badly farkled (or messed up) disk.

Hold down the up-arrow key. Super Utility will attempt to read subsequent tracks. If it succeeds with all tracks other than zero, and your system doesn't have double density, you may be trying to read a NEWDOS-80 disk which was created with a double density track zero, but with all the other tracks formatted in single density. However, it's unlikely anyone would format a disk in such a manner. What is more probable is that something farkled track zero without affecting the rest of the disk.

If Super Utility succeeds at reading track zero, there are two special "wrong drive-type" patterns to be alert for. In one, the track number shown under the heading "Track" increases by twos, while the track count under the heading "Source" increases by ones. This probably means you're using a 35 or 40 track drive to read a disk which was formatted in an 80 track drive. Try switching the diskette to an 80 track drive and starting again.

This reading will not be the best, there are likely to be many errors. The reason being that it's theoretically impossible to read this disk at all! Well, almost . . . the idea is that this is a 35- or 40-track drive, which should be trying to read TWO of the 80's tracks at the same time. Theoretically, it should be confused and not be able to read either. The real situation is that it can read "something" relatively often. I think it probable that there is never going to be perfect alignment, so one of the tracks will predominate, and at least part of it will be read. At least enough is read for you to see the track numbers change as mentioned. "Track" increases by two, track count or "Source" increases by one.

A note of caution here.

If you have the configuration table set up for double-step and are reading a 35/40-track disk on a 35/40-track drive (as you normally would) then it will mimic the

80-track diskette on 35/40-track drive characteristic we just talked about. Be sure you are reading what you want, where you want, and the way you want to. All it takes is a quick glance at the SU displays and configuration tables.

In the other "wrong drive-type" pattern, the track count in the "Source" column goes up by twos, while the track count in the "Track" column goes up by ones. Also, every other time the head steps (you should hear a tick from your disk drive each time it steps), you will get a series of "ID Read Error" messages. This indicates that you're using an 80 track drive to read a diskette which was formatted in a 35 or 40 track drive. Either switch the disk to a suitable drive, or go back to Super Utility's configuration module and configure the drive you're using to double-step (use an equal sign ("=")).

Once you have Read ID Address Marks happily reading your disk, hold down the <X> key until Super Utility starts filling in the last three columns of information (see note 1). Keep your eye on the "Data" column. This tells you the DAM type for each sector on the track. Now press the up-arrow repeatedly to step through the tracks (See note 2). With any DOS other than Model III TRSDOS, the "Data" entry for all non-directory sectors should say 'S>td,' for "standard." When the entries in the data column change from S>td to 'R>ptc' (for "read protected") or 'U>df' (for "user defined"), you have located the directory track. With TRSDOS III, things are switched around. The directory has "standard" DAM's and the rest of the disk has "read protected" DAM's.

The directory will usually be on track 17 decimal for 35 or 40 track diskettes. For 80 track disks, the directory is normally on or near track 40. On a Model I, if the DAM was "U>df," you're looking at an LDOS disk. "R>ptc" indicates any operating system other than LDOS.

While looking at the directory track, use the space bar to freeze the action. Look for the lowest numbered sector. If you don't see a zero, start the scrolling again by pressing <ENTER>, and freeze it with <SPACE> again. Repeat this procedure several times, each time checking for the lowest numbered sector visible on the display (you might not catch the track's lowest sector on the screen the first few tries). If the lowest sector you can find is one, you're probably looking at a Model III TRSDOS diskette. All other standard DOS's should start each track with sector zero.

Another thing to watch for is whether the directory, as revealed by the change in DAM's, starts at the beginning of its track, or somewhere in the middle. If the directory doesn't occupy the whole track, then you'll see some sectors with S>td DAM's, and other sectors in the same track with R>ptc DAM's. In that case, the directory will probably "wrap around" and occupy some of the next track as well. When this occurs, you may be fairly certain you're dealing with NEWDOS-80 or DBLDOS.

Take note of the number of the directory track(s), and its density. Then, press <X> to go back to mode one (see note one). Hold down the up-arrow and let auto repeat take over. The read write head should step in track by track.

When you reach a track whose ID marks can't be read, Super Utility will "stick." If you get an error message that says "Not Formatted," or "ID Read Error," and Super Utility doesn't find any sectors for that track, you may have past the last formatted

track. If the source track is 35, 40, or 80, this is especially likely. To be certain, you may try forcing the head to search for more tracks further toward the center of the disk. To do this, press the up-arrow a few times.

When you have found and noted down the last track on the disk, there are a couple of other places on the disk to inspect--the Boot and GAT sectors. Start by pressing <SHIFT><BREAK> to get back to the main menu. Then go to the configuration program. Configure the drive you're using to the correct density, track number, and directory track number your inspection has revealed.

Here's how to configure for the DOS. First of all, you may have already discovered which DOS the disk contains. If the directory had Udf DAM's, then the DOS is LDOS. Use "LS" if the tracks were in single density, and "D" if they were double. If there were no sector zeros, then the DOS is either Model I double-density or Model III TRSDOS. Use "TD."

If the directory started in the middle of a track and wrapped around to the next track, the DOS is NEWDOS-80 or DBLDOS. If track zero is single density and the rest of the disk is double, you won't be able to determine which of the two it is until you examine the GAT. Use "B" in the mean time. If track zero is double density, NEWDOS-80 is indicated. Use "NDR." If you don't know the DOS, use "TS" for any all single density disk, "LD" for an all-double density disk, or "D1S" for a double density disk with a single density track zero.

Go to Zap's Display Sector program. When prompted with "Drive, Track, Sector", input the target disk's drive number, and default on the other two values. Super Utility will display track zero, sector zero or one, depending upon whether you're looking at double density TRSDOS or some other DOS.

If the sector you see is mostly zeros, you're looking at a data (or non-system, or non-booting) disk. If the sector is full of data, you may be looking at a booting disk, or it might still be a data disk. You should also see an all too familiar message or two near the end of the sector--"DISK ERROR" or "NO SYSTEM."

While you're looking at the boot sector, look at the third byte. For most DOS's, that should contain the number of the directory track. If it doesn't agree with the number you came up with when you used Read ID Marks, it could mean one of two things.

1) You may be confused because Read ID Marks gave you the track number in decimal, but the byte in the boot sector in in hex. Convert the number base of one of the values and see if they agree after all.

2) You may be looking at a DBLDOS, NEWDOS-80, TRSDOS 2.7DD or a TRSDOS 1.3 disk. These are the only DOS's whose third byte on the boot sector doesn't have to point at the directory. DBLDOS has no directory pointer at all. TRSDOS III uses the second boot-byte, rather than the third, as the directory pointer.

Double density NEWDOS-80 disks with single density track zeros maintain a second boot sector at relative track zero, sector zero. This happens to be the same as physical track one, sector zero. It is this second boot whose third byte is required to point to the directory track.

3) You may be looking at a NEWDOS-80 disk. Boot byte three is pointing at the directory, all right. But since it uses "illogical" tracks, it won't seem to be pointing where we would expect.

You might think that if the answer were (3), we'd have already known that we were looking at NEWDOS-80. After all, when we ran Read ID Marks, we'd have discovered the directory to be spread across two tracks. However, this particular NEWDOS-80 disk might just happen to have its directory contained within a single physical track despite its "illogical grans". If such were the case, we wouldn't yet know what we were dealing with.

The next thing to do is use Zap to look at the GAT sector--the first sector in the directory. The GAT sector contains the disk name, date, the hash of the master password (which will probably be unrecognizable), and the AUTO command (if any). There is a good chance that the disk name will identify the DOS for us. This is especially true if the disk is a system, rather than data diskette.

If you go through the whole rigmarole described above and still don't discover what kind of disk you're looking at, try zapping through the disk. System modules usually have ASCII copyright messages embedded in them.

When you know the disk's DOS, density, track-count, etc, go to the configuration module and update the configuration table to incorporate all the hot new information. Finally, run Zap's verify program and Disk Repair's Check Directory program. While you're in the Disk Repair menu, use option eight to display the diskette's directory. If it has lots of "/SYS" files ("/DOS" files on Multidos), you're probably looking at a system disk. If only BOOT/SYS and DIR/SYS are listed, you've got a data disk. By now you should have a comprehensive picture of the mystery disk.

Chapter IV
Disk Errors and How to Fix Them

Before getting too deeply into errors and error recovery, there is one point I would like to emphasize. Super Utility's features far exceed the capacities of competing programs, but it can't do miracles. So don't get overconfident. The most important step you can take to avoid the loss of crucial data is preventive. Please make backups.

Another worthwhile prophylactic measure is to periodically perform a Verify Sectors and Format without Erase on all important disks. In other words, pretend a problem exists and follow the problem recipe outlined below. This may seem a lengthy process, but it is a time-investment which, in the long run, may save you many more hours than you put in.

One simple way to classify disk errors is to break them up into two categories--minor and drastic. Minor errors are those in which only a small number of bytes have been clobbered. Drastic errors indicate massive alteration or obliteration of disk data. Super Utility can correct most minor disk errors. But when drastic errors occur, you'd better have those backups handy.

The techniques I'll be describing here deal with the repair of errors which fall into the "minor" category. When a minor error occurs, it usually restricts itself to one of the two subfields of one sector. I will refer to errors in the header subfield as header errors, and errors in the data subfield as data errors. Another way to lump errors into categories is to separate them into errors which occur in the directory and errors which occur elsewhere on the disk. Directory errors may be much more critical than others, or they may be much simpler to deal with, depending on whether or not they're in one of the first two directory sectors.

The first two directory sectors contain the Granule Allocation Table (GAT) and Hash Index Table (HIT), respectively. The unique thing about these tables is that the crucial information they contain may be derived from the other directory sectors. Therefore, if one or both of these sectors is wiped out, Super Utility can reconstruct them for you, almost completely automatically--as long as the rest of the directory is intact. Now that we've had a quick overview of the taxonomy of disk errors, let's take a closer look at header errors. One insidious thing about them is that, though they are minor in the sense of only involving a byte or two, they are still drastic in the sense of being nearly impossible to fix. A sector which contains a bad header may be made usable, by Format without Erase. But the data in the bad sector might be beyond recovery. Remember, the header contains four bytes of information which identify and define a sector--namely the track number, head number, sector number and sector length. These bytes, you will recall, are preceded by a one byte ID mark, namely FE hex, and followed by a two byte CRC. Any alteration to the data in this area makes it impossible for DOS or other software to access the sector in question; usually the afflicted sector can't even be located!

When a sector becomes totally lost to DOS, trying to access it results in error messages like: "Sector NOT FOUND" (this is Super Utility's own error message), "SEEK

ERROR DURING READ" (TRSDOS error 02), or "SEEK ERROR DURING WRITE" (TRSDOS error 10). The message you get may vary, depending on the DOS (or other software) attempting the disk I/O.

NOTE: All the DOS error numbers in this section are in decimal.

If a header CRC error occurs, DOS may be able to find the sector, but the data it contains will still be inaccessible. This is because the FDC generates an interrupt when it detects the error condition, and doesn't read any further. In such cases, you may get a message such as "Sector NOT FOUND, ID CRC Error" (Super Utility's message), "PARITY ERROR DURING HEADER READ" (TRSDOS error 01), or "PARITY ERROR DURING HEADER WRITE" (TRSDOS error 09).

One thing about these errors you should be aware of is that, despite the error messages, they are not parity errors, in the normal sense, but CRC errors. "Parity" usually refers to a one bit flag which indicates an odd or even number of one-bits in a given field, usually a byte. The CRC is a 16 bit indicator which reflects a great deal more than a one-bit count.

If you get a header (or ID) error, it's time to go for the backup. If you don't have a backup, and are really desperate to recover the data, you may be able to do so, at least partially. The method involves doing a track read, printing out the result, reformatting (without erase) the track with the problem sector, and re-entering the lost data by hand, using the Modify Mode of Zap's Display Sector module. This is a somewhat laborious procedure, and it often doesn't succeed. However, it's a worthwhile last ditch attempt to recover from a variety of wipeouts. I describe the procedure in detail under the heading TRACK RESCUE. Errors which affect a sector's data subfield may be much easier to deal with. If the Data Address Mark got clobbered, you're out of luck--it's like an error in the header subfield; the sector may be totally lost. DOS will tell you something like "DATA RECORD NOT FOUND DURING READ" (TRSDOS error 05) or "DATA RECORD NOT FOUND DURING WRITE" (TRSDOS Error 13). Super Utility will probably give you the good old "SECTOR NOT FOUND" error message. The exception to this is an error in which the DAM gets changed into one of the other legal DAM bytes (there are four possible DAM's for the Model I, and 2 for the Model III). In such a case, Super Utility will be able to restore the disk.

By the way, don't confuse the "DATA RECORD NOT FOUND DURING READ" type message with one like "LOST DATA DURING READ" or "LOST DATA DURING WRITE" (TRSDOS errors 03 and 11, respectively). These are two entirely different kettles of error. "LOST DATA" doesn't necessarily indicate anything wrong with the diskette at all. It means that the CPU couldn't catch the data as fast as the FDC was grabbing it from the disk. The problem is often that the drive is spinning too fast. Have it timed!

Except for clobbered DAM's, a data subfield error may be fixed relatively easily. Super Utility's message, when it encounters such a sector, is "DATA CRC Error." DOS may present you with a message like "PARITY ERROR DURING READ" (TRSDOS error 04) or "PARITY ERROR DURING WRITE" (TRSDOS error 12).

Here's a recipe for dealing with problem disks:

PROBLEM RECIPE

1) Follow the instructions under the heading "VERIFYING A DISK". This will show you how to test the disk with Super Utility's "Verify Sectors" program. Then come back to this recipe.

A) If the final result of "Verify Sectors" is "0 bad sectors," even though you may have needed several retries on some sectors, follow the instructions under the heading "FORMAT WITHOUT ERASE".

B) If the final result of "Verify Sectors" is more than zero bad sectors, then follow the instructions under the heading "STUBBORN ERRORS."

2) Follow the instructions under the heading Directory Repairs.

END OF PROBLEM RECIPE

One other error message which DOS may occasionally give you is "ATTEMPTED TO READ SYSTEM DATA RECORD" (TRSDOS error messages 06 and 07). According to the TRS manual, this is usually the fault of a user program. What is actually happening is that DOS is trying to read a sector which has incorrect DAM's. This could happen under a number of circumstances. One possible cause could be your using Super Utility's Read Protect Directory program and give the wrong address or length for the directory. The careless use of Zap's Alter Data Address Marks could have the same effect.

To overcome the problem, use Zap's Read ID Address Marks to locate all sectors with incorrect DAM's. Remember, except with Model III TRSDOS, all non-directory sectors should have "standard" DAM's. Model III TRSDOS should have "standard" DAM's in the directory and "read protected" DAM's everywhere else. Use the Alter Data Address Marks option to correct aberrant sectors.

VERIFYING A DISK

If your disk seems to have errors, boot up Super Utility, go into Zap, and use the Verify Sectors program (Zap selection #2) to go over the problem disk. If Super Utility isn't configured correctly for the target disk, be sure to use the proper DOS and track-count overrides with the Verify command.

Each time Verify Sectors reports an error, make a note of the track and sector number displayed, and the type of error. If you have a printer, instead of taking notes, you can set 'DUAL=Y' from the configuration module, before starting this procedure. Retry reading each problem sector several times (if you're desperate, use the <C>ontinuous or <N>onstop selections and retry every problem sector for a couple of minutes).

At this point I would like to establish a couple of definitions:

1) From now on, I will refer to problem sectors which Super Utility succeeds at reading after some retries as compliant problem sectors.

2) I will refer to sectors which Super Utility can not read, even after a large number of retries, as stubborn problem sectors.

As you note down each problem sector, also record whether it is compliant or stubborn. If a problem is stubborn enough to defy four or five read attempts, you can usually write it off as completely stubborn. Compliant problem sectors may be repaired automatically by the Format without Erase routine. But Format without Erase will be one of the last steps in fixing your disk, because a premature Format without Erase can make it impossible to fix certain stubborn problems which other techniques might save.

When Verify Sectors is finished, it will display the number of stubborn sectors (it will call them bad sectors). Take note of this number, and return to the Problem recipe.

FORMAT WITHOUT ERASE

WARNING: It is extremely important to correctly configure Super Utility for your disk before using Format without Erase. Either use Super Utility's configuration module or use the proper DOS specifier with each Format without Erase command. If you use DOS specifiers with Format without Erase, you should use a track-count override as well. If you don't know how to enter DOS specifiers and track-count overrides, the section on disk-extending in the MISCELLANEOUS Chapter contains an example which should be helpful.

Format without Erase is item number three on the Format menu. When you run it, it will pause at the first problem sector it finds (see note 1). If your notes indicate that this is a stubborn sector, just use <S> to skip it. If, on the other hand, it's a compliant one, enter <C> and let Super Utility retry the sector until it is read and reformatted--or, if your notes indicate that the current problem sector is the first of several compliant problem sectors, with no intervening stubborn ones, enter <N> and let Super Utility zip through them all automatically.

