THE GUILFORD 99'ER NEWSLETTER

VOL.3 NO.3 MARCH 1986

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to P.O. Box 21691, Greensboro, NC 27420. The Software Library is for dues paying members only. (Herman Geschwind, Editor)

OUR NEXT MEETING

DATE: MARCH 4, 1986. TIME: 7:00 PM PLACE: Glenwood Recreation Center, 2010 S. Chapman Street.

The subject of our March meeting will be a demonstration of "mini-utility" programs . A series of short programs to aid in various aspects of programming tricks and shortcuts. After the regular program, part 2 of the TI-Writer tutorial will be given. Part 2 will cover use of the Text Formatter.

TI SHOPPER

by Bob Carmany

Last month, we touched on an area of software that is not normally noticed by the TI owner. This is very unfortunate because there are some excellent programs included in this relatively new concept called "freeware". The "freeware" concept is the marketing of low-cost programs that don't have the hype of an INFOCOM or ATARISOFT advertising campaign. In fact, the only way that you might find out about "freeware" is through a friend or in the pages of MICROpendium.

Last month, Herman Geschwind did a brief review of the "freeware" disk directory program by Marty Kroll. The program (even without documentation) is one of the best around. I catalogued my personal library of some 538 programs with it and had no trouble at all. The menu is self-explanatory and the sort took some 57 seconds for the entire library. It is certainly worth the $10 that Marty Kroll asks for the program.

That brings us to the heart of the "freeware" concept. The programs are really not free -- that is, if you have a conscience. Most of the authors ask a "donation" of $5 to $10 and request that you pass a copy of the program along to a friend. They don't expect an overwhelming response from the monetary end but I'm sure that they would appreciate whatever they get.

There are several outstanding "freeware" programs on the market. Danny Michael, Route 9, Box 460, Florence, AL 35630 has two outstanding programs available. NEATLIST is an assembly language utility that will produce a cross-listing of your XB program variables in an easy to read form. SCREEN DUMP (now in version 3.0) is one of the best screen dumps available for Epson/Gemini printers available. Both programs come with full documentation and are easy to use. He asks that you send one initialized disk for each with a postpaid mailer or, if you send him $5 for each, he will supply everything including the disk. Danny says that if you like the program, you may pay him but not more than $10 --- and pass a copy along!

Barry Conor, 2 Cleveland Crescent, Dartmouth, Nova Scotia, Canada B3A 2L6, has a Wycove graphic program available. Future versions are to include printer dumps and disk file saving routines. An initialized disk, postpaid mailer and $1 will get you this program. He also welcomes questions about Wycove Forth.

Rene LeBlanc,8719 E. San Lucas Dr., Scottsdale AZ 85258, has a reconfigured version of TI-Forth available. He asks for a postpaid mailer and two SS/SD disks for the system which is based on Tom Freeman's XB version of TI-Forth.

MICROpendium has an XB loader for TI-Forth available. Send a mailer and a SS/SD disk to them at P.O. Box 1343, Round Rock, TX 78680.

Bill Knecht of the Houston UG has two "freeware" disks of music programs wit graphics. The programs are in XB and require memory expansion. Two SS/SD disks and mailer are required or send a check for $6 to him at 816 Yorkshire, Pasadena, TX 77503.

And last but not least, FILEREADER is a program that will read most types of files and either dump them to a printer or re-write them as a DIS/VAR 80 file that can be read by TI-Writer. Again a disk and mailer or $5 and the author will provide everything. This one is from Martin Smoley, 6149 Bryson Dr., Mentor, OH 44060. He also asks $2 if you are pleased with the program.

This is a representative cross section of what is available in the "freeware" market at the present time. If you get a copy of the program, the authors would greatly appreciate a $5 or $10 donation. To be truthful, programs like the Danny Michael duo and the public domain DM 1000 and the new TI-Writer XB based programs are superior to what is available on the commercial market. And, a little donation will encourage these active authors to continue to write programs for the benefit of the TI community.

FUN WITH THE GRAM CRACKER

by Herman Geschwind

After having received the GRAM KRACKER from Millers Graphics, the first order of business was to save all those valuable cartridges to disk and patch Extended Basic with the new "Call" subroutines provided by Millers Graphics. But there is more than that which the GK will do with its powerful memory editor and the ability to save the Operating System to disk and reload it as an editable GRAM.