After Format without Erase has gone over the entire disk, it will go back to track zero an verify all it has done. Unless your disk is physically damaged or worn, this verify should not encounter any problems. Finally, Format without Erase will update the directory.

The directory update does two things. For one thing, it can increase the number of available tracks in the allocation table. This is unlikely, but it can happen as a result of your "extending" a disk the hard way. To extend a disk is to increase the number of tracks it has, without losing any of the data currently on the disk. E.g, you may extend a 35 track disk into a 40 track disk. The normal way to extend a disk is to use Super Utility's Standard Format routine. This is a quick and highly automatic method. I describe it in this book in the Miscellaneous Chapter. Another way to extend diskettes is via the Format without Erase routine.

Suppose you want to turn a 35 track diskette into a 40 tracker using Format without Erase. You'd go to the Format menu and select the Format without Erase option for the disk in question. When you specified the drive number, you'd use an override command to indicate the new number of tracks--e.g, if the target disk were on drive zero, you'd answer the 'Drive(s) ?' prompt with

0=40

When Format without Erase started the 36th track (track number 35, since the first track is numbered zero), you'd get an I/O error, because that track was as yet unformatted. You'd enter <S> to skip the sector, and receive a similar message for the next sector. To add five tracks to a single density diskette (at ten sectors per track), you'd have to skip 50 sectors. If you were extending a 35 track double density diskette (18 sectors per track) to 40 tracks, you'd have to enter <S> 18*5 times, for a grand total of 90 S>kips.

This is certainly an unnecessarily laborious way to approach the task of extending a disk--especially when you could achieve the same end so easily with the alternate technique I describe in the MISCELLANEOUS Chapter. But if you were to follow the above procedure, the Format without Erase routine would faithfully update the directory to the new track-count, as soon as it verified the reformatting process. The second thing Format without Erase does when it updates the directory is to liberate any tracks which were previously locked out. This means that you'll have more free space available. But if a gran had been locked out because the disk was flawed, data later saved to that gran may be in jeopardy. So if you rescue a questionable disk with Format without Erase, back it up a couple of times and throw away the original. Or keep the original for a scrap disk--but don't entrust important data to a disk which may be physically flawed. After updating the directory, Format without Erase will tell you how many sectors it couldn't read on its first (reformat) pass, and how many it couldn't read on its second (verify) pass. Remember, the data in any sector which failed on the first part has been lost, even if it was readable on the second pass.

This is the end of the Format W/O Erase section. Return to the Problem Recipe.

STUBBORN ERRORS

There are two kinds of stubborn errors: 1) peccadilloes, and 2) nightmares (sometimes there's just no getting around technical language). I'll consider each in turn.

1) Peccadilloes

Peccadilloes are errors which result in the the Super Utility error message: 'DATA CRC Error'. Remember, every sector has a data subfield, and a two byte CRC is recorded at the end of that field each time data is written to the sector. "DATA CRC Error" means that there's a discrepancy between the CRC recorded when the data was written, and the CRC which the FDC computes when it reads the data at a later time. This indicates that something in the data subfield has changed since the sector was last written to.

There are two ways such a change could happen. One way, which I'll call a data peccadillo, occurs when one or more bytes of actual user data gets altered. The other

way, which I'll call a CRC peccadillo, occurs when one or both of the recorded CRC bytes get corrupted, while the user data remains intact. Either type of peccadillo results in disagreement between the recorded CRC and any freshly computed CRC.

Both kinds of peccadilloes can be fixed just by reading and rewriting the problem sector with Zap's Display Sectors option. Ask Zap to display the problem sector. When you get the "DATA CRC Error" message, just select the mini-menu's <S> option. Super Utility will then display the data as it found it.

Next, you enter the modify mode by pressing <M>. The last step is to key <ENTER> <U> <ENTER>. (Editor's note: <ENTER> <ENTER> would work just as well.) This resaves the sector. When the sector is resaved, the FDC automatically recomputes new checksums, which are in accord with the rest of the data, and saves them on the disk in place of the old ones.

If the peccadillo in question was a CRC peccadillo, all this is fine. But if the error was a DATA peccadillo, the data in the "fixed" sector will still be corrupt, though Super Utility, DOS, and other software will now be able to read the sector without generating an error condition.

Unfortunately, both types of peccadillo look the same to the FDC. Only the user has a chance of telling which kind took place. The best thing to do is to inspect the sector while you're still looking at it with Zap. If you're looking at a word processing file, editor/assembler source file, or other ASCII text, it should be fairly easy to spot corrupt bytes. On the ASCII side of the display, you'll see graphics, or other meaningless characters, where there should be legible material. Just enter the modify mode (making sure to set data-entry to ASCII), and type over the garbage.

A word of warning is in order. Sometimes a word processor's text formatting characters, or other information, may look like garbage. If you have Lazy Writer files with underlined sequences, those sequences will look like trash to any disk monitor, including Super Utility. Editor/assemblers often have line numbers or file headers with high bits set. Again, these can look like garbage when viewing disk sectors. I haven't yet seen Mumford's Instant Assembler, Disk Version, but I understand it uses a space saving tokenized source format, both in memory and on disk. Such files will probably also look trashy when examined with Super Utility. So always try to know what you're doing before zapping a disk. If possible, look at the corresponding sector in a backup file, before making any changes.

If the file is not a text file, than it's much more difficult to decide whether your peccadillo was CRC or data, and if data, which data. If you're looking at a BASIC program, in its normal compressed format, than you would have to hand detokenize it, and sort out all the line numbers, and pointers to line numbers, to get an idea of what belongs and what doesn't. This book is not the place to describe such procedures.

If you're looking at machine language object dump, things can get even more difficult. You would have to disassemble the code, and see if anything looks screwy. In doing so, you would also have to sort out DOS's load file format codes. If you don't know what load file format codes are, there's some information on them in the 'PATCH SECTORS' section of this book. If you have no knowledge of machine

language, then this book can't help you decipher farkled object files. But if you have a backup of the file, you can just follow the formula described below.

Assuming you have a backup, it may be easiest to simply copy the backup file onto the problem file. However, sometimes this may be undesirable. If you've updated the problem file since the last backup was made, copying the entire backup file would wipe out the update changes. But copying only the afflicted sector would preserve the update work, providing that the target sector was not one of those affected by the update.

I've noticed that every time I mention something in this section, I seem to have to point out that there are two kinds of that thing. Now I must say that, for the purposes of this discussion, there are two ways a file may be backed up. One is by backing up the entire disk which contains the file. For some perverse reason, a disk which is a backup of an identical disk is called a "mirror image" backup, even though it's not a mirror image, but a true replication. The other way to backup a file is to just copy it over to a different disk, or a differently named file on the same disk. I'll call this type of backup a "file-backup."

In the case of a mirror image backup, the backup sector will be in the same relative position on its disk as the problem sector is on its own. But when you work from a file backup, the problem sector and its backup sector will probably occupy different places on their respective disks (especially if both the problem file and its backup are on the same disk). But the problem sector will still have the same relative position within each file. In other words, the target sector should have the same File Relative Sector Number (FRSN) in both the problem file and backup.

When copying a problem file from a mirror image backup, you can use Zap's Copy Sectors option. Just specify the same track and sector for both the source and destination, varying only the drive number if you're using more than one drive.

Note that if the damaged file has been updated or changed in any way since the backup was made (other than just being extended), then the methods here probably won't work. Check the sector to be copied, the previous and subsequent sectors, and any other source handy (printouts, your recollection, etc.) to determine if the replacement sector is the EXACT replacement required. If not, then you must evaluate the effect of putting in an "almost" exact replacement, or opt for going to more elaborate procedures of recovery (see recovery tips). If the file is text, such as word processor files, it may not be too critical, but at the other extreme, machine code (a CMD program) will be intolerant of even a one-bit flaw!

If you're working with a file backup, there are two possible procedures which are useful for finding the proper backup sector. Which is better? That depends on the individual situation and your temperament.

METHOD A

First, determine the FRSN of the problem sector. To do this, go to Super Utility's File Utilities section and run option one, Display File Sectors. Enter the name of the problem file. If the disk containing the backup file is also on line at the time, be sure to include the correct drive number as part of the file spec.

Super Utility will find the file and display various facts about it, and then prompt you with 'Choice ?'. At that time, it's waiting for you to enter the FRSN of the sector you want displayed. Take a guess at the proper FRSN or just press <ENTER> to default to zero (remember, the number of the first item in a relative count is usually zero). Use the arrow keys to step through the file until you come to the problem sector. You can identify it by its track and sector number, which are displayed on the left side of the screen, just as they are in the ordinary Zap mode. Note the problem sector's FRSN. This will be displayed near the screen's lower left hand corner, to the left of the hex digits "E0"--under the heading 'RSEC', for Relative Sector. If the problem prevents Zap from loading the sector, you can still determine its FRSN by the FRSN's of the adjacent sectors. Next, use Display File Sectors to look at the backup file. When Super Utility has located the file and asks you to enter your choice of sector, enter the FRSN which you have just determined on the original. The sector displayed should be the backup of the problem sector. Make a note of its track and sector number as displayed at the left side of the screen, next to the hex digits "70" and "90", respectively.

You can now use Zap's Copy Sectors option to copy the backup sector onto the problem sector.

METHOD B

Use File Utilities' Compare Files to compare the problem file with its backup. If all goes well, Super Utility will report data differences in only one sector, and that should be the problem sector.

Super Utility will report the differing sector(s) in terms of its FRSN. Take note of this number and go to Super Utility's File Utilities section. Use Display File Sectors to view the problem file. When you see the 'Choice ?' prompt, enter the FRSN displayed by Compare Files. Super Utility will try to read and display the sector. If it succeeds, the track and sector numbers will be displayed in the usual Zap format. If it fails, they will be displayed in the error message. In either case, note them down. Then do the same with the backup file. Finally, copy the backup sector onto the problem sector.

2) Nightmares

An ancient computer blessing goes, "May your nightmares all have backups." This is especially true when a nightmare error occurs on a directory track (except for the HIT and GAT sectors. If that should happen, only a mirror image backup of the disk can restore things--unless you want to try to hand reconstruct the blown directory sector.

There is a major decision you must make in dealing with nightmares--you must choose and follow one of two divergent paths. 1) Format without Erase, and 2) Track read reconstruct.

If you have a backup, the decision is easy. Format without Erase, and then recopy the lost sector(s). But remember, Format without Erase does destroy the data in nightmare sectors--only compliant errors are cured.

If you don't have a backup, you may still opt for path 1. You'll lose some data, but at least you'll be able to reuse the disk (though I would consider a nightmare prone disk a poor choice for most projects). If the nightmare occurred in a directory sector (other than the GAT or HIT), you'll loose a great deal more than a single sector's worth of data. Any file whose primary directory entry was in the damaged sector will become inaccessible. Files whose extended directory entries were contained in that sector will probably become useless as well.

Since a directory sector can hold up to eight entries, you may loose eight files in one fell swoop. If one or two of those files are crucial system files, the entire disk may become inaccessible to you--especially if you have a one-drive system.

If you're willing to sacrifice the other files cataloged in that directory sector, the problem of the lost system files is one of the easiest to cope with. Most DOS's have their system files in the same place on every disk. The placement of the system files' directory entries is also fairly constant. So you can Format without Erase the nightmare disk. Then copy the corresponding directory sector from another disk onto the nightmare sector. Just make sure that the source disk has the same DOS as the nightmare disk, is in the same density, and has the directory on the same track.

Finally, you might want to use Zap's modify mode to zero out those sections of the directory that don't apply to the system files. Non-system directory entries are probably invalid for the disk you copied them to. If you prefer to experiment, you may leave the entries in the directory and try to access the files they describe. If the files don't work, then kill the the files, or go back and zero their directory entries.

In case you're not at all familiar with directory structure, Figure 6 depicts a typical directory sector. It contains eight entries, each of which takes up two lines of the display. If you look at the ASCII side of the display, you can make out the file names embedded in the first line of each entry.

```
#     00#5F00 0000 0053 5953 3320 2020 2053 5953#_....SYS3    SYS
HEX   10#EB29 210E 0500 1200 FFFF FFFF FFFF FFFF##)!.....########
DRV   20#0000 0000 0000 0000 0000 0000 0000 0000#................
 0    30#0000 0000 0000 0000 0000 0000 0000 0000#................
TRK   40#1E00 0000 0042 4153 4943 5220 2043 4044#.....BASICR  CMD
 17   50#782F 9642 1700 1D24 FFFF FFFF FFFF FFFF#X/#B...$########
TRU   60#1000 0005 0047 4554 4449 534B 2042 4153#.....GETDISK BAS
 17   70#9642 9642 0700 1B01 FFFF FFFF FFFF FFFF##B#B....########
SEC   80#0000 0000 0000 0000 0000 0000 0000 0000#................
 07   90#0000 0000 0000 0000 0000 0000 0000 0000#................
STD   A0#1000 00CF 0044 4953 4B44 554D 5042 4153#...#.DISKDUMPBAS
OSD   B0#9642 9642 0300 1820 FFFF FFFF FFFF FFFF##B#B... ########
      C0#1000 00AE 0047 4554 5441 5045 2042 4153#...#.GETTAPE BAS
      D0#9642 9642 0500 1C01 FFFF FFFF FFFF FFFF##B#B....########
      E0#0000 0000 0000 0000 0000 0000 0000 0000#................
+00   F0#0000 0000 0000 0000 0000 0000 0000 0000#................
```

Figure 6

The file name extensions are all the way to the right. Those files whose extensions are SYS are system files (Multidos uses "DOS" instead of "SYS"). So if a line ends with SYS (or DOS), leave it and the following line alone when you zero out the non-system entries.

Note: A TRSDOS double-density (Model I or III) directory does not code the system files in the "normal" manner, so you will not see any entries for system files in the directory of a TRSDOS III disk. If you must attempt to recover or reconstruct a farkled TRSDOS III system file, consult your Super Utility Manual for the proper procedures.

Finally, run Disk Repair and use the Repair GAT and Repair HIT options. The system files should now be accessible to the system, and the disk should boot (providing it was a booting disk to begin with) and behave normally--except that you've lost access to some of your files.

If a nightmare occurs on a HIT or GAT sector, use Format without Erase to make the sector usable. Then run Repair HIT or Repair GAT as required.

RECONSTRUCTING A TRACK

If you just can't give up on salvaging your nightmare ridden sectors, no matter how rocky the road nor how unlikely your prospect of success, try this method.

DON'T Format without Erase--yet! Go into Super Utility's Memory Utilities section. Notice that item 15 is labeled Track to Memory. This function may be able to help salvage at least part of your lost data. The reason for this is that it performs a track read. There are two different ways the FDC chip can be told to read or write information--sector or track. Normally the sector method is used, except during formatting. Track reads aren't as reliable as sector reads, because they dispense with some of the safeguards I described in the technical introduction. A track read does have one advantage in dealing with farkled disks, however--namely, the FDC does not go into a tizzy if a sector ID is mangled or missing, or if a CRC does not compute. In fact, it doesn't treat ID's or CRC's as anything out of the ordinary. It reads in everything on the track as data--including gaps, formatting marks, file load format codes, and all.

As you may recall, the FDC won't complete a sector read if it encounters an ID error, or header CRC error. So by performing a track read you may get data into your computer that the FDC wouldn't look at during a sector read.

Assuming that your track read succeeds completely--and that's a pretty big assumption--you might think that you only have to do a track write to get the information back on the disk in usable form. Uh-uh! Here's where one of those little FDC kinks steps in to make a difficult situation even more difficult.

Believe it or not, if you do a track read of track A into a memory buffer, and track write that buffer onto track B, track A and track B usually don't end up containing the same data. That's because a track read reads it all in like it is, but there are certain bytes which a track write changes. For instance, trying to write an F7 hex during a track write, will not put an F7 on the disk. Instead, it will write two

CRC bytes, which might have any value. Trying to include any byte from F8 to FF hex in a track write, will result in the specified byte being written, but with an altered clock frequency. When the FDC later tries to read it during normal sector I/O, it will look like an address mark instead of a data byte.

I'll give you the general strategy for track read reconstruct, and then take a closer look at some of the details. First, of course, you do a track read of the nightmare track into a memory buffer. Then you separate the wheat from the chaff. That is to say, you look at all the stuff read into the buffer and try to figure out what represents genuine user data, and what represents genuine garbage. Then you get all the good stuff down on paper. Finally, you reformat the nightmare track and use Zap's Modify Memory mode to reenter and save the data, sector by sector.

Now for the gruesome details: When you do the track read, make sure the drive is configured correctly for the disk. It is especially important to do the track read in the proper density. When you ask Super Utility to do the actual read, it will request the drive and track number. After you provide that information, it will ask for the start of the buffer into which the track will be read. It's usually a good idea to default by pressing <ENTER>. Super Utility will select a safe buffer.

Finally, you will get the prompt, 'Sync to ID marks ?'. Super Utility asks this because there are two ways the FDC may be instructed to track read--with ID sync or without. If you elect to sync, the FDC will try to sync with ID fields whenever it encounters them. If you elect not to sync, the FDC will just read everything it can, from the first byte on the track, making no attempt to resynchronize when it encounters an ID field.

One would expect that track read with sync would invariably produce more reliable results than track read without. Actually, the reverse seems to be the case. You should try several track reads of each kind on your nightmare track. Each time, inspect the result carefully. You'll find that often even two track reads of the same kind do not produce identical results. Choose the with or without sync option that seems to work best for your problem. When you have finally decided that the material you've track read into the buffer is as good as it's going to get, it's time to start in on the wheat and chaff thing. Figure 7 shows the result of track reading a formatted Model I TRSDOS track--a "virgin" track which contains no data other than formatting information and "fill" bytes. Figure 8 shows a similar track after files have been saved on it. Looking at the first figure, you will notice the track is buffered in memory starting at location D400 hex. It is laid out in the manner described earlier. First there a rather long gap. The track read may pick up some garbage at the beginning of the track as well, but in this case, the read was clean. Skip through the bytes until you come to the first FE hex in the buffer (in the example, it's displayed at memory location D411 hex). That marks the beginning of the first physical sector on the disk.

Following the FE is the byte sequence 06 00 00 01. The first three bytes indicate that the track number is six, the head number is zero, and the sector number is zero. The one at the end of the sequence is the length code. In IBM standard (which TRS-80 DOS's use), one means a 256 byte sector. The next two bytes (D6,4A) are the CRC for the header-subfield.

```
   D400#FFFF FFFF FFFF FFFF FFFF F000 0000 0000############.....
HEXD410#0FFE 0600 0001 D64A FFFF FFFF FFFF FFFF#.#....#J########
MEMD420#FFFF FFFF 0000 0000 0000 FBE5 E5E5 E5E5#####.......######
   D430#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D440#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D450#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D460#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D470#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D480#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D490#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D4A0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D4B0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D4C0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D4D0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D4E0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
+00D4F0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################

#  D500#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
HEXD510#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
MEMD520#E5E5 E5E5 E5E5 E5E5 E5E5 E5A4 0CFF FFFF##############.###
   D530#FFFF FFFF FFFF FFFF FF00 0000 0000 00FE##########......#
   D540#0600 0501 29BF FFFF FFFF FFFF FFFF FFFF#....)############
   D550#FFFF 0000 0000 0000 FBE5 E5E5 E5E5 E5E5###......########
   D560#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D570#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D580#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D590#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5A0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5B0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5C0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5D0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5E0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
+00D5F0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################

   D600#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
HEXD610#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
MEMD620#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D630#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D640#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D650#E5E5 E5E5 E5E5 E5E5 E5A4 0CFF FFFF FFFF############.#####
   D660#FFFF FFFF FFFF FF00 0000 0000 00FE 0600########......#..
   D670#0101 E57B FFFF FFFF FFFF FFFF FFFF FFFF#..#{############
   D680#0000 0000 0000 FBE5 E5E5 E5E5 E5E5 E5E5#......##########
   D690#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D6A0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D6B0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D6C0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D6D0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D6E0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
+00D6F0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
```

Figure 7

```
   D400#FFFF FFFF FFFF FFFF FFF0 0000 0000 000F############.......
HEXD410#FE06 0000 01D6 4AFF FFFF FFFF FFFF FFFF##....#J#########
MEMD420#FFFF FFFF FFFF FFFF FFCB 7EB7 2804 FE00############~#(.#.
   D430#2819 2C20 F53A 5451 3C20 060C 79FE 0438#(., #:TQ< ..Y#.8
   D440#E03E 18B7 C9C1 E1C1 E118 E7E5 C545 CDC1##>.######.###E##
   D450#4A28 03C1 E1C9 E5C5 CB7E 20E9 3E05 856F#J(.######~ #>.#O
   D460#115D 5106 0B1A BE20 DC23 1310 F8C1 7932#.]Q...# ##..##Y2
   D470#5451 E1F1 F1C5 E57E E607 4F06 003E 1085#TQ#####~#.O..>.#
   D480#6FED 5B6A 51E5 21A2 61AF ED52 E118 2179#O#[JQ#!#A##R#.!Y
   D490#FE07 2816 7E23 E566 6FAF ED52 E128 1123##.(.~##FO##R#(.#
   D4A0#417E 2366 6FAF ED52 2806 E1C1 3E19 B7C9#A~#FO##R(.##>.##
   D4B0#E178 32B2 4FC1 C3A7 4FCD 9648 CD15 4EC8##X2#O###O##H#.N#
   D4C0#FE18 C03E 1032 1D4F 3A54 514F 3C20 014F##.#>.2.O:TQO< .O
   D4D0#CDFD 5020 0B38 09CD 2E51 C0CD AB50 2801###P .8.#.Q###P(.
   D4E0#00FE 4E0F 3A54 513C 2006 0C79 FE04 38E4#.#N.:TQ< ..Y#.8#
+00D4F0#3E1A C945 3A4E 4E77 CD41 51C0 CDC1 4AC0#>.#E:NNW#AQ###J#

#  D500#E536 0023 3600 2336 0023 3600 233A AF4F##6.#6.#6.#6.#:#O
HEXD510#7723 C501 0F00 EB21 5D51 EDB0 EB36 0023#W##...#!]Q###6.#
MEMD520#3600 2306 0A36 FF23 10FB FC4B FEFF FFFF#6.#..6##.##K####
   D530#FFFF FFFF FFFF FFF8 0000 0000 0007 FE06#########.......#.
   D540#0005 0129 BFFF FFFF FFFF FFFF FFFF FFFF#...)#############
   D550#FF00 0000 0000 00FB E5E5 E5E5 E5E5 E5E5##.......#########
   D560#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D570#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D580#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D590#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5A0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5B0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5C0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5D0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D5E0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
+00D5F0#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################

   D600#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
HEXD610#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
MEMD620#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D630#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D640#E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5 E5E5##################
   D650#E5E5 E5E5 E5E5 E5E5 A40C FFFF FFFF FFFF##########.######
   D660#FFFF FFFF FFFF 0000 0000 0000 FE06 0001#######......#...
   D670#01E5 7BFF FFFF FFFF FFFF FFFF FC00#.#{############.
   D680#0000 0000 01FB C1CD D64A E1C0 CDA7 4F37#.....####J####07
   D690#C9DD 4E06 CD2E 51C0 DD7E 07CD B650 3E1E###N.#.Q##~.##P>.
   D6A0#C045 7D32 A24F 54DD 5E07 1A77 CD41 51C0##E}2#OT#^..W#AQ#
   D6B0#CDC1 4AC0 3690 2CC5 3ABE 4A77 2C06 1436###J#6#,#:#JW,..6
   D6C0#002C 10FB E506 0A36 FF2C 10FB D113 C1CD#.,.##..6#,.##.##
   D6D0#D64A C03A BE4A 47CD C14A C07D C61E 6F36##J#:#JG##J#}#.O6
   D6E0#FE2C 3600 CDD6 4AC9 EBDD E5E1 3680 233E##,6.##J#####6##>
+00D6F0#00B7 3E00 2802 F680 F620 7723 3600 23D5#.#>.(.### W#6.##
```

Figure 8

Next we have another gap consisting, in this case, of 12 FF's and six zeros, for a grand total of 18 bytes. Then comes the Data Address Mark, FB.

After the DAM comes the data. The track has not had any files saved to it yet, so the data field contains what ever "fill bytes" DOS used for formatting. As you see, in this case the fill byte is E5, and there are exactly 256 of them.

E5 hex is the "worst case" data pattern for single density disk I/O. That means that if a physical disk sector is in marginal condition, a series of E5's is the data pattern most likely to cause an error. The worst case pattern for double density is the byte pair 6D B6.

Most DOS's fill data areas with worst case patterns when they format. This is so you'll become aware of error prone disks before you entrust vital data to them.

After the E5's comes the byte pair, A4 0C. This is the data CRC. Finally, we have another 18 byte gap, another FE sector ID, and all the shenanigans start again with sector five. And so it goes.

If you examine the second figure, you'll notice it's pretty much the same as the first, except for the sector data, and the data CRC's. That should give you an idea of what a track-read track should look like when nothing has been munched. Now suppose you track read a nightmare track. When you examine the buffer, you should see a pattern like the one in figure 8--up until the point where the disk got clobbered. If you're lucky, there'll only be a few bytes of garbage, and then normality will resume. Unfortunately, here's what is much more likely to happen: after the FDC passes over the garbage, it will be out of sync with the remaining data on the track. That is to say, when it tries to read the garbage, it will lose some bits, or pick up extra ones. Thereafter, even if the FDC reads all the subsequent bits faithfully, they will be not grouped into bytes correctly. In effect, the data will become shifted. Since shifted data looks about as much like garbage as genuine garbage, this can make it difficult to figure out what's going on in the track. Here's where Super Utility's decryption features come to the rescue. That's right, all that encrypting-decrypting stuff isn't just for people who want to play with secret messages. It can help you save your disks!

What you have to do is retry the track read operation several times, with and without sync. When you're satisfied that you've got the best results that you're going to get, examine various parts of the garbagy looking stuff, and use the decryption facility to shift through all eight possible phases. The way to do this is well covered in your manual. Briefly, what you do is go into Display Memory and look at a trashy part of the track read buffer. Make sure you're in the paging mode (not the modify mode), and press <@>. This makes the DCR (for DECRYPTION) prompt appear near the bottom left hand corner of the screen. Then enter <:>. This will cause all decryption changes to be visible on both the hex and ascii portions of the display.

Press <@> again to get the DCR prompt back. Then enter SL1 (for Shift Left 1 bit) or SR1. If the display that results looks like what you're hoping for, congratulations--you've probably struck pay dirt! Do a screen print, or, if you don't

have a printer, copy the data by hand. If a shift of one doesn't do the trick, try again with shifts of two, three, ..., through seven (Note: for a fuller treatment of the decryption mode, see the Undocumented Features Chapter).

Even if you do manage to capture meaningful data, you'll have to know what it looks like in order to recognize it. Of course, there's always the chance that you can spot true data by the pattern of gap, header ID, gap, DAM, 258 bytes of what ever (the 256 byte data field plus two bytes of CRC), gap, etc. That's why I detailed the pattern. But things are a lot easier when you know what the data looks like.

If the track contained ASCII word processing material or the like, half the battle's already won. If it contained a machine language program, it would help if you knew some of the code. If such is the case, you may find what seems to be a sequence of the program that you're looking for, except that extraneous bytes are thrown in. These could be load file format codes. For more information on load file format codes, see the section of this book which deals with patch sectors. Also, your favorite DOS manual may shed some light on the subject. My own introduction to the topic was via the technical section of my LDOS manual.

If nothing you do makes your lost data appear in recognizable form, you're just out of luck. However, if you do manage to tune the data in, your work's just begun. Print it all out in sequence. Isolate the sector (or sectors) which were farkled by the nightmare. It will be missing the header subfield (or part of it,) or the DAM, or else the header-subfield CRC will be wrong. If an ID or DAM is missing, study the printout and figure out where it belongs. Part of the data will probably be gone too. Mark the printout to indicate what's missing from where. If you reconstruct the sector correctly on paper, you'll get it right on the disk too. Now you're ready to start setting up the disk. Since it's always safer to work with backups, backing up the farkled disk should be the first thing you do. Use Super Utility to make the backup, and just S>kip the problem sector(s) when Super Utility prompts you with the mini-menu. Then go into Zap and display the nightmare sector on the backup. Use the modify mode to type in the sector data. If more than one section was nightmared, repeat as necessary--and good luck!!!

Refer to RECOVERY TIPS for more help on recovery.

DIRECTORY REPAIRS

After all else has been done, it's time to check, and if necessary repair, the directory. Simply go to Super Utility's Disk Repair module (item five on the main menu) and select option nine, "Check Directory." If Super Utility reports zero errors, that's it. If Super Utility reports a HIT or GAT error, run Repair HIT Sector or Repair GAT Sector.

In Repair GAT Sector, you will be asked whether you want to fix just the allocation table, or the entire sector. Usually, just fixing the allocation table will be sufficient. Fixing the entire sector also replaces the disk name, date, master password, and auto commands with Super Utility's defaults. You'd want to do that is if those fields contained garbage. You'd know that was the case if Super Utility displayed trash whenever it claimed to be reporting a disk's name and date.

If Check Directory, or any other operation, results in a message to the effect that the directory can't be located, you probably have to run Read Protect Directory. Since, when you have this problem, Super Utility can't locate the directory on it's own, it can't read protect it without your help. It will ask you for the track and sector the directory starts on, and its length in sectors. 35-40 track diskettes normally have the directory on track 17 decimal. 80 track disks usually have the directory on track 40, to keep it in the center of the action. NEWDOS-80 and DBLDOS may have it elsewhere. Directories are usually one track long--that's ten sectors for single density, and 18 for double. Again, don't take anything for granted with NEWDOS-80.

If you're not sure about the directory's location and length, go Zapping through the disk until you find it. You've already seen what a typical directory sector looks like. That was a file-location sector. The first sector in a directory is the GAT. It contains the disk name, date, master password, and auto command in its last several bytes. That should help you recognize it. Figure 9 is a sample GAT. The second directory sector is the HIT. HIT's look pretty garbagy. Figure 10 is a typical HIT sector.

```
#     00#FFFF FFFF FFFD FCFC FCFC FCFC FCFC FCFC##################
HEX   10#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
DRV   20#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
 0    30#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
TRK   40#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
 17   50#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
TRU   60#FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC##################
 17   70#FCFC FCFC FCFC FCFC FCFC FCFC FCFC FCFC##################
SEC   80#FCFC FCFF FFFF FFFF FFFF FFFF FFFF FFFF##################
 00   90#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
STD   A0#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
OSD   B0#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
      C0#FFFF FFFF FFFF FFFF FFFF FF21 0000 E042#############!..#B
      D0#5452 5344 4F53 2020 3034 2F30 352F 3830#TRSDOS  04/05/80
      E0#0D0D FFFF FFFF FFFF FFFF FFFF FFFF FFFF#..###############
+00   F0#FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF##################
```

Figure 9 - A typical GAT Sector

After you have located and read protected the directory, rerun the Check Directory program, and run any repair programs indicated. There is one case when Check Directory will be unable to find the directory even though the disk is a perfectly good one, and that is when you try to read a Model I TRSDOS disk (single-density) on a Model III. The problem is that the DAM used by Model I TRSDOS for the directory track is not one that the Model III can read! By using "Read Protect Directory", you can make the Model III read your Model I disk directory, but the process opens up a new can of worms: now Model I TRSDOS will not be able to find the directory!

In order for Model I TRSDOS to be able to use this disk again, you must boot Super Utility on a Model I, put in the disk which you read-protected on your Model III, and repeat the process. This time the directory will be re-written to the disk with the correct DAMs.

```
#     00#A22C 2E2F 2C2D 2A2B 0000 0000 0000 0000##,./,-*+........
HEX   10#0000 0000 0000 0000 0000 0000 0000 0000#................
DRV   20#2800 0000 0000 0000 0000 0000 0000 0000#(...............
 0    30#0000 0000 0000 0000 0000 0000 0000 0000#................
TRK   40#F2C5 0074 006C 0000 0000 0000 0000 0000###.T.L..........
 17   50#0000 0000 0000 0000 0000 0000 0000 0000#................
TRU   60#0000 0000 F069 0000 0000 0000 0000 0000#....#I..........
 17   70#0000 0000 0000 0000 0000 0000 0000 0000#................
SEC   80#0000 0000 E300 0000 0000 0000 0000 0000#....#...........
 01   90#0000 0000 0000 0000 0000 0000 0000 0000#................
STD   A0#0000 0000 0080 0000 0000 0000 0000 0000#.....#..........
OSD   B0#0000 0000 0000 0000 0000 0000 0000 0000#................
      C0#0000 0000 007C 4B00 0000 0000 0000 0000#.....|K.........
      D0#0000 0000 0000 0000 0000 0000 0000 0000#................
      E0#0000 0000 0000 0000 0000 0000 0000 0000#................
+00   F0#0000 0000 0000 0000 0000 0000 0000 0000#................
```

Figure 10 - A typical HIT Sector

RECOVERY TIPS

There are more things that can be said about error recovery than there are pages in this book. More is being added all the time because of the changing computers, DOSes, programs and whatever. So, with that in mind, I will cover as much as time and space allow.

First, let me say that there is a wealth of information already given in the SU manual and this book if you only take the time to read and apply it. Further, there are references mentioned in this book, and others to be found in computer stores and magazines, that can be of help. Most are not very expensive, or difficult to use ... considering the cost of the lost data. So, take a little time and start a library of information on files, DOSes, disk controllers and the like. Use them as often as practical. You might try fixing a bad disk when the time can be afforded and a backup is already available, just to get the feel of it. Nothing can be lost, but a lot of knowledge and experience can be painlessly (well, almost painlessly . . .) gained.