One of the things that has irritated me for a long time is the character set provided with the 99/4A, particularly the lower case letters, which really are not lower case at all but rather a smaller version of the upper case set. For use with TI Writer and FastTerm I had patched in a disk file with a much better looking character set which was developed by the Titan Users Group of Raleigh, NC. The objective now was to load this character set in GRAM so that it would be available permanently, also for Basic, Extended Basic and other applications. To locate the lower case character set table in GRAM proceed as follows:

From the GK main menu select 5) Edit Memory. Press Function 1 to select Grom/Gram, then Function= to toggle HEX display. Press Function 5, "Search" and for Start and Finish key in 0000 and 1000 respectively. Next press Function 9 and after disabling Write Protect key in the string 00003848484834. This is the Hex representation of the character "a" or ASCII 97. Move the cursor to the last "4" of the search string and press enter.

The Memory Editor should now show the start address of the LC character table. For my console this address was g0858. This was for a version 2.2 console. This address more likely than not will vary slightly from console to console.

Once having located the starting address, press Function 9 to place the cursor in the memory window at g0858 (or whatever your starting address may be) and carefully type in the following table of HEX values (without the addresses in HEX which are for reference only).

>085B 00003848484834 >085F 60203824242478 >0866 00003840404038 >086D 0C0838484843C >0874 000038447C403C >087B 18242070202020 >0882 00003C443C0438 >0889 60202834242424 >0890 1000701010107C >0897 08001808084830 >089E 20202428302824 >08A5 3010101010107C >08AC 00006854545454 >08B3 00005824242424 >08BA 00003844444438 >08C1 00007824382020 >08C8 0000304838080C >08CF 00005824202020 >08D6 00003C40380478 >08DD 20207820202418 >08E4 00004848484834 >08EB 00004444282810 >08F2 0000D454545428 >08F9 00004428102844 >0900 000044443C0478 >0907 00007C0810207C

A second change involves replacing the zero with a slashed zero (for this tip I am indebted to Millers Graphics). Following the search procedure described above, look for the string 0038444444444438 which should be located at g0720. Once located, replace it with 003844C54644438. Next look for 7C444444444447C which is the horrible looking squarish capital "O". Replace it with 0038444444444438 and now you have the original zero as a replacement for the square "O".

As a last step in customizing search for the letters TEX. (Should be at address g046E or thereabouts). Once located type in your first and last name. (If they are longer or shorter than Texas Instruments, abbreviate or key in blanks to make it fit). After having made all these changes, be sure to restore the GK write protect and return to the GK Main Menu and then to the title screen. If everything went fine, you should now have your personalized color bar title screen.

Select Extended Basic and key in: 100 FOR I=97 TO 122::PRINT

This is test to see that your lower case character set is ok. As a last step use the GK so save your customized operating system to disk. Have fun!!

USING THE MERGE FORMAT

by Robert Petrocone

(Continued from last month)

In the first part of this article we learned how to write a program which could output lines in the merge format. These lines could then be merged into memory and we would have a program. Using this process could be very useful , especially when a program requires repetitive lines of the same statement, but with different values. ie: DATA, CALL CHAR etc. The process we learned however, would involve writing many long lines of CHR$ values etc. but, since many things about the contents of a line may be predicted, we can shorten and simplify the process by using defined functions and a couple of other things.

PRINTing THE LINE OF CODE

From what we have learned, we know there are two standard parts to a line of code, the line number, represented by two CHR$ and a CHR$(0) which marks the end of the line. Assuming that the line numbers are sequential, they can be predicted.

First off, for now on, let us make LN equal the line number and LINE$ equal the line of code without the line number or CHR$(0). Now, we can write a line like this:

```
100 INPUT "ENTER FIRST LINE NUMBER:":LN
-
-
- Program generation of LINE$
-
-
500 PRINT #1:CHR$(INT(LN/256))&CHR$(LN- 256$INT(LN/256))&LINE$&CHR$(0) :: LN=LN+10
```

Here we can see that the program line is automatically figured and incremented and the CHR$(0) is also added.

THIS EQUALS THAT

Another way to shorten the process is to make CHR$ values for the different commands equal to a string variable. For example:

```
PRINT$=CHR$(156)
READ$=CHR$(151)
```

Remember, the values for these equalities come from the generated program from the first example in part one of this article. By doing this, we can write lines of code without having to continually look up the correct values for the different statements.