I'll try not to be too repetitive here about stuff covered elsewhere in this book or the manual. I will try to point you in the right direction and explain things that may not be explained clearly elsewhere.

For instance, it has been explained before that backing up a disk to be recovered is highly recommended. This is for many reasons. I will try to give you enough reasons to make you think about it. The important part though, is THINK.

Here are some of the reasons for the backup (copy) to be made. First, because the error is on this disk, the disk itself should be suspect. They wear out through use, sometimes just sitting in the drive. The drive in most computers is being accessed quite often, some drives even run all the time, in others, the head is constantly

loaded or clicks on and off repeatedly as used or even any time any other drive is used. The liner of the diskette is slightly abrasive to pick up any dust particles, ashes and whatnot. Thus they are good for the short run but help accelerate wear in the long run.

Along this same line, diskettes that are used "flippy" style, whether by punched holes or in "flippy" drives will wear faster than single sided ones. This is because the diskette is turning backwards when flipped over. This causes more of the particles picked up in the liner to shift about and possibly abrade the disk, which only has so much magnetic material to wear off before the performance goes down.

In any case, a disk that has a problem should be considered untrustworthy, at least until proven otherwise. So make a backup and reformat the suspect disk as a precaution. Of course, only reformat AFTER the recovery is fully complete! If it can't pass this backup without errors, don't trust it with important data in the future.

Second, when working on a damaged disk, it is your only copy of the data to be saved (assuming no complete, up to the minute backups are available). This means that any goof, such as ZERO SECTORS instead of COPY SECTORS or FORMAT instead of FORMAT WITHOUT ERASE is the end of the recovery. There is no room for error. Further, the disks original data, unless it is totally wiped out is still on the disk, even if it can't be read with the tools and techniques at hand. You would be surprised at what can be done by an expert with the proper tools! And one may pop up at the next club meeting or routine maintenance check. Don't get caught saying "If I only knew you were going to be here -- I formatted it!"

A third reason for backups is the original is the only authority to consult, that is, if there is any doubt about any data on the disk, the original may have the only correct answer. It may be lost do to damage, but it is sure to be lost if it is formatted over, or FORMAT WITHOUT ERASED. I recommend using this last command to refresh diskettes, not recover data. It is too easy to lose data with it and the data is totally lost after the FORMAT (the unreadable data is lost not the ok stuff). The outdated (or whatever) backups should be left in unmolested condition as much as possible too, for the same reasons, it's the only good backup you have.

Only massage the copies of the original or backups. Only use the originals for making backups (working copies) and data retrieval, never write on them. When the data is FULLY recovered or given up as totally lost, then FORMAT the disk to see if it can be put back in service or should be discarded. If there is partial recovery, keep the disk(s) around, something may pop up later to allow more to be recovered, such as a hot shot SU4 or 80 track drives where none were available. Diskettes are much cheaper than data.

This is my fourth and last reason for making and working on backups whenever possible. Diskettes that are old or worn may not have much magnetic material left on them and that may be the cause of the errors and lost data. The more they are used, the worse the situation gets. The best thing is to use them as little as possible, doing as much recovery on the copy as is possible.

PREPARATION

We need to take a moment here to prepare for the work ahead. It is always a good idea to have several fresh, empty diskettes handy, and a few with a minimum DOS system on them. Next, after we have the work diskettes ready, we need to have a note pad and pencil, this is mandatory. There are just too many things to do and see to rely on memory. A printer is a big help, but no substitute for good note-taking.

Last, you need to keep as many reference materials as possible close at hand. Manuals, books, articles, notes, newsletters, anything that may help. If you don't already have one, start assembling a library of information on your system, important programs and related things of interest. You will never regret it. This goes for programming, hardware repair, testing and even user information. In my book, you can't have too much. It would help if you had related information to this specific recovery handy too, such as printouts, flow charts, and similar material. Also, the most recent users may be able to shed some light on how it was damaged, which could help in its recovery. Use everything that you can, if the importance warrants it.

BACKUP / COPY

Ok, let's look at backing up. SU has enormous flexibility for this. Its very smart too. Let SU do its best first. Refer to the manual on BACKUP. When sectors can't be copied, note them on paper, this is just the beginning.

Note: What is unreadable today on this computer/drive combination may be recoverable on another. The temperature of the day and many other environmental considerations (humidity, AC line voltage, etc.) can affect the ability of a computer to do its job. Some computer disk controllers are more sensitive than others, some disk drives too. The alignment is sometimes critical. If possible, always use another computer and drive on the stubborn errors. The drive that actually made the sector that is now bad can sometimes be the best one to read it with, because of the exact alignment of the tracks. However, another drive may be far more sensitive and close enough in alignment to do a better job. Try as many drives and computers as is practical for the situation. This alone will cure more unreadables than you would imagine possible.

Also try reading 35- and 40-track disks on 80-track drives if you can. If the track is half good you have a better than 50/50 chance to get it. Be sure to try reading stubborn tracks and sectors with each of the half tracks available (80 track drives are exactly one-half the width of 35/40s and two of them are theoretically over each). The reason is that if there is any out of alignment, one of the two tracks will be more centered and get a better reading. Also, only half the track may be flawed, allowing the other to read ok.

In a pinch, try using a 77-track drive, it MAY be able to help. it can read quite a few of the tracks on 35s (and 40s up to its 35-track limit). It may even be able to read some of the tracks on the 80-track diskettes, though not many. It's worth a try if you have access to one.

So, now we have the "bad" disk backed up, and there are some tracks and sectors that did not come along. Put away the original bad disk for now. I think the first thing I would do is try a DIR, if that works Try copying the wanted file(s) to another disk. If THAT works, test the files to see if they are correct. That means run them if they are programs, load them if they are data (to whatever uses them) etc. until you are sure that you have the right stuff. Don't get fooled with Murphys NAME IN THE DIRECTORY ONLY trick. Verify, in whatever way you can, that you have the whole and complete data you want.

On the other hand don't go to a lot of trouble fixing sectors that don't need to be fixed. This little section is on verifying that the bad sectors are indeed needed (belong to important files, etc). Use the commands in the FILES utility that tell which programs are allocated to which tracks. Check to see that the bad sectors are in the files that need to be restored. It is hard enough to restore just the needed files, without trying to restore files you have a good copy of. If you want the practice without the pressure of having to do it, well, that's a good reason to try. Just don't get fooled into thinking you have to.

Of course there are times when the quick cure of above does not work or is not feasible. Perhaps you are trying to save an only copy of a self booting program disk or the latest zapped version of DINKYDOS and don't want to wait 3 weeks for another copy. Now we get serious.

EASY STARTING

Assuming we have at least one copy of the original bad disk to work with (more would be better, make the rest from the copy not the original, to save wear and tear), let's examine the possibilities. The original disk could be almost good, just a few errors to fix. That is Case 1. We may have the original, badly damaged (many unreadable sectors); Case 2. We may have working backups, outdated or otherwise, not suitable themselves as replacements, but possibly useful for repair work; Case 3. It is possible to have Case 2 and 3 or 1 and 2 together, or Case 1 or 3 alone.

Case one is the simplest, but as we attack it we will find the complexity grows as the numbers of errors and the kinds of errors are discussed. On the other hand this will prepare us for the other cases. The main force of attack is finding a reasonable method of retrieving a good working copy of the original data, whatever it was. We may need to make some sacrifices along the way in the interest of efficiency. It stands to reason that nearly anything can be recovered if enough time, effort, and resources can be applied.

After the backup revealed that there were still sectors that were not usable yet, we can start to use SU's special talents to try to recover them. Let's assume that there is only one sector that is giving us problems. Since we can usually repeat any procedure on each succeeding bad sector encountered, this will be the basis of a very powerful recovery technique.

The very first thing to do is to use the ZAP Utility. With SU loaded and waiting, press <SHIFT>-<BREAK> to get to the main menu. Since you got SU to do a backup, you know or can look up in the configuration table, the density, track count, sector size and count and many other important pieces of information about the diskette

under study. By the way, this is where notes and system printers are handy, copying down the configuration table, bad sectors and their errors, how many retries were required or tried on both marginal and "unreadable" sectors, etc.

PARITY ERRORS

The simplest error to fix is the parity error on a text file. That is because you can see the errors in the ASCII display (almost always, anyway), and the text file is the most tolerant of errors. Also, typing in corrections is easy, obvious, direct and it can get the job done.

Usually a parity error can be cured just by trying to read the sector, getting the error and then writing it back to the same sector. In our case, we are working on a copy of the disk so we don't have the parity error any more but we don't (or may not) have all the data right either.

Again, you must check to determine that the file is bad, won't load, has junk in it, etc. Then, going back to the original disk, try to do a few copies to your work disk on unused sectors. Use the R, C, and N options to get several best attempt sectors, noting, of course, where they are and what errors they gave. If you get one on the screen with no error, copy it to the correct sector and you should be on your way. If you get errors, parity in this case, it is time for a little experimenting.

The disk controller can sometimes get part of the data off a disk even though it gets an error. In fact, it can get the sector data correct but it is the parity that is wrong. So it pays to see what we have, and test it in whatever way we can. One way, as I have mentioned, is to try it and see if it works. Another is to compare it to a known good one. If we had that we would not be wasting time finding another one, we would just copy it. Suppose we could guess if it was right or guess what to change to fix it? A way to do something like that, is to use Compare Sectors to see if they differ. Thus we do several tries to read the same bad sector (copied to a "test" disk or a safe place on the copy disk). Alternatively, we could read in, off the bad disk, several attempts to read the bad sector to memory and then compare the memory "sectors". Be sure to read these into consecutive memory locations, and not the default set by SU, because SU will write the sector data to the same place each time it is given the choice (default) and overwrite all previous reads.

If we are lucky, the sectors will be different and the location of the difference will tell us where the error is, or at least something about the error. If we look at the sectors now we may even see the error, it may be a hex number where all ASCII is expected, or an ASCII character that is clearly out of place. Correct it in the real file on the repaired disk, and try it out again.

Maybe it is not obvious, but the errors were at the same spot. Make that byte a 20H (blank), as that is the most innocuous of ASCII characters, but use some logic on it. If the bad byte seems to be in a line number or string of identical characters, or other clue giving arrangement, make a guess and test it out. Remember you are working on the copy, you can't lose any more data, you can't hurt anything. If you get a wild idea (inspiration), TRY IT.

If the errors were all the same and the sectors are identical, examine the sectors carefully to see if you can spot the bad stuff. Remember ASCII is a small subset of the 256 hexadecimal numbers, ranging mostly from 20H (32 dec) to 7FH (127 dec), although they are officially defined from 0 to 127 decimal. The lower numbers are special, like 0DH is a carriage return, and those above 127 are graphics for TRS-80's. It should be easy to spot radical errors and take a little time to get more subtle ones.

That was the easiest error to work on. Ready to quit? I mean, go on?

Suppose the sectors are not being read very well, you might try using the READ TRACK command in the memory section of SU. It is not as reliable as the sector read normally but it can be better under some conditions (like CRC errors). It may get you some more data to use to compare, that has a different character than the sectors read separately, since this is a quite different form of read. That will many times give you things you could not get with sector reads.

Examine the memory where the track was read in to pick out the beginning of the sector in question. This will take some study as the memory image will contain the whole track including the sector and track IDs, filler material and all the rest (see discussion of track layout). It may also contain bad data because of the original problem, which will make things more confusing. Still, it's worth a try if you're desperate. I'll come back to track reads again.

Another possibility for those stubborn errors might be to look for "killed" copies of the file under investigation that you might get the needed sector from. Use the RECOVER FILES utility to see if there are any traces of the file in the directory that are good or at least good enough. It is another long shot but a possibility none-the-less.

If we suppose for the moment that nothing in the sector can be saved, the whole sector can be set to blanks and deleted the next time it is loaded into the editor or word processor.

A few cautions should be observed though, since Murphy is such a clever fellow. If the sector is to be filled with blanks (hex 20, 32 dec) it may be a good idea to add a few 0DH characters evenly dispersed in the sector. These are carriage returns that are required by some word processors such as Newscript. They usually won't hurt even if they are not needed. If you are working on ASCII BASIC files (saved with SAVE"NAME/BAS",A) or EDTASM (or similar) files, it may be helpful or even required at times to put in some bogus line numbers. You can usually examine the "good" text in the surrounding sectors to determine the exact format of the numbers and edit the offending sector with a compatible number.

For instance, in ASCII BASIC files, the line numbers are just ASCII numbers, 30 H to 39 H. Pick a number that falls in the correct range, that is larger than the previous good line number and smaller than the next good one. This is not even necessary in most cases but some DOSes don't "p" the lines as they are loaded in from disk (NEWDOS80 for example), so be on the safe side.

EDTASM files use a funny looking number that is just a regular ASCII number with the high bit set (don't ask me why!), so they look like B0 to B9 hex instead of 30 to 39. Again, make the bogus numbers fit into the sequence.

After modifying the bad sector(s), try loading the file into a word processor or editor, preferably the one it was made on so there is less chance of other conflicts. If it loads ok, scan the text till the bad stuff shows up, delete it, make any other changes required and save it again, right away, don't try anything else till it is saved. There are many situations that give the appearance that the data or file is ok to use, but turns out to be not quite ready. This can sometimes be cured by the simple procedure of saving and reloading. It has to do with renumbering, pointers, end of file indicators and all that esoteric stuff.

If it did not load and you got a DIRECT STATEMENT IN FILE error you didn't break up the lines into small enough lengths for BASIC to load in. Go back and add another line number near the beginning of the bad sector and another near the end. That should fix it. If it is other than that, you may have let some HEX characters through that the loader did not like, take a look and correct if necessary. If that doesn't turn up anything, try using ONLY 20H characters in the sector, or only 0DHs. Examine the documentation of the editor or word processor you are using to try to determine what is required by its loader. Also try to use another program, a different word processor or editor. Last (that I can think of) is try to copy a good sector over again onto the bad sector, adjust the numbers to be compatible and try again.

A slightly more difficult problem is the BASIC file in its binary or compressed state, as opposed to the ASCII files we have been working on. This one (and the ASCII one) can give some extra clues to help you when you try to load the bad file. It will usually load up to the point there is something wrong. Thus you can list it and note exactly where it stops listing, then go to SU and examine the area of the problem. Again you can usually "cut" out the problem area by fooling with the line numbers, if you can't determine what the real problem is and fix it.

Let's move on to more difficult problems. If you are trying to recover machine code or data files, it is likely that few users will have the expertise to be able to recognize small errors in the code. I would suggest using the compare sectors technique described above, possibly including the track read.

This may be one of the very few times that you could use another program. What may be of some help is a disassembler. It may be a real problem to get the code in loadable form, load it into memory, find the exact area that the problem is at, disassemble it and THEN if you are good, find and fix an error. I leave it to the reader as an exercise.

Suppose now that the errors were even more difficult, such as the dreaded CRC error. If it is a data CRC error, that is not as bad. It is similar to the parity error. The sector is still (usually) readable but the data has a fifty-fifty chance of being corrupted since the error could be in the CRC or in the data. Actually its not anywhere near fifty percent, there are many more data bytes that could be bad than CRC bytes, but you get the idea. Start reading and comparing again.

If it is a header CRC error then there are only two ways to recover a Case 1 error (no available backup). Try the repeated sector copy or read to memory and hope

you get a good read one time. That's one way. Or try the track to memory read. That is the other way. If they don't work, the only other possibility is to try on other drives and even other machines, hoping for more sensitive hardware. Use all the tricks I have told you about to ferret out the last scrap of useful information and good luck.

Case 2

Case 2 is just a bad Case 1, and it will just take repeated applications of the techniques described earlier. It is even more important to not do unnecessary work. So be sure that the files and sectors that you work on are necessary to the recovery of the data that you really want or need. Don't bother with system files, common utilities, empty sectors or killed files. Use of the FREE SPACE, FILE LOCATIONS, SECTOR ALLOCATION and DISK ALLOCATIONS utilities will help. Of course there must be a readable directory, both error-wise and DOS-wise (careful of odd ND80 diskettes).

There is not much more that you can do to badly damaged disks than can be done to individual sectors. Perhaps the read track feature may be more helpful on several consecutive bad sectors. Also, here it may be more worth your time to try to locate another computer system or two so you can get the advantage of differing drives and sensitivities, maybe even swapping system components to get the best combination.

For instance, try the system on some old and known poor disks by reading and formatting it with different combinations of drives and controllers (expansion interfaces). Do not use the bad diskette yet, wait till you have the best facilities. It will usually be obvious right away which is the best setup, because there will be notably less errors. Do the tests and recovery late at night to reduce the possibilities of AC line noise, or low voltage due to machines, air conditioners, and the like going on and off.

You may be lucky enough to have available 80-track (or even 77 may help) drives to recover your 35/40-track disks (can't do much recovery of 80-track disks on a 35/40 drive though). Note here that the double-density adapters, especially the newer ones have much better data separation and thus are more sensitive. Similarly the add-on data separators will help. If you don't have one, this may be the stimulus to get one, just in case it could help.

It may be a good time to tune up the system while you are at it. Clean the 40-pin connectors (a simple pencil eraser on both sides is pretty good). Do a memory test and disk diagnostic if you have one. Be sure to test for the correct speed of the drive. Slow the step-rate down even if the drive is rated fast. Keep the speed-up mods in slow speed too. Every little bit helps, and in this case, we need the reliability more than the enhancements.

Well, that is all I can think of. I would say here, patience and perseverance are probably your best utilities.

Case 1 and 3

There is a lot more that can be done when there are other sources to call upon for recovery of data from a damaged disk. That is not to say that there is a lot more to tell you, but there is a lot more work you can do to save the bacon ... if that is what you decide is really necessary.

There are different techniques that can be brought to bear on this case because there is more information available to work with. The problem is sometimes "too much" information to sift through, deciding which is the more valid!

The backups and printouts of the material to be recovered should be examined and compared to the offending sector(s). If you have files that have been modified but are similar, use the search features in SU to find similar locations in the bad sectors, then carefully repair them. It may take some time, and there is not as much chance of success, but you have a lot of tools to use.

If large sections of the file are missing, you can use the sector read commands to get similar sectors from backups into memory, and then block move commands to repair things. I have a thing against typing in code anywhere it can be avoided. That is because I am such a sloppy and error-prone person that I hate to give Murphy such a good shot at me. So, I recommend using the sector and memory manipulating commands whenever the data wanted is in a machine readable (disk files, cassette, memory etc) form.

The same situation exists here as before. We are really just using all the earlier skills of recovery with enhancements due to our added informational resource, i.e. backups and printouts.

Let's again start out simple. Suppose that you have some printed material that may help. Listings, printouts, source code, "JKLs" and any other possibly useful printed matter. Any time in the earlier methods that you ran out of recovery material, you could refer to the print-out and see if the needed data was available. If so, just use SU to type it in. If there is a lot to type in, you better be careful, and have at least one friend check your work. This is most important in the machine code files, and, as we have seen, least in the ASCII files.

Examine the material closely for clues to the necessity of recovery, importance of certain sectors, location of critical data or formats of such. Analyse and use every clue given there. Not much more can be said here specifically about printed matter. Refer to it when the need is there.

Backups, Files and Disks

Take the situation that probably will arise most often. You have the backups, but don't know if they cover everything. In fact, they probably don't, because changes and updates surely have been made since the last backup. What to do?

Don't panic. SU to the rescue! If the backups were just updated slightly or the files were just extended and not modified in the middle, then we can do the repair relatively easily. See the section of this book called Problem Recipe. I think that will help.

If the backups are old, or extensive differences have been made, we need to do some serious transplant surgery and heavy grafting. It is not likely for the sectors to be transportable from backup to bad disk (copy). On the lighter side, the differences are not likely to have changed the same data that was destroyed unless that was very extensive (over 30% of the material to be recovered was destroyed) or Murphy is in a really rotten mood just because you bought SU and this book. He might take that as a personal challenge.

So, if we consider that the backup material is mostly usable, but not in the right place, i.e. conveniently located on the proper sector boundaries, we must do a little work. Not much, as SU has a way. Remember when I said we could read tracks, do searches for data, block moves and sector writes? Well maybe you were sick that lesson. The idea is to first find the data.

Use the FILES utilities to examine the sectors in the backup files. If you have any trouble locating the specific spots, look at the sectors just before the damage and try to pick-out some sequences to search for. If you have a printer, print-out the last good sector and then use the string search or word search on the backup disks to zero in on the data. Remember, it will probably not come on sector boundaries, so be alert. After each "find" check the rest of the data near-by to be sure this is the right place. As a help, it should not be too many sectors (relative) different from the beginning of the file than the bad sector. It should be more carefully checked if it is machine code than if it's ASCII files, of course. In fact, it may be a good idea to do some fancy foot work here. The trick is to take all the data you have, synthesize it into a good replacement for the bad sector. To be on the safe side we will construct the sector BEFORE the bad sector as a test of our ability to create a good sector from "raw" data.

To do this, assume we have found the area of the "backup" that is the material that comes just before the code we are really looking for. Now load that material into memory. That is, load two sectors of the backup that come before the material we really need, and make a note of where it is. Load in about 5 sectors, so we are sure to bracket the desired material, and it is all there waiting for us in memory.

Use the memory utilities to shuffle around the good data area until you think you have it set up as one prior sector (256 bytes) and then the recoverable sector(s) next. We can do this by going to the last good sector on the "copy" disk, noting how it looks at the first of the sector, and then find the location in memory of the backup material we just loaded in.

Do a block memory move, starting at the point that is the same as the last good sector we just located. Move about 1000H just to get everything and move it to the boundary (the default first memory location that SU loads to). That means we should have the "good" sector as the first sector in memory, and the rest in exact order after it.

Now you do a memory compare with the sector before the bad one. That is, the last good sector on the copy disk, and the new sector made up in memory. A good compare means you surely have it right and you can write the next sector(s) out to the repair (copy) disk.

If it does not compare, don't despair. Examine the differences for the possible updated material that makes this file different. It may just be a slight change or update. The more you know about the file in question, the better you can make the decision to use it or not. I'd try it to see if it works before I gave it up.

Be sure to use as much information as you can get when deciding. If you can, check with the author, users, printouts and listings, anything that might tip the balance of recognition and understanding.

There may be several backups, printouts and other information that is slightly conflicting. Try to analyse the thing as a whole to see if you can determine which pieces are the real ones and slowly build up the sectors bit by bit (byte by byte?). Once the information is in the memory, you can do a sector write to the copy disk's "bad sector." Be very careful of the sector boundaries, they are the worst cause of a one-byte error. Sectors and memory both start at zero, not one, etc.

Like I said, there is a lot more to do, but it is really just the same things I've mentioned, expanded as far as you can.

DIRECTORY AND DOS RECOVERY

There is a whole section of possibilities I have neglected. If there are errors in the directory, they can make the whole disk unreadable and seemingly unrecoverable. You might need the disk whole, as in the case of a new, fully zapped, updated DOS disk that got an extra unwanted zap in the mail.

Let's cover directories first. They are easier, and more important, than almost anything else. There are some short-cuts that can be used on directory errors, and a host of Super Utilities (I was wondering when I could work that in) to repair the directory, at least to the point of being able to save the important files. Remember that it is not always necessary to get the directory back to a "no error" condition. Just get it well enough to read and copy files to another disk! This is even true if you are trying to recover the full disk, DOS and all (at least on the more standard ones, I can't be sure these techniques can save TRS-80 CP/M, but ...).

Again, starting on the easier stuff first, let's examine the repair features of SU. The especially easy errors to fix are the bad GAT and HIT tables. These are the first and second sector of the directory. As mentioned, these are made up of information duplicated in the rest of the directory and are used to speed up disk access.

The repair GAT and HIT sector utilities are pretty-good and clearly explained in the manual. The only thing I might add is a few hints on special situations. For instance, if the GAT or HIT sectors have CRC errors in the header, the sectors will not be readable at all. In this case, I would copy over a bogus sector from a data

disk or system disk that was configured exactly the same. I can then "repair" that sector with the repair commands. This may mess up the password, disk name and date and a few other trivial items, but they could be restored later with regular backups, ATTRIB, change name, and change parameters utilities.

It is especially important to have SU configured correctly when doing repair on the directory. It is equally important to copy over these sectors from very similar disks (one of the backups is ideal). It may even be necessary to copy over all or a part of a hit sector just to get the required information where it belongs. For instance, when SU repairs the HIT of TRSDOS 1.3 and 2.7DD (on Model III and I respectively) it leaves certain bytes alone (see manual on HIT REPAIR). In this case, if this area is damaged or the whole sector is unreadable, use COPY SECTOR DATA or COPY SECTOR to get the necessary sector data to the repair (copy) disk.

You can use the directory check feature of SU to check your progress. Now, if any sectors in the rest of directory are totally unreadable, or all the readable information has been saved, you could try using SU to clean up the bad sectors, or zero them out if nothing useful is left there. Then, use the SECTOR ALLOCATION and DISK ALLOCATION utilities to find any conflicts or overlapping files.

If one file is allocated to some sectors that are allocated to another file, either the allocation is wrong (an error in the directory), or one of the files has been partially overwritten. Try loading or testing each of the two files to get a clue. Also try stepping through each file with the DISPLAY FILE SECTORS. You might even note where on the disk the discontinuity occurs, examine that area with sector reads to determine if both of the files are still there (not really overwritten, the directory is in error). This is not too difficult if the files are continuous on the disk.

Of course, you can use all the previously described techniques to repair the directory too. What ever it takes. This, as I said, is the most necessary part of the disk to recover. It needs to be at least repaired enough to read it and find files. Then they can be copied to another disk and you are done.

DOS RECOVERY

If you MUST recover the system, There are a few things you can do to help. One important thing to remember is, again, you can usually get away without recovering the whole disk. That is because the disk may contain an enormous amount of FREE space, much redundancy and several very regular and duplicable structures.

To give an example, let's pretend we have a disk that has the latest updates of your favorite DOS, and got damaged in shipment. You would like to save as much of it as you can. I will assume that you have several copies of the older version that is very similar. We start on a "copy" of the defective disk again, getting as much as we can from the bad disk using everything we've learned. If the directory is bad we fix it if we can. Now, in this case, if we can't fix it, we can make another reasonable assumption, that most of the systems and even many of the utilities are at the same relative places on the disk. A few comparisons with sector reads and file reads (when possible) can help verify this.

If there is a sector (not HIT or GAT) in the directory that is not good, do the trick of finding where everything is on the disk (DISK ALLOCATION) and then do a directory check. These two things will give you a wealth of information to sift through. Look for sectors allocated but not assigned, Files not allocated to sectors, and the like. Compare this to the same thing on your most recent original version to see if things match or mismatch. Take your time, If it is worth doing its worth doing. Right?

Ok, We got the directory readable. Now what, Sherlock? Make yet another backup of your most recent original system disk and then copy over (not BACKUP) each file using SU's COPY FILES utility. It may be that not all the files can be copied. For these, you will have to recopy from the most recent original system disk those files that will not copy or do not work after they are copied. Oh yes, check them well, just because they copy in this case does not mean they are good. You could have got one with a bad sector, misallocated or other problems not normally seen.

Alternatively you could examine the files with the bad sectors and compare them to those on the original. Copy in the sectors from the original. They are unlikely to be much different from the updated version and most if not all the updates will still be there, only this one sector will be old.

Finally, you could use the compare sectors or compare files utilities to examine the differences in the most recent original system disk and those on your new recovery disk. Then note any differences and compare them to the notes you took on the damaged sectors and files from your previous work. You should have a good chance at telling if the final recovery is good.

If the differences are at sectors that had no problems, they are very likely the update to the system we are trying to save. Study those differences that fall on the "bad"sectors and review the procedures used to restore them. An educated guess and a lot of testing of the repaired system disk should get your confidence level up to actually using the disk for your needs. If you can't be absolutely sure about a "save" but need to use it, then mark the disk carefully and send off for the replacement. In the mean time, you can use the one you have, killing it when the real good one comes. This is a lot of work for a temporary save but I can imagine several instances where it would be necessary.

This kind of recovery can get to be very exotic and could require a quite knowledgeable person. But try it, it can only help. The disk is lost already, whatever you save and learn is your reward. That's about all I can do for you.

GOOD LUCK and remember, BACK IT UP!

Chapter V
Miscellaneous Stuff

When you boot in Super Utility, the first thing that happens is that the "title page" is displayed on the screen. Then, before the bulk of the program is loaded, a quick memory test is performed on your system. If a memory fault is detected, a message to that effect is displayed and the program load is aborted.

You may think it's a little snobbish of Super Utility to refuse to inhabit a computer that flunked its memory test--but the Surgeon General has determined that running Super Utility in a flaky system is hazardous to your disks. The memory test lasts only a few seconds, but on some Model I's, this may be long enough for the disk drive to time out. So don't be alarmed if your drive does a little extra clicking when you load Super Utility. It's all for a good cause. When the memory test has been passed, the rest of Super Utility is loaded. During this process, the Super Utility 3.0 logo will be on the screen. If the drive tends to slow or get stuck at any point, you know that your system has trouble reading that part of the disk. In that case, it may be time for some preventive maintenance--such as head cleaning, speed adjustment, and possibly even head alignment. Maybe, just maybe, you need a replacement Super Utility disk.

After Super Utility has been loaded, it performs one last disk access before quieting down. During that access, it loads the configuration and patch sectors.

Many computers utilize something called interrupts. TRS-80 disk systems are among those that do. This means that 40 times every second (30 on the Mod III - Ed.), the TRS-80 abandons what it's doing and executes special subroutines. These are called service routines. They help the system keep on top of situations which require frequent attention. For instance, the memory locations which keep track of time in a regular disk operating system (not Super Utility) are updated during service routines.

There are certain timing-sensitive operations, such as tape I/O, which would be thrown off by interrupt-interruptions. Certain stages of disk I/O also require freedom from interrupts. Interestingly enough, other stages of disk I/O can't take place without them. To accommodate such requirements, the Z-80 microprocessor has been given the ability to ignore (disable) interrupts or respond to (enable) interrupts, according to directives in its programming. During normal operations, interrupts are enabled a very high proportion of the time--what amounts to nearly constantly.

You have probably noticed that at nearly all times, there is a set of moving graphics characters at the edges of the display of SU. These are called "alive" characters. They are attended to by the interrupt routines and show the service routines are active (alive).

The alive character probably won't be of much practical use to you. Some DOS's (LDOS and VTOS) offer it as an option. If a program bombs and the system hangs, knowledge of whether the service routines are still running may help you diagnose the cause of the flake-out. An alive character gives you that information. However, it's very unlikely that you'll find yourself debugging Super Utility.

There is one way in which the alive character may be useful to you, though. It has to do with the fact that there are two ways of getting Super Utility to abort an operation. Usually <BREAK> will terminate any activity. However, there are a few instances in which only <CLEAR> will do the job.

Generally, <BREAK> works when interrupts are occurring (remember, that's nearly always), and <CLEAR> may be used when interrupts are disabled. So the motion (or lack of motion) of the alive character will let you know which of those two keys is appropriate.

An exception to this principle is that <CLEAR> is always used to abort a screen print and empty the print buffer. Also, you can try using <CLEAR> to terminate an activity when you don't want to be returned to the menu.

The cryptography mode may be entered from either Zap's sector-display mode, File Utilities' Display File Sector mode, or Memory Utilities' memory-display mode. Both the memory and sector display modes present a screenful of information--the former about a disk sector and the latter about a 256 byte memory block.

The screenful contains hex information on the left, and its ASCII equivalent on the right. Remember that these display modes each have two sub-modes--the paging mode and the modify mode. In the paging mode, there is a single cursor at the top left hand corner of the screen, in the "margin." You get into the modify mode by pressing <M> when in the paging mode. You'll know you're in the modify mode when you see the two cursors, one in hex area and one in the ASCII area, of the main display. To enter cryptography, you must be in the "paging", not the Modify mode.

To enter cryptography, press <@>. A 'DCR' (for DeCRypting) prompt will appear on the screen. Super Utility is now ready to manipulate the display for you. It does so by performing any of a number of logical or arithmetic operations on the display material. For example, you may ask Super Utility to add "1" to every byte in the display. The display will be updated accordingly. There is an important distinction to grasp here. Only the display is being altered. The actual memory or disk sector being displayed is not. You may perform a number of cryptological manipulations and then cancel the cryptological display. You will see that the target Memory-block or sector still contains the original data. There is one exception to this. That is what the new feature is all about. We shall get to it shortly.

Let's go through a sample cryptography session. Get into Memory Utilities' memory display routine. Use F000 hex as the target area. F000 is well above Super Utility's code and buffers, and any changes we make there should have no effect on the program's operation.

The first thing to do is to fill the memory block with contents which will make the effects of any shifts or rotations readily apparent. If the default data entry mode (as revealed in the upper left of the display) isn't already hex, make it hex by pressing <H>. Then press <M> to enter the modify mode. The cursor should be over the first byte of the sector. Type "55." The first byte in the displayed block should now be 55 hex, and the cursor will have advanced to the second byte. Use the left arrow to move the cursor back over the first byte. Press <P> for Propagate. Then type "FF." "FF" is hex for 256. All 256 bytes in the memory block should now be 55.

Press <ENTER> to leave the modify mode and return to the paging mode. Note that so far we have used only standard memory-modify procedures, not cryptography features.