DEFINED FUNCTIONS

There are still some statements that can not be handled by just having strings equal the proper CHR$ value as above, these are quoted and unquoted strings, CALL statements and line numbers which are in conjunction with GOTO's, GOSUB's etc. Fortunately, these can be handled quite nicely by defined functions.

First, lets look at quoted and unquoted strings. We know that a quoted string must be preceded by CHR$(199) and that a unquoted string must be preceded by a CHR$(200). Following that, there must be a CHR$ which gives the length of the string and finally, you give the string. This can all be handled easily with two defined functions like these:

```
FOR A QUOTED STRING: DEF QS$(X$)=CHR$(199)&CHR$(LEN(X$))&X$
```

FOR A UNQUOTED STRING: DEF UQ$(X$)=CHR$(200)&CHR$(LEN(X$))&X$

Now, when you are writing a line of code and you have to enter a quoted or unquoted string, just use the defined function. For example, if you were going to print 'HELLO', you would type this: LINE$=PRINT$&QS$("HELLO")

This example also assumes that PRINT$ has been set equal to the correct value as I discussed previously in this article. Note, after you had typed this, you would branch to the line were it prints the line as we learned above under PRINTing THE LINE OF CODE.

CALL statements are similar to unquoted strings but, with a extension. A CALL statement is the CHR$ value for CALL plus a unquoted string for the function. ie: CLEAR, SCREEN etc. Here is the defined function for a CALL statement which must be used in conjunction with the defined function for a unquoted string: DEF CALL$(X$)=CHR$(157)&UQ$(X$)

Here is a example which does a CALL CLEAR:

```
100 DEF UQ$(X$)=CHR$(200)&CHR$(LEN(X$))&X$
110 DEF CALL$(X$)=CHR$(157)&UQ$(X$)
-
-
- Open file, input starting line number etc.
-
-
-
500 LINE$=CALL$("CLEAR")
510 GOSUB 1000
520 CLOSE #1 :: END
1000 REM FIGURE LINE NUMBER AND PRINT LINE$ TO FILE
```

Finally, there are line numbers within a line which can be done with a defined function. They are handled similar to a quoted or unquoted string except that they are preceded by a CHR$(201) and that no length is given. Following the CHR$(201) are two more CHR$ which give the line number. These are done exactly as the regular line numbers are done. Here is the defined function: DEF LN$(X)=CHR$(201)&CHR$(INT(LN/256))& CHR$(LN-256&INT(LN/256))

So, GOTO 700 may be written: LINE$=GOTO$&LN$(700)

Assuming GOTO$ has been assigned the correct value.

LOWERCASE
by Bob Carmany

Here is a short program that will produce a series of true lowercase letters in your programs instead of the "small caps" of the normal TI character set. When you RUN the program, the character set will appear across the bottom of the screen. If you want to use the characters in one of your own programs, simply erase line 100, 180, and 190 and resequence the remaining lines so that they are at the beginning of your program. Then, whenever you select a lowercase letter, it will appear as the corresponding letter in the demo instead of a small capital letter.

```
100 CALL CLEAR :: FOR U=33 TO 94 :: CALL CHARPAT(U,U$):: CALL CHAR(U,SEG$(U$,3,1 4)):: NEXT U
110 CALL CHAR(97,"000038043C243C",98,"202038242424380",99,"000018242024180",100,"0 4041C2424241C")
120 CALL CHAR(101,"00003C243C203C",102,"081410103C101",103,"00001C24241C0418",10 4,"20203824242424")
130 CALL CHAR(105,"0010001010101",106,"0008000808080083",107,"2020242830282424",108 ,"3010101010101")
140 CALL CHAR(109,"00007854545454",110,"00003824242424",111,"00001824242418",112 ,"0000382424382020")
150 CALL CHAR(113,"00001C24241C0404",114,"00003C2420202",115,"00003C203C043C",11 6,"001038101010180")
160 CALL CHAR(117,"0000242424241C",118,"0000242424181",119,"0000545454542B",120, "000044281028440")
170 CALL CHAR(121,"00002424140B102",122,"00003C0810203C")
180 PRINT "abcdefghijklanopqrstuvwxyz"
190 FOR DE=1 TO 2000 :: NEXT DE :: END
```

FORTH FORUM
by Bob Carmany

This is the last in the series of disk management files written in Wycove Forth. The last six screens will allow you to copy entire disks and handle all of the tasks that you formerly had to use DISK MANAGER or one of the other disk utility programs to do. Now, you will not have to exit Forth to perform any of these tasks.