Press <@> to get the 'DCR' prompt. The first thing to do is to tell Super Utility whether you want the results of your cryptography experiments to appear in the ASCII portion of the display only, or in the hex side as well. If you were looking for a hidden message, such as a copyright notice, you would only be interested in the ASCII. If you were looking for disguised machine code, you'd want to see the results of your alterations in hex as well. For the purpose of this demonstration, we'll select the latter mode. To do so, enter colon <:>.

Notice that each time you enter a cryptography command, the 'DCR' prompt vanishes and Super Utility is ready to receive normal paging commands--but the display is still in whatever cryptography mode was last selected. For instance, press <@> to get 'DCR' back. Now tell Super Utility to shift the display two bits to the right by entering "SR2" (for Shift Right 2).

As you see, the display changed in accord with your command, though the 'DCR' prompt vanished and Super Utility is ready to accept normal paging commands. From now on, all memory and sector-displays will be shifted two to the right, until you cancel this condition. As a reminder, the letters SR2 will remain near the bottom left hand portion of the screen.

You may override the shifted display by entering different cryptography command. Or you may cancel all cryptography and make the screen revert to normal by keying <@> and pressing <ENTER> without pressing any other keys.

You may use parallel syntax to shift or rotate right or left anywhere from zero to seven bits. To AND every display byte with a value, get the 'DCR' prompt and then enter <A> (for "AND") followed by the value. For example, you could enter "A1" to AND the display with one. Use the same syntax for OR and Exclusive OR using "O" or "X" instead of "A." To add or subtract a one byte value from every byte on the display, use a "+" or "-" followed by the value to be added or subtracted. For instance, "+5" would add five to every byte in the display.

There is also an automatic, or movie, mode. This only works for all DCR modes. Answer 'DCR' with an up-arrow for automatic increment or a down-arrow for automatic decrement. You may follow the arrow with a number between 0 and 255. This determines a timing factor which controls the display change rate. Try it, it's fun!

You can use cryptography for encrypting as well as decrypting. For instance, suppose you want to create a sector on your disk containing your name, address, a copyright statement, and perhaps some other information--all hidden. First find an unused sector--you can use File Utilities' Disk Allocations options to do so. Then use Zap's Display Sectors to display it.

Go into the modify mode and zero the display. An easy way to do that is to use the <P> command to propagate a series of zeroes through the sector. Then type <SHIFT><ENTER> followed by <A> to change the input mode to ASCII. Type your message into the sector display. When you are done, press <ENTER>. Answer the next

prompt by entering <U>. This will save what you've accomplished so far to disk (in non-encrypted form) and return you to the paging mode.

Now you may start encrypting. Suppose you decide to hide the message by adding 5 to every byte. Press <@>. Then answer the 'DCR' prompt by entering "+5." If only the ASCII side of the screen changed, select the full screen update by getting the 'DCR' prompt and entering <:>.

All that remains now is to save the sector to disk in this disguised format. But remember, the cryptography features discussed so far haven't actually changed the sector-only the display. In other words, even if you were to resave the sector, it would still contain the same data.

In order to make the change "real," use the <@> key to get the DCR prompt. Then enter <!> (Exclamation mark). The "!" is the previously undocumented feature. It will cause the actual sector buffer to be altered so as to conform to the display. Saving the sector now will result in your message being stored on the disk in the encrypted form. If, instead of the sector-display mode, you had been in the memory-display mode, the encrypted material would have been transferred to memory as soon as you entered the <!>.

Some of you may have wondered what software (other than Super Utility itself) Kim uses during his hours at the keyboard--what are the software tools of his trade? I, at least have been curious about this, so I asked him. Kim uses the LDOS disk operating system, the EDAS editor/assembler, and Jake Commander's Macromon (Shadow) monitor.

Another thing you may have wondered about is the unusual disk addressing scheme used by double density NEWDOS-80 and DBLDOS. Why are the "relative tracks" different from the physical ones? This system is called Disk Relative Sectors or DRS. There is a bit of a paradox involved here. This apparent complication really involves an attempt to make things simpler and more transparent. It stems from the fact that single density is divided into two grans of five sectors each, but double density tracks contain three six-sector grans. DRS lets you pretend that everything's always the same. In other words, you can act as if double density disks still consisted of ten sectors per track.

As an example, let's consider a double density NEWDOS-80 disk with a double density track zero. In reality, every track contains 18 sectors numbered zero through 17. I will call such a real track, which consists of 18 sectors, a physical track or p-track.

NEWDOS-80 will take the first ten sectors (0 through 9) of the first p-track zero and call them track zero. I'll call them relative track (or r-track) zero. That leaves eight sectors (10 through 17) in p-track zero out in the cold. So NEWDOS takes those eight sectors and groups them with the first two sectors on the next real track, and calls this 10 sector aggregation track two (that's r-track two to you).

Notice that the first r-track in our example occupies part of one p-track. The second r-track spans a real track boundary and occupies parts of two separate real tracks! Here's a map of how the first several p-tracks would be divided up into r-tracks. All numbers are in decimal:

Physical Track	Sector	Relative Track
0	0- 9	0
0	10-17	1
1	0- 1	
1	2-11	2
1	12-17	3
2	0- 3	
2	4-13	4
2	14-17	5
3	0- 5	
3	6-15	6
3	16-17	7
4	0- 7	
4	8-17	8
5	0- 9	9
5	10-17	10
6	0- 1	
6	2-11	11
6	12-17	12
7	0- 3	
7	4-13	13
7	14-17	14
8	0- 5	
8	6-15	15
8	16-17	16
9	0- 7	
9	8-17	17

Now suppose that to keep up appearances of normality, the directory is placed on r-track 17. It's actually on p-track nine. But NEWDOS-80 doesn't want us to be concerned with that. The idea is that when we want NEWDOS-80 to read a certain part of the disk, we tell it where to look in r-tracks and r-sectors. NEWDOS-80 will do all the conversions internally, and look at the correct real track.If you have a double density disk with a single density track zero, NEWDOS-80 starts relative track zero at real track one, sector zero. So a 35 track diskette will have 36 actual tracks, since NEWDOS-80 doesn't count the single density track.

If all that's not complicated enough, NEWDOS-80 also lets us vary the number of sectors per gran (or lump, as NEWDOS-80 calls them) instead of sticking to the conventional five.

And if you have a double density system, it also allows the following combinations of formats:

Method	Track zero	All other tracks
1	single density	single density
2	single density	double density
3	double density	single density
4	double density	double density

As you can see, NEWDOS-80 allows all possible combinations. But Super Utility DOES NOT support them all. If you want all of Super Utility's features to work on your NEWDOS-80 disks, you must use standard configurations. Method (3) is not supported. Also, you will have to use grans which consist of five sectors.

For operations which are not file oriented, or don't in any way involve the directory, this restriction isn't too important. For such operations, configure Super Utility as if you were using another DOS. If the disk has format method (1), use "TS" (Model I TRSDOS). If you used method (2), use "D1D" (double density Model I DOSPLUS) or "L1D" (Model I LDOS with SOLE track). If you used method (4), use "D3D" (double density Model III DOSPLUS) or "L3D" (Double-density LDOS). By the way, if you use format (2), NEWDOS-80 does another trick. It doesn't use or count physical track zero at all. When you boot the disk, p-track zero, sector zero will be read loaded into memory. But this is done by the ROM's bootstrap routine, not NEWDOS-80. As far as NEWDOS-80 is concerned, p-track zero doesn't exist. It starts r-track zero, sector zero, at p-track one. It even maintains a duplicate boot sector at p-track one, sector zero. Use "NDR" for this one.

Remember, in the new version of Super Utility, you can no longer prevent the system from updating the configuration table and it may not remain the way you left it. Therefore, you may have configured Super Utility with a correct track-count, but Super Utility may no longer be using it. Therefore, it's a good habit to use DOS specifiers and track-count overrides when working with TRSDOS III and NEWDOS-80 disks.

As an example, suppose you want to check the directory of a 40 track Model III TRSDOS disk. The disk is in Drive 0. When Check Directory asks you to enter the Drive(s), instead of entering "0," enter "0TD=40." TD is the DOS specifier for TRSDOS III and 40 is the real track-count. Now, regardless of what is in the configuration table, Super Utility will be able to handle things properly.

Here's a another warning about TRSDOS 1.3 (Model III) and TRSDOS 2.7DD (Model I). System files are not logged into the directory. This has some unpleasant implications. For instance, suppose Zap's Verify Sectors routine reports a bad sector on a TRSDOS disk. You then use File Utilities' Sector Allocation program to see if that sector is allocated. It's possible that Super Utility will report the sector as unallocated, even though in reality the sector contains a system file. You might then Format Without Erase the disk, selecting the S>kip option whenever Super Utility balked at the bad sector. If you did so, you would probably end up with a disk which wasn't functional. One preventive measure would be to look at all the system files on one of your TRSDOS double density disks. Make a note of all the sectors they occupy. The system files should be located in the same relative positions on any other TRSDOS disks you have, especially if they're all descended from the same master disk. Keep the list handy. When Super Utility tells you a sector is unallocated, cross check it with your list of system sectors to be sure.

Another peculiarity of Model III TRSDOS is the way it treats passwords which encode to zero. It changes them to something else. A result of this is that when you use File Utilities' Compute Passwords, the password it gives you might not work. This will happen when the password happens to be one which encodes as zero.

If you should get a password that doesn't work, just use Purge's Remove All Passwords utility. Or use Zap to change the individual file's password in the directory. Start by looking at the disk's directory sectors until you find the entry for the target file. Remember, each directory entry occupies two lines of Zap's display. The encoded hash value should be the first two bytes of the second line of the file's entry. If Super Utility gave you a password which didn't work, you should find that those two bytes are zeroes. Use the modify mode to change either or both of them to anything else. Then go back and rerun Compute Passwords. The password given this time will work.

While we're on the subject of Super Utility and passwords, here's something else you should know. Figure 11 reproduces the input and output of a typical session with File Utilities' Compute Password.

```
E>ncode or D>ecode ? d.
Filename ? su6a/l.
Reading       Drive 1,  Track  40,  Sector  17
SU6A/L
Access    =   FB2DH
OM        =   FB2DH
Update    =   E32FH
MU        =   E32FH
<KEY> to continue:_
```

Figure 11

As you can see, both the update and access passwords are decoded for us. The format of the output is as follows: first the filename is displayed in uppercase. Below that is the word "Access" followed by some spaces and an equal sign. To the right of the "=" is the encoded access password in Hex. On the next line is the decoded access password, followed by some spaces, and equal sign, and a repetition of the encoded access password. The format is repeated for the update password. The decoded update password is displayed once, while the encoded update password is displayed twice. This is all fairly clear. But now suppose we run Compute Password on a file which doesn't have any passwords. The display would look something like the one shown in figure 12. The passwords have been displayed as a series of blank spaces--which is what they actually are. 9642H is the way a blank password encodes in the directory. But if you haven't used Compute Password before, the display can be a little confusing. You might think that Super Utility is trying to tell you that 9642H is the decoded password. So, if Compute Password ever gives you an answer that looks like the one in figure 12, remember--it means the passwords are blank.

```
E>ncode or D>ecode  ?D.
Filename ? su6b/l.
Reading      Drive  1,  Track  20,  Sector  17
SUB6/L
Access   =   9642H
         =   9642H
Update   =   9642H
         =   9642H
<KEY> to continue:_
```

Figure 12

Furthermore, the algorithm which the password routines in Super Utility use will depend on the type of the disk last accessed. This means that if the last disk accessed was TRSDOS 1.3, then the TRSDOS 1.3 algorithm will be used. If it was a TRSDOS 2.7DD then the TRSDOS 2.7DD algorithm will be used.

The new version of Super Utility now includes full support for the Radio Shack double density board for the Model I. Super Utility is now able to recognize automatically which doubler is installed in the computer and use it properly. Also, full support for Radio Shack's double-density system for the Model I, TRSDOS 2.7D, is built in. While you may configure Super Utility to recognize the Radio Shack doubler ("Doubler=R") or someone else's ("Doubler=X") it is not really necessary as Super Utility will determine the type of equipment it has to deal with, and adjust accordingly.

Judging by the calls and letters Breeze/QSD gets, there is a certain mistake which people make rather frequently--one that can completely crash the program. What they do is overwrite part of Super Utility with other data. A common time for this to happen will be during the use of Memory Utilities--especially Sectors to Memory and Track to Memory. When asked to specify a buffer, they will choose one too low--one right in the middle of Super Utility itself. When the track or sectors are read in, they overwrite a vital part of Super Utility. The crash may materialize immediately, or it may happen later on when a different program module is used. The best way to avoid this is to just press <ENTER> and default when Super Utility asks you to specify a buffer location. Super Utility will then automatically choose a safe area of memory and inform you of its location. Remember, when in doubt, default!

If you need to know the safe memory areas that can be used while using SU, it is from the default address to the end of memory (FFFFH). It can easily be found by doing a fake read and pressing <ENTER> to get the default. That's the lowest safe memory you can use.

EXTENDING TRACKS

Extending a disk means adding tracks to it without erasing the data already on the disk. For instance, you can extend a 35 track diskette to 40 tracks. Elsewhere in this book, I described a very awkward method of extending disks using Format without Erase. Here's the easy way to do it.

Go into Super Utility's Disk Format section and select Standard Format. Answer the "Name," "Date," and "Password" prompts as you please. Answer the "Use Configuration ?" prompt by entering <N>. This will result in your being prompted with "DOS Type, Tks, Dir, St Tk ?"

You are being prompted to enter the DOS specifier, the track-count, the directory track number, and the starting track. Suppose you have a 35 track Model I TRSDOS disk on drive one which you want to extend to 40 tracks. Its directory is on track 17. Follow the instructions in the preceding paragraph. When Format gives you the "DOS Type, Tks, Dir, St Tk ?" prompt you will enter "TS,40,17,35."

"TS" is the DOS specifier for Model I TRSDOS. "40" is the number of tracks you want the disk to end up with. "17" is the number of the directory track. The disk, to begin with, has 35 tracks. Since the first track is numbered zero, the highest one is number 34. So you want to start the extension process to start with track 35. After adding the tracks to the disk, Super Utility will ask you if it should rewrite the directory and boot. It is very important that you answer "no," since an affirmative answer would result in the current directory's being wiped out.

After the format is completed, finish the process by going to the Disk Repair module and running Repair GAT. Use a track-count override equal to the new number of tracks on the disk. In our current example, when Repair GAT asks 'Drive # ?', you'd respond '1TS=40' (This procedure will give you a 40-track TRSDOS disk, but does not mean that TRSDOS will automatically recognize the extra tracks. TRSDOS for the Model I can be extremely stubborn about refusing to recognize that it has been handed a free gift --Ed.)

Earlier, I referred to the danger of using the double-step configuration for writing. To review, the double-step mode lets you use an 80 track drive (96 Tracks Per Inch) with diskettes formatted in 35 or 40 track drives (48 TPI). In such circumstances, it is safe to read. But if you try to write in the double-step mode, the written track will not be as wide as a true 48 TPI track.

The write operation may seem to be successful at first. The trouble usually comes when and if the target diskette is placed back in a 48 TPI drive. The data which was written in the 96 TPI drive may no longer be legible. Because of this, it is recommended that you software write protect any drives that are configured to double-step.

In my own experience, I have gotten away with writing to disks while in the double-step mode. I have even been able to read those disks in my 35 track drive. However, there was one precaution I observed which probably contributed to my success.

The only disks I wrote to in the double-step mode were disks which were freshly bulk erased. Here's a recap of my formula. To begin with, I have one 35 track 48 TPI drive and two 80 track 96 TPI drives. Occasionally, I've needed to back up a 40 track 48 TPI disk. In such a backup operation, my 35 track drive can't handle the innermost five tracks of either the source or destination disk. So both disks had to be placed in my 80 trackers, both configured to double-step.

Before starting the backup, I bulk-erased the destination diskette. Then I performed the backup. The copy I ended up with seemed to work with 48 TPI drives. The bulk erase evidently ensured that there wouldn't be conflicting data streams under the wider 48 TPI read/write head.

The bulk eraser I used was Super Utility's Software Bulk Erase routine (in the Format menu), configured the drive for 80 tracks without double-step. Since this procedure worked for me, it may work for you. But don't count on it! Test the process thoroughly before entrusting valuable data to it.

If you use a product called the Patch, you may have some difficulty using Super Utility. the Patch is generally incompatible with any program which uses a debounce delay in its keyboard scan. If you have installed the Patch, you may have to install a patch that disables Super Utility's debounce-delay.

I once read something in Infoworld that stated that no disk-software protection scheme could work because whatever is read under head A can be simultaneously written under head B. By now you probably realize that with the TRS-80's FDC, this just isn't so. Thus, no program can automatically copy every protected disk. Human intervention may be needed to decipher the formatting tricks used and to set up a similar format on the destination disk. Super utility does an excellent job of attending to this chore automatically.

When it first came out, Super Utility's Special Backup could copy almost any software on the market. More convoluted protection schemes have evolved. Owning Super Utility no longer guarantees that you'll be able to backup any disk you buy.

Kim does not intend to improve Special Backup. Nowadays, most software venders have rational backup and replacement policies, similar to Kim's own. Such policies allow the user reasonable safety and convenience with regard to backups, and spare the programmers and vendors the gargantuan expense incurred by leaving themselves vulnerable to software pirates.

UNDOCUMENTED FEATURES IN SUPER UTILITY

This section is meant to give you additional capabilities and flexibility by describing several undocumented features, with comments on them and Super Utility in general.

The first thing I'll talk about is two undocumented DOS specifiers that expand the flexibility of Super Utility.

The U DOS Specifier

The U DOS specifier stands for Unknown DOS, which can be used if the DOS on the disk to be examined is not known for sure or it is not a "normal" dos. Perhaps not a DOS at all, just a specialized disk with its own loader like Super Utility.

It might be used in conjunction with the # sign (see chapter two in the users manual) to determine the density and then the material in chapter III of INSIDE SUPER UTILITY 3.0 which relates to identifying unknown disks.

As with the other DOS specifiers, Unknown can be entered in several equivalent forms. Here are a list of those legal forms:

U U1 UD U3 US

These specifiers allow the user to specify an opposite density first track, single or double density, and will work with any number of tracks. It will also work with the double step feature which allows 48 TPI drives to be read by 96 TPI drives (see the "equals" discussion in the Super Utility users manual, chapter one). Or you can just use U and see what happens.

A word of caution is required here. The new automatic features for DOS and density recognition may cause some confusion at first, because the configuration table will get updated and modified with out your being aware of it quite often. This is especially problematic when doing this unknown disk work. You set a parameter and try a few things and then start getting errors on things that you know should work, or just did work a few moments before. Suspect (and inspect) the configuration table. Many times the density gets changed and thereafter nothing is readable.

Get in the habit of adding the DOS specifer right in the command. For instance when ZAP/DISPLAY SECTORS asks for the "Drive, Track, Sector", you put 0US instead of <ENTER>, and force Unknown, single density. The zero (drive number) is also required.

This specifier makes the sectors per track and track count infinite in size so that all sectors on a diskette will be found, even if they have very large sector numbers. For instance, if you have a disk that was made for a specific purpose and has its own special formatting, it may have sectors with numbers that are not in sequence or there may be more of them than usual.

Some versions of CP/M on the TRS-80 Model 1 would have 18 sectors of 128 bytes each on some of the tracks (usually on the main tracks, the "system" tracks having 256 in the sector). No standard DOS specifier could correctly be used on it. U is very handy with such disks.

Actually U's sector count is not infinite, it is just the largest allowed, 255. Remember, any time you get a sector not found message you can just press "S" for Skip and Super Utility will go on, or display a bogus sector display, allowing you to use the normal commands (such as arrows) from there. You should use greater than or less than, ">" or "<", for such disks.

Another point to remember is that you must change the tracks manually with the up/down arrows. Also the side of the disk that is being examined is not changed automatically in this mode, though it is displayed as usual in the lower left corner along with the density information. Further, this mode will read any size sector into the sector buffer but if the sector buffer is larger than 256, you must read the sectors one at a time. This is because Super Utility increments the address in the buffer for the next sector for a 256 buffer length (standard on nearly all TRS-80 software, CP/M not with standing). Thus the sector that is, say 512 bytes will take up the next 512 bytes of the buffer, but reading another sector will overwrite the last 256 bytes of this previously read sector in the buffer. So just look at them one at a time.

If you must have them in memory all at the same time and one after the other (contiguous), then use the memory commands to block move them to another location.

The U specifier can be used effectively with the "#" (see chapter 2, ZAP "1"). But care must be taken to keep control. Don't let these automatic features railroad you into unusable configurations.

Super Utility can automatically set the Unknown specifier if given the chance also. For instance when using the "!" command in ZAP,"1". This happens when Super Utility can't make up its mind about the DOS on the disk.

While I'm at it, from the configuration table (main menu, number 9) the display for the U DOS specifier is U** even if you enter UD or U1S etc. This is also true of the configuration display when it gets changed from ZAP "1" or any where else using the optional DOS specifier when asked for the drive number. And of course it is also true when Super Utility updates this automatically with the "#" option.

The X DOS Specifier

This DOS specifier is available only on the MAX-80 version of Super Utility. It is for the 8" drives supported on the MAX-80 computer from LOBO.

Similar to the previous specifiers, this one can be entered in several forms:

X XS XD

Of course, a MAX-80 and an 8" drive must be used. The S, as usual, stands for single density. The D for double density. The format which this DOS specifier recognizes is the one created by LDOS on 8" disks. The X specifier obviously cannot be used to read CP/M disks, as some MAX-80 owners would like to think, for the same reasons given above. It MAY work on other formats (e.g., DOSPLUS), but it's NOT designed to. So if you want to use it to read an 8" disk created by a system other than LDOS, you're on your own.

Because of the different hardware architecture of the MAX-80, a separate version of Super Utility must be used for this computer. Don't try to use this version on a regular TRS-80, because it won't boot up at all.

MOST ASKED QUESTIONS AND LEAST PAINFUL ANSWERS

"What is the difference between the DOS specifier T1 and TS?"

This question points up a common misunderstanding of the DOS specifiers. There is a table in the manual (and an explanation in this book) that explains the relationship between the various specifiers. I will take a moment here to try to simplify and wrap up this material. The manual has a DOS specifier table for both the Model I and III and for either single- or double-density (where applicable). Each DOS supported is represented (see the undocumented specifiers section in this book).

Now each specifier under Model I (or under Model III) for the density and DOS stated, are equivalent. Some are just short-hand for the full three character specifier. If only one or two characters are entered, the defaults for the rest are assumed.

For instance T1 used as a DOS specifier means TRSDOS (T) Model I (1) single-density (assumed or default). Similarly, TS is TRSDOS (T) Model I (assumed or default) single-density (S).

Thus, all the forms shown in the table are equivalent. T = T1 = TS = T1S. Some will use the defaults and one is fully defined (T1S).

One last point of possible confusion is the two columns shown in the table for Model I and Model III. They show the defaults that would be used for that computer. This means that in the above example, only T1 and TS are valid for the Model I. If you are working on a Model III, then use the second column (note that these specifiers are INVALID on the Model III).

As a final example, under LDOS, if you type in the specifier L only, it will mean L1S if you are using a Model I machine and L3S if you are on a Model III.

The 3 character specifier should present little problem, as long as you limit yourself to the column of the table that pertains to your machine.

"How do I chose the DOS specifier?"

There are three main things you need to know to chose the correct specifier. The specifier can be divided into the three characters that are usually entered. Refer to the DOS specifier table in the manual or the specifier explanation in this book for more detail, but here is a summary.

The first character is the specific DOS that you are using. T for TRSDOS, L for LDOS, D for DOSPLUS, M for MULTIDOS, N for NEWDOS80 VER.2 and B for DBLDOS. Usually you just specify the correct character for the DOS that made the disk.

Next, the density of the first track is specified by the number 1 for Model I and 3 for Model III. Since the Model I is single-density until forced otherwise by a doubler, the first track is single-density.

Next, the density of the rest of the disk is specified by the letter S (for single) or D for double. If the disk is double density ("D") and it is either NEWDOS/80 version 2, DBLDOS, or Model I MultiDOS "P-density" you will want to add the letter "R" to indicate to Super Utility that the disk is formatted using the DRS (Disk Relative Sector) scheme (e.g., "N3DR", "M1DR", "NDR", etc).

Finally you tell Super Utility whether your disk is single sided by appending an apostrophe ('), or double density by appending a double quote ("), to the specifier, for example L3D", or N3D'. Super Utility normally assumes single-sided diskettes, so you could skip this step if single-sided diskettes were all you had. You must enter a quote, however, if your diskettes are double-sided, or Super Utility could easily wreck your directory (by turning it into a single sided directory -- that is, locking out the entire back side of the diskette).

"If I have a Model III do I set the Doubler=N ?"

No. Leave the Doubler=R or =X because it is a double-density machine. In fact, the only time it should be =N is on a Model I that does not have a doubler at all. At all other times leave it Doubler=R or X.

"To back up my ND80 disk that is double-density, except track zero, what should the specifier be"?

Make it N1D. Do not use the N1DR specifier in this case because that tells SU to ignore the single-density first track. It does format the disk with single-density track zero but does not copy the data from the source track to the new disk.

"Why can't I read a single-density Model I disk on my Model III?"

Actually you can, there is only a little problem. The Model III uses a 1793 disk controller, where the Model I uses the 1771. Under most conditions they are compatible. The only problem comes in the special header information put on the disk during formatting. Model I TRSDOS (which uses a 1771 disk controller), uses two of the 4 available data address marks (DAMs) to differentiate between a directory and the rest of the data on a disk. As it happens, the Model III controller (1793) can only differentiate between 2 of the 4 and it is not the two needed to read Model I diskettes on the Model III while using the standard TRSDOS recognition procedures.

The simple solution is to use the READ-PROTECT directory feature to convert the Model I DAMs to those readable by the Model III. Remember though, if you're going to want to read the disk on a Model I again, it will have to be repaired on that machine (the Model I) before it is usable again.

This only applies to reading the directories and file handling, including backups. All other SU functions will work on these disks.

"During the display of a sector, I am confused as to what SU shows me on the screen in ASCII, especially for the control and graphics codes."

On the video screen for the Model I, the first 1FH characters (31 dec) are displayed as "@" for 00, A for 01, B for 02, etc. That is the control code is printed by just adding 40H (64 dec) to the code value and printing it on the screen.

For example, the famous 0D carriage return is printed on the video screen as an upper case M.

The characters displayed for ASCII values above 7FH (127 dec) are the graphics characters themselves for the first 64, and then a repeat for the last 64 characters (which are the space compression codes according to the BASIC manual). So character BFH (191 dec) looks the same as FFH (255 dec); a full, solid graphics character block.

By the way, there are many differences in the displays, due to the many different character generators used in TRS80s, and the variety of upper/lower case modifications done. Don't worry too much about it, but if you want to see your "character set" RUN this in BASIC:

```
10 FOR A = 32 TO 255:PRINT A;CHR$(A),:NEXT
```

A last word regarding what characters look like. When printing, these characters are slightly different from the screen. First, the control codes are always replaced with periods. Second, if your printer is not capable of printing graphics, SU sends "#" characters for all ASCII values above 7FH (127 dec). Third, if you tell SU, via the configuration table, that the printer can support TRS-80 graphics, it sends the exact character value. Last, if you use the M option for MX80 type printers that use the "offset value," SU sends the correct offset to print the TRS-80 graphics correctly on the MX80 in its extended capability mode.

"Does SU support double sided operation under MULTIDOS?"

On the newer version of MULTIDOS, the system considers both sides of the disk as one "volume" of information, with one directory, and each track as having a front and a back. SU supports this version. However, there is an earlier version that treats both sides of the disk as different drives and SU can not be counted on to perform reliably when doing directory, file or file copy commands on this version. The other utilities should work fine on either side of these disks.

"I try to use the D command in ZAP utilities to get to my directory track, but it doesn't work!"

There are two things to keep in mind here. One is to make sure you separate the D from the drive number by a comma or space, so that Super Utility will not take it as an override DOS specifier. For example, if you specify "0D," you will change the conifugation of drive 0 from whatever it was to a DOSPLUS disk! The correct form of the command is "0,D" or "0 D".

The other thing to keep in mind is that when the D is used to go to the directory track, it will go to the directory track as defined in the configuration table for that drive. If you swapped a disk, and Super Utility had not had a chance to figure out where the directory was, this form of the command could take you to a data track. To make sure you go to the directory track, make sure the data in the configuration tables are correct. Or let Super Utility figure out the directory location for you by using any of the procedures which access the disk directory.

Chapter VI
ZAP! You're Dead!
or, How Not to Destroy a Disk and Not Know It

Zap is probably Super Utility's most overwhelming module. It has the largest submenu and the greatest number of non-menu commands. It's also one of the easiest modules with which to cause irreparable damage, if you don't understand what you're doing. This chapter will put you and Zap through some paces together. If you follow the examples presented here, you'll explore all of Zap's features in a safe, step-by-step, fashion.

Let's start by creating a target disk (or maybe I should say a victim disk). Boot up Super Utility. Then remove it from drive zero and replace it with an unformatted disk, or a scrap disk. Configure the drive for Model I TRSDOS. In case you were absent the day I covered configuration, that means you have to go to Super Utility's configuration module, and enter the "T1S" DOS specifier for drive zero. Then go to Super Utility's Format program and format the disk in drive zero. Answer the 'Use configuration?' prompt with <Y><ENTER>. This will put a standard Model I TRSDOS format on the disk.

After you've formatted the disk, press <SHIFT><BREAK> to get the master menu. Then press <ENTER> to get the Zap menu. Look at the boot sector of the disk you've just formatted: press <ENTER> to select the Display Sectors option. Super Utility will prompt you with 'Drive,Track,Sector ?" Press <ENTER> again to default to 0,0,0. Super Utility should now read the disk's boot sector.

On the ASCII side of the display, you'll see the message, 'NOT A SYSTEM DISK' towards the bottom of the screen. This message would be displayed if you tried to boot the target disk. If you want to verify this, put the disk in drive zero and press reset. After you've seen the message, reboot Super Utility and get it to display the target disk's boot sector again. If you don't have ready access to a Model I, repeat the formatting process, but with an "LD" DOS specifier, instead of "TS." Use Zap to observe the 'NO SYSTEM' message and then try to boot the disk on a Model III. When you've satisfied your curiosity, Please reformat the disk using the "TS" DOS specifier, so that you'll be able to follow the rest of this session.

Suppose we want to change the "NOT A SYSTEM DISK" message to something a little more dignified, like 'BOOT YOURSELF MAC'. The first thing to do is prepare for ASCII input. If you haven't changed your Super Utility defaults, you should currently be in the hex input-mode. You can confirm this by noting the word "HEX" at the top of the display's lefthand column. You should also be in the Paging mode. This is indicated by the single blinking cursor above the word "hex." If you were in the modify mode, you'd see two cursors in different parts of the screen.

Change from the hex to the ASCII mode simply by pressing <A>. The word "HEX" should be replaced by "ASC" to indicate the new status. You could just as easily have entered the decimal, binary, or octal mode, by keying <D>, <B>, or <O>. Verify this by keying <D> and watching the input-mode designator change to 'DEC'.

Leave decimal as the input mode, for the moment, and go from the paging to the modify mode by typing <M>. The double cursors should appear. For the time being, we will be concerned mainly with the cursor on the ASCII side of the display. Use the arrow keys to place it over the "N" at the start of the message.

You'll notice a new field, called the cursor address field, has appeared above the input-mode designator. It should now contain the number '17.' Like many of Super Utility's displays, it is in hexadecimal. 23 would be the equivalent decimal value. The field contains the relative position of the cursors within the sector display. The first byte in the upper lefthand corner of the sector display is byte zero. If you count bytes from that corner, you'll find that the "N" (with the cursor over it) is indeed sector relative byte 23 decimal (17 hex). If you're count was off by one, you probably started counting with one instead of zero.

There are other ways you could have moved the cursor to the start of the message. Press <CLEAR>. This will "home" the cursor--move it to relative byte zero back at the upper lefthand corner of the data area. Now press <G>, for "GOTO." The cursor will change to arrows asking to be pointed in the proper direction. Send it to its destination by typing the digits 023. Presto! The cursor is back over byte 23 at the start of the target message.

An interesting point has come up here. Why was it necessary to type '023' instead of just '23'? It's because the decimal input mode requires three characters to specify a byte. Remember, a byte is an eight bit value ranging from 0 to 255 decimal. When you input a number in the modify mode, you always have to enter as many digits as it would take to express 255 in the current input mode. In other words, you need to input two hex digits (since FF hex=255 decimal), or three decimal digits (255 dec=255 dec) or three octal digits (FF hex=377 octal) or eight binary digits (FF hex=11111111 binary). You could circumvent this requirement, in a sense, by keying in fewer digits and then pressing <ENTER>. But since <ENTER> itself is a keystroke, there's not usually much to be gained by that approach.

Here's yet another way to move the cursor to the desired position. Use <CLEAR> to home it again. Now press <L> (for locate). Type 078. Again, the cursor goes where we want it. The trick is based on the fact that 78 decimal, or 4E hex, is the value of an ASCII "N." Verify this by looking under the hex cursor while the ASCII cursor is over the "N." So in asking Super Utility to locate 078, decimal, we sent it off after the "N."

One more time, please. Home the cursor. Now type <L><L>. Again, the cursor's over the same old place. Super Utility took the first "L" to stand for Locate. When you typed the second "L," Super Utility assumed it stood for Last. So it searched for the value used in the last search, 78 decimal or 4E hex. Naturally it ended up in the same place.