All you have to do to copy a disk is to type in the designation for your destination disk (ie. SSSD for Single-Sided Single Density, etc.). Make sure that you have ALL of the disk utilities loaded before you start, though. The 2 Drive Backup Program makes use of some of the words contained in Screens 26-39. Remember, after you type the individual screens in, you can save them as any number that you wish with the SAVE-SYSTEM word.

If there are any questions that you would like answered about Forth, feel free to send them in to the newsletter and we will try to get an answer for you. Or, if anyone is interested in a series of Wycove Forth Terminal screens, they are also available. Just send in a request for what you want and we will try to get it for you.

This was to have been the "end" of the "Forth Forum" as a column. It is not to be, however. There have been some inquiries about other screens for both Wycove and TI-Forth so it looks like the column will continue for a while, at least. Future columns will deal with sound and graphics for both systems as well as some more utility screens.

SCREEN # 37

```
 0 ( Read several blocks from a file )
 1 ( Block numbers start at 0 )
 2 ( Data must be in VDP. As written it
 3   must be at PAB@+10 )
 4 : FILE-BLOCK-READ ( reads PAB@+10 )
 5   ( first-block# #blocks adr len
 6     disk# -- #blocks-read f )
 7   >834C C!
 8   PAB@ DUP >834E ! !FN>VDP
 9   >834D C!
10   >8320 DUP >8350 C!
11   PAB@ 10 + ( dest VDP address )
12   OVER ! 2+ !
13   >114 DISK-DSR >20 ?DISK-DSR
14   >834C @ SWAP ;
15
16
17
18
19
20
21
22
23
24
25
```

SCREEN # 38

```
 0 ( Write several blocks to a file )
 1 ( Can overwrite existing blocks or
 2   extend a file as needed. Will not
 3   shorten a file. )
 4 : FILE-BLOCK-WRITE ( from PAB@+10 )
 5   ( first-block# #blocks adr len
 6     disk# -- f )
 7   >834C C!
 8   PAB@ DUP >834E ! !FN>VDP
 9   >834D C!
10   >8320 DUP >8350 C!
11   PAB@ 10 + ( source VDP address )
12   OVER ! 2+ !
13   >115 DISK-DSR >20 ?DISK-DSR ;
14
15
16
17
18
19
20
21
22
23
24
25
```

SCREEN # 39

```
 0 ( Store file directory information,
 1   creating a new file. Destroys any
 2   previous data in the file. Sectors
 3   may be preallocated if desired. )
 4
 5 : !BYTE ( n1 n2 adr y -- n1 adr+y )
 6   >R SWAP OVER C! R> + ;
 7
```

SCREEN # 25

```
 0 ( 2 DRIVE BACKUP PROGRAM )
 1   0 VARIABLE COMPLETE
 2   0 VARIABLE BPOINT
 3   >0102 VARIABLE DBITS
 4         >0408 , >1020 , >4080 ,
 5 ( Drive Numbers )
 6 1 CONSTANT MASTER  2 CONSTANT CDRIVE
 7 : ?CHECK ( f -- )
```

```
 8 : FILE-INFO-WRITE
 9   ( #data-sectors flags #recs/sector
10     eof-offset rec-len #recs-or-sect
11     adr len disk# -- f )
12   >834C C!   0 >834D C!
13   PAB@ DUP >834E ! !FN>VDP
14   DUP >FF AND SWAP >100 U* SWAP DROP
15   >8320 DUP >8350 C!
16   9 + -1 !BYTE -1 !BYTE -1 !BYTE
17   -1 !BYTE -1 !BYTE -1 !BYTE 1- !
18   >115 DISK-DSR >20 ?DISK-DSR ;
19
20
21
22
23
24
25
```