Now let's change the input-mode back to ASCII, without leaving the modify mode. Press <CLEAR> while holding down <SHIFT>. You should see a cursor on the input-mode designator at the top of the display's lefthand column. Key <H> and you'll be back in the hex input-mode. Press <<SHIFT> <CLEAR>><A> and you'll be back in the ASCII input-mode.

Now that you're using ASCII input, any alphanumeric key you press (including all punctuation and special characters) will be interpreted as ASCII input. Therefore you no longer have the <G> for Goto or <L> for Locate or Last features available. The <G> and <L> keys will now be interpreted as ASCII input.

Time to change that message. It would be a good idea to enter the new message in capital letters. To avoid the need to hold down the shift-key, engage Super Utility's CAPs lock by pressing <<SHIFT><ZERO>>. Now, type the new characters right over the old message. Both the old and the new message are 17 characters long. The old one has a period at the end; the new one doesn't. So don't type a period at the end of 'BOOT YOURSELF, MAC'.

While you were typing the new message, you may have noticed that the disk didn't spin. In case you're wondering how you can be modifying a disk without the drive turning, here's the answer. For the moment, you're only updating a buffer. The disk remains as is until you're positive that the information is correct as entered. If you make a mistake, use the arrows to put the ASCII cursor back over the error and retype the character. When you've finished entering the new message, press <ENTER>. You'll get the following modification menu:

U>pdate, R>eturn to modify, or C>ancel

If you enter <U>, the changes you made on the video will be saved to the disk sector and become permanent (the sector will be Updated). Incidentally, hitting <ENTER> without any other input defaults to <U>. So don't press <ENTER> unless you're sure you want the sector modified.

Entering <C> will Cancel the changes you've indicated. The unmodified sector will be reread into the sector buffer, and you'll be returned back to Zap's paging mode.

Entering <R> will Return you to the modify mode. Your changes will not be saved to the disk, but they will remain in the buffer so you can continue modifying where you left off. <R> is useful for recovering when you accidentally hit <ENTER> in the modify mode. To save the modified message we've created, type <U><ENTER> from the modification menu. That's all there is to it. If it seemed like a complicated process, it's only because we took many detours along the way, to play with special Zap features. If you want, test the modification now, by trying to boot the target disk.

There are still many Zap features left unexplored. Reboot Super Utility, go into Zap, and display the target disk's boot sector again. Go into the modify mode and put the cursors anywhere near the middle of the screen. Hold down the less-than key (<) for a moment, and watch what happens. This is like "delete" in Scripsit or the Electric Pencil. The character at the cursor is deleted and the characters to its right are moved over to fill the vacancy. The process ends at the end of the current sector. zeroes are "pulled in" to replace the deleted characters. There is no "wrap through" to the next sector. Only what you see on the video changes.

The greater-than key (>) is the inverse of the less-than key, corresponding to a word processor's insert character function. Each time you press it, all the characters to the right of the cursor are "moved over." The entire sector following the cursor is pushed to the right. A zero appears in the vacated slot under the cursor. The last

character in the sector goes over the edge and is lost. Again, there is no wrap through to the next sector. What you see is what you get. Unlike most other printable characters, "<" and ">" work even in the ASCII modify mode. So if you want to type in a greater or less-than sign as a disk modification, you'll have to change the input mode. If you go into the hex input mode (by pressing <<SHIFT><CLEAR> <H>), you can enter a less-than sign by typing 3C. A greater-than sign would be 3E.

Let's look at some of the fancy options available in Zap's paging mode. If you're still in the modify mode (two cursors), go to the paging mode by pressing <ENTER> to get to the modification menu, and then <C><ENTER> to complete the journey back to paging. If you've been staying with me, you're now in the paging mode, still looking at track zero, sector zero of the victim disk. Tap the right arrow key. The display should advance to the next highest numbered sector: sector one. The left-arrow key works the same as the right-arrow key, except that it moves you to the next lowest sector, instead of the next highest. Use it to step back to sector zero.

Now press <<SHIFT><right-arrow>>. As you see, the right arrow seems to work the same way, shifted or unshifted. The difference becomes apparent at a track's highest or lowest numbered sector. These keys auto repeat, so hold down <<SHIFT><right-arrow>> and watch the sectors skip through the display. The last sector you'll be able to read that way will be the highest numbered sector on the track, sector nine. When you attempt to pass it, Super Utility will give you a 'Disk Read Error, Sector not Found' type message. This is a handy warning that you've come to the end of the track. Answer the mini-menu by entering <S> for S>kip. When you find yourself back in the paging mode, use the left arrow to back up a sector or three. Now advance again, this time using the right-arrow without the shift-key. After the track's last sector has been loaded, the display will advance to the lowest numbered sector of the next highest track. The difference is the same for the left-arrow versus shifted left-arrow keys. The unshifted arrow is usually more convenient to use. But if you want to restrict yourself to a certain track, use the shifted arrows.

If you want to jump several sectors (e.g. from sector zero to five), you don't have to step sector by sector. Just press the number key which matches the sector you want to go to. For instance, to go directly to sector five of the current track, just press <5>. If you have double density, you may be wondering what to do if you want to jump to any of the sectors from 10 through 17. To do so, press <S> for Sector. Super Utility will then prompt you to enter the number of the sector you want displayed.

The functions of the up- and down-arrows are similar to those of the left and right arrow, except that the track changes, instead of the sector. The shifted up- and down-arrows take you to the disk's highest or lowest configured track, respectively. To jump to a track without using an arrow key to single step, press <T> for Track. Super Utility will prompt you to enter the track and sector number you wish to examine. If you want to look at a disk in another drive, press <CLEAR> and you will be prompted to enter the drive, track, and sector number.

There is yet another way to step from sector to sector. The greater-than (>)and less-than (<) keys work similarly to the right and left arrow keys. There is a difference, but to demonstrate it, we'll have to create a special track on the victim disk.

Go to Super Utility's format module again. Select the Build Format Track option. Answer the DOS specifier prompt with A. Answer the Track prompt with five. When you're invited to press <ENTER> to see buffer, do so. You will then be in the memory zap mode. Make sure the input mode is hex. Then Press enter to go into the memory modify mode.

Type <L> (for "Locate,"). Then type FE. The cursor will position itself over the first FE in the sector. As described elsewhere in this book, FE is the ID address mark which declares the start of a sector header subfield. The first byte after the FE should be an 05, indicating the target sector is on track five. The next byte, the header byte, should be 00. The following byte is the sector number. Super Utility will not always start the track with the same sector. But if this is the first format track you've created since booting up Super Utility, the sector byte should be 08, indicating sector number eight.

Place the cursor over the sector byte. Now change it to 40 by simply typing 40 over the 08. What we've done is to create a memory image of a nearly normal formatted track. The only reason for the "nearly" is that it has no sector eight. Instead, it has a sector 64 (40 hex=64 decimal). The next step is to put that memory image onto an actual track.

Press <BREAK> to get back to the Format menu. Select option 5, Write Format Track. Answer the drive and track prompt with the drive number or the target disk and track five. The result should be a sector eight-less track five on the target disk.

Press <<SHIFT><BREAK>> and go back to the Zap menu. Display track five, sector zero. Now hold down either the right-arrow key or the shifted right-arrow key and let Super Utility zip through the sectors. After sector seven, Super Utility will stick for a moment, and display a sector not found error. This is natural enough, since it's looking for sector eight and there's no such sector. Answer the mini-menu prompt with <S> for S>kip. Then use the left arrow to back up a few sectors. Now start your approach again, this time using the greater-than key (>) to step through the sectors. As you see, Super Utility now has no trouble at all finding sector nine, which is the next configured sector after seven. After nine, the greater-than key will step Super Utility to sector 64, which is the track's next highest configured sector. To sum up, the greater-than key finds the next highest configured sector on a track, regardless of the numbering scheme. The less-than key (<) works similarly, but finds the next lowest sector.

Zap's "Read ID Address Marks" function has two modes. In mode one, the last three columns of data (headed "CKCRC," "IBM," and "DATA) are not displayed, and in mode two, they are. You may toggle between modes by holding down the <X> key until the mode changes.

Though mode two has more information than mode one, for most purposes, mode one is preferable. The reason is that it works more quickly, and a certain disk r.p.m/interrupt phase lock problem is less likely to develop. This phase lock problem has to do with the fact that an accurately timed disk drive spins at the rate of 300 revolutions per minute, or five times per second. TRS-80 interrupts occur 40 times per second. Note that 5, the number of disk revolutions per second, is an exact divisor of 40, the number of interrupts per second.

This relationship can cause problems which have plagued many TRS-80 owners. It is responsible for some of the "silent deaths" or "time-outs" which darken the lives of TRS-80 users. Super Utility is fairly immune to this difficulty, but it can slow things down when using "Read ID Address Marks" in mode two. It can also make it difficult for mode two to find all the sectors on a track.

One way to help over come this problem is to break the phase lock by pressing <SPACE> to freeze the action and letting the disk stop turning. Then restart it by pressing <ENTER>. Read ID Address Marks' default mode is mode one. Mode one is not prone to phase lock problems.

The up and down arrow keys have auto repeat in Zap's "Read ID Address Mark" function. However, auto repeat is very slow in mode 2 (the mode in which the three right-most columns of data are displayed (see note one). Auto repeat is also slow when Super Utility has trouble reading the ID marks. In either of the above cases, you will do better to tap the arrow key repeatedly, instead of holding it down.

Super Utility has a "mini-menu" which is invoked whenever a disk I/O error is encountered. At such times, you will usually see a description of the error condition followed by the mini-menu. It should look something like this:

R>etry, S>kip, C>ontinuous, N>onstop, or Q>uit ?

If you answer the prompt with <R>, Super Utility will try the disk I/O operation once more. If it succeeds, it will continue with what ever function was under way when the error occurred. If it fails, you will be returned to the mini-menu.

Selecting <S> from the mini-menu will skip the problem sector, causing Super Utility to resume the interrupted function at the next sector.

Selecting <C> is like selecting <R> repeatedly. After <C> has been entered, Super Utility will go into an indefinite retry loop which will continue until the disk I/O succeeds, or you interrupt it with the <CLEAR> key. If you press <CLEAR>, you will be returned to the mini-menu. If, on the other hand, Super Utility eventually succeeds with the difficult I/O, it will continue with the interrupted function. If another problem sector is encountered, Super Utility will return you to the mini-menu. Selecting <N> from the mini-menu is like selecting <C>. Super Utility will automatically retry until it succeeds or is interrupted. However, selecting <N> also causes the mini-menu to be bypassed on future I/O errors. When such errors occur, Super Utility will automatically go into the continuous retry mode. If Super Utility gets stuck on an unreadable sector, interrupt it with the <CLEAR> key. You may then use the <S> option, which will cause Super Utility to skip the impossible sector and resume its task. Choosing the <S> option at this time will also cause Super Utility to revert to prompting you with the mini-menu whenever an error occurs. All this has been covered in your Super Utility manual. The only reason I'm rehashing it here is to bring up the following point:

If you choose <N> from the mini-menu prompt, it will take you out of the mini-menu-prompt-mode. Super Utility will remember this mode change until you reverse it by pressing clear and entering <S> to skip a sector. If you don't do this, you may later use another Super Utility function and, upon encountering an error, be thrown

into a retry loop. For instance, you might use the <N> option during a Sector Verify. Then you might perform a number of different Super Utility operations. If you don't encounter any I/O errors, you may forget you're in the no-prompt mode. 20 minutes after the Sector Verify, you might try to read a sector with Zap. Suddenly, the disk starts repeating the same operation over and over again, and an error message flashes on the screen and disappears.

For this reason, sometimes when this manual says you will get the mini-menu, you won't. Instead, you'll go into the automatic retry loop. When this happens, simply wait a few seconds to give Super Utility a chance to successfully complete the operation. If it continues to fail, just press <CLEAR> to get the mini-menu prompt, and then enter <S> to skip the problem sector and revert to the standard mode.

AFTERWORD
by
Kim Watt

First off, I would like to express my appreciation to Paul Wiener for his professional work. He has provided SU+ users with a great piece of art which is technical and novice both in the same breath. During some of those mind stretching moments that we all had in reading this manual, Paul has lightened things up a bit allowing us to take a deep breath and absorb it all in. Also, I would like to thank Renato Reyes and Dennis Brent for their technical coordination and editing. All in all, this manual is very complete in every aspect, and SU+ users who started out as novices before reading this manual, should be 'intermediate' users by this time.

In reading this book, those of us who are not real familiar with the machine we are using will have learned a lot of new material. So at this time I would like to take the next step forward and help everyone to be an expert SU+ operator.

A lot was mentioned in this book on how to configure the SU+ system, but let me tell you how to do it in "expert" mode. To be an expert, you must go ahead and trust yourself and SU+ enough to HARD CONFIGURE your SU+ diskette. I know that many of you have refused to take off the write protect tab on your master disk, so I am giving y'all my permission to go ahead and do it right now. Go right ahead and configure like a madman. Make everything just right for the way that you normally use SU+ using instructions wherever you find them. Don't worry about things like where the directory is (unless ALL your disks are the same), and what type of DOS you have set. What you want to do is tell SU+ how fast your drives can go, how long to wait on the first disk access, and if the drive is actually in the system or not. If you don't have 4 drives, mark them with a minus symbol (-) as the first character. SU+ won't bother with them anymore. Then go ahead and write that data back to the disk.

Now that wasn't so bad after all, was it?!

Everytime you boot in your disk from now on things will be just the way you want. Go ahead and try it right now.

At this point, every time you boot up SU+, it will know exactly where everything is, and how fast to make it go. This will save a little time whenever you use the SU+ program. No more need to configure, just jump right over to the section that you need, and go, go, go.

The SU+ program, as we all know, is very huge and very complicated. But by using the menu structures, complicated commands may be passed to the program easily without having to memorize lengthy rules of syntax. This is what allows even the beginning SU+ user to manipulate data in a sophisticated manner. A new user who boots up SU+ for the very first time really doesn't realize just how much power has

been made available to him. After reading the manual, and playing with the program for awhile, the user moves his skill level up to the novice category. If the user is very familiar with the TRS-80 and disk systems in general, he can easily move himself up from there to the intermediate level. Some of us, however, need more time to reach this level. We thirst for knowledge, not being satisfied with knowing that pressing this button will repair a GAT table, we want to know WHY, and by the way, just what the hell is a GAT table anyway?!

SU+ has default conditions spread throughout the program's operation. Use these defaults to their advantage. We know that drive 0 is the default for every drive prompt, so why don't we just go ahead and use drive 0 as our standard. This will force us also to remove the SU+ diskette from that drive, and tuck it safely away for the next time it is needed. Using this technique, from the main menu, we can just press <ENTER> and hold it down, and the repeating keyboard will take us right into the zap menu, and answer prompts and display track 0 sector 0 of the disk. No need to select the menu options by number if it is the first one. Remember, SU+ always defaults to the first condition displayed on the video. <ENTER> will go to the zap menu just as fast as <1> <ENTER>, but with half of the keystrokes. If we had wanted to zap drive 3 instead, we could have hit the <ENTER> key twice from the main menu, and entered <3> <ENTER> at the resulting prompt. Likewise, if you are zapping, and are in the modify mode, and wish to write the sector back to the disk, just press the <ENTER> key twice. Learning these things will help you speed up your throughput on the program's operations.

Many techniques and theories have been offered on how to recover data from a diskette that has gone sour on us. All of the discussions have been very thorough, and some very enlightening. I will now give away a big secret on diskette recovery. Use retries heavily as disk errors come up. Believe it or not, this simple procedure will get everything off of the bad disk that is going to come off. All of the correctable CRC errors should be corrected during this process, and if that was the only type of error encountered, you should have a fairly accurate copy at the end of it. Now you can give SU+ a run for its money by read protecting the directory, and repairing the HIT and GAT sectors.

If all of this doesn't work, you can just about kiss those files goodbye, or else hire a professional to muck through your disk and re-enter the directory by hand. Enough just cannot be said about making backups of your valued disk files. SU+ will come in handy when your originals fail and you have no backups, but there is no reason for you not to have a backup. SU+ (and you) would much rather be copying one good disk to another than trying to make some sense out of a jumbled disk.

When you have recovered data from the bad disk, don't try to re-use the disk like it is. Just go ahead and reformat it after copying off all of the files. If it appears flaky during formatting, don't use that disk anymore. I don't feel at all bad about throwing a $5.00 diskette in the trash can when I think of myself punching holes in the wall later when an entire day's work suddenly disappears to NOT FOUND messages. Always use high quality diskettes from a reputable manufacturer. You spent all that money on the computer, don't go feeding it dog food now.

The final step in becoming the 'expert' user of SU+ is to use it all the time. Nothing is like hands-on experience. We could publish three books a week on SU+ and never convey the same experience as being right there pushing the buttons. Go ahead

and get a feel for how everything works. Back up a few diskettes and practice on them. If you make a mistake, you will not hurt anything, and you will get a good feel for what you and SU+ can do together. Hopefully you will make a good team!

Dallas, Texas
June 1982