SCREEN # 26

```
 0 ( 2 DRIVE BACKUP PROGRAM )
 1 : COPY-SECTOR ( bitadr sector# --
 2   bitadr : copy sector if necessary )
 3   DUP >R 8 /MOD SWAP >R OVER C@VDP
 4   R> DBITS + C@ AND COMPLETE @ OR
 5   IF R READ-SEC ENDIF R> DROP ;
 6
 7 : XBACKUP ( cdrive density #tracks
 8   #sides sectors 0=normal/1=full -- )
 9   >C00 PAB !
10   1 FILES
11   COMPLETE ! >R
12   FORMAT ?CHECK
13   0 MASTER RDS ?CHECK
14   -1 >D00 !VDP >D00 BPOINT !
15   ( initialize list of sectors )
16   >C38 ( pointer to sector used )
17   R> 0 DO
18     CURSOR-POS DUP @ I 4 .R SWAP !
19     I COPY-SECTOR
20   LOOP CLR-BUFS
21   DROP >1000 PAB ! 3 FILES
22   4 SPACES ;
23    -->
24
25
```

```
 8   -DUP IF
 9   CR ." Error code " . ." detected
10   ENDIF ;
11 : CLR-BUFS ( -- : read in sector )
12   BEGIN
13     BPOINT @ @VDP DUP 1+ WHILE
14     ->102 BPOINT +!
15     BPOINT @ 2+ PAB ! CDRIVE WDS
16     ?CHECK
17   REPEAT DROP ;
18 : READ-SEC ( sec# -- : read sector
19                link it to others )
20   >8370 @ BPOINT @ - >106
21   IF CLR-BUFS ENDIF    BPOINT
22   @ 2+ PAB ! DUP MASTER RDS ?CHECK
23   >102 BPOINT +! BPOINT @ !VDP ; -->
24
25
```

SCREEN # 27

```
 0 ( Various backup options )
 1
 2 : SSSD CDRIVE 0 40 0 360 0 XBACKUP ;
 3 : DSSD CDRIVE 0 40 2 720 0 XBACKUP ;
 4 : DSDD CDRIVE 2 40 2 1440 0
 5   XBACKUP ;
 6 : SSDD CDRIVE 2 40 0 720 0 XBACKUP ;
 7
 8 ( To copy non-standard disks, e.g.
 9   TI-FORTH use complete=1 )
10 : TI-COPY CDRIVE 0 40 0 360 1
11   XBACKUP ;
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

NOTES FROM THE PROGRAM CHAIRMAN

by Bob Carmany

Following is a preview of programs planned for 1986:

MARCH SPECIAL: Kernersville Library. A joint meeting with the Winston Salem users group. A demonstration of TI Logo. (Date, time and directions will be in the next newsletter).

APRIL: April's program will demonstrate TI-Writer in uses other than as a word processor. The program will focus on how to use TI-Writer to write Basic or Extended Basic as well as Assembler programs. Part 3 of the TI-Writer tutorial will follow

the regular portion of the meeting. This presentation will deal with the use of the "Utilities" option.

MAY: The program for May will deal with "Graphic Tricks". There will be a demonstration of short programs that produce graphics and a discussion of how they can be used as subroutines or incorporated into longer programs.

JUNE: File management will be the topic for this month. The emphasis will be on how to organize, recall, and manipulate records and set up a simple data base.

JULY: July's program will deal with the use of CALL LOAD on the TI. There will be illustrative programs that will increase the speed of some procedures as well as how to "peek" into the inner workings of the TI.

AUGUST: With the beginning of school, the program for August will be Educational Games. Not necessarily the games that come on cartridge or are commercially produced, but games that you can create at home for your children or get from the group library.

SOFTWARE NEWS
by Herman Geschwind

C99: What is utterly amazing is the wealth of programming languages that became available for the 99/4A after "Black Friday". In addition to Basic and Extended Basic, Line-by-Line and Editor/Assembler, TI-Forth and Mycove Forth we now have a reasonable subset of the "C" language, thanks to Clint Pulley of Canada. "C" is a compiled language (the compiler created by Clint actually writes Assembler source code which then needs to be assembled to be executable). I downloaded the compiler, support libraries and instructions along with several test programs. The compiler is very fast and the assembled code is very compact. I was impressed enough to invest in a more generalized book on "C" to explore this further. What impresses me is that "C" code can be as simple as a Basic program or as involved as Assembler, depending on your skill level and the objectives of your program. "C" is currently the programming language of choice for software houses that write for the IBM PC, the 68000 variant of the Atari, the Apple MacIntosh and the Commodore Amiga. Of all the programming languages "C" is claimed to be the most processor independent which means that "C" source code can be re-compiled for different systems with great ease. Clint is to be commended for his effort in making this interesting language available to the TI community.

GPL-DIS: The Graphics Programming Language, for so long TI's closely guarded secret, is now becoming available to the TI community at large. Unlike "C", GPL is very processor dependent in being restricted to the 99/4A. TI used it to write the operating system and much of the cartridge software. GPL is a slightly higher level variant of Assembler, which means that certain tasks can be accomplished in GPL with one or two statements where several dozen would be required in Assembler. GPL also will compile to much tighter code than the same program in true Assembler. GPL-DIS is the first harbinger of things to come. GPL-DIS will disassemble GROM or other GPL code. Output can be directed to screen, printer or disk in HEX or ASCII. What is needed now is a GPL compiler so that GROM can be disassembled, changed and re-compiled! The possibilities that this opens up are very tantalizing indeed. Millers Graphics has promised a GPL Assembler/Compiler and manual for release in early 1986. Interestingly enough GPL-DIS was written by a TI user in Germany.

MICROSOFT MULTIPLAN
by Tom Kennedy

ELECTRONIC SPREADSHEETS...CELLULAR ANALYSIS...FORMULAS... CELLULAR REFERENCING... WORKSHEETS...ABSOLUTE REFERENCING...

These are buzzwords of a form of Data Processing that on the surface appears to be incomprehensible to all but accountants and the bridge crew of the Star-Ship Enterprise. As Word Processing is to writing a letter, Data Processing is to using a multiplication table.

Many people have a hard time using spreadsheets, because working with data in this format is similar to learning a new language. But once you learn to use the commands, and the procedure of working with data in a two-dimensional row/ column format instead of a one-dimensional equation, you'll find many ways in which Multiplan will allow you to "crunch numbers" faster and easier than using a calculator and notebook. More than that, Multiplan is flexible enough to be used anytime you want to display, or use, numbers or words in a row/column format. In fact, you could even adapt Multiplan to use it as a Word Processor!

What is a spreadsheet? In business, you often hear reference to "our books". The "books" that the businessmen, and you & I, keep are a pen & paper record showing the Debits and Credits of various bills paid and assets gained, plotted against a

scale of time. Each intersection of row and column contains an entry for a value. The last column and/or last row contain a summation of all previous columns or rows. In an electronic spreadsheet, you recreate the printed form with the addition that each "cell" (a row/column intersection) can also contain a mathematical equation, or "formula", that automatically acts upon pre-defined cells and displays the result accordingly. If any value in any cell is changed, the formula instantly updates displayed results. This self-maintenance ability is what pays off in using an electronic spreadsheet, such as Multiplan.

To operate the Multiplan software on the TI, you must have 32K memory expansion, and at least one disk drive. An RS232 card and a printer are also handy, but not mandatory because unlike word processing, where the end result is a printed piece of paper, the end result with a spreadsheet is useful data, which may be used many ways. Most worksheets are well over 80 characters in width, (up to 2016!) and this requires a cut-and-paste job, so a wide-carriage printer is preferable.

To load Multiplan, you insert the cartridge and program disk, select Multiplan from the menu, and press <ENTER> to load. Before pressing <ENTER> you can select one of eight screen color combinations by pressing the space bar.

The first thing you will see is a grid across the top and left side of the screen. These numbers are the row and column locations in the top left, or "HOME" position. There are 255 possible rows and 63 possible columns, with the screen framing a small section. Each "CELL" is referred to by its row/column location, such as: R1C1, R10C22, etc. In R1C1, The Home position, there is a solid rectangle, as large as the width of the cell. This is your cursor, or "CELL POINTER", which is where any entry will appear. Below the cell grid is the COMMAND LINE, which shows the primary commands you will use. There are a number of sub-commands related to each of these, but you must type the first letter of the primary command first, or place the cursor over the command and hit <ENTER>.Below the command line is the MESSAGE LINE, which prompts you for further information when needed. In the bottom left corner is the current cell pointer location, and to the right of that is the contents of the current cell. Lastly, in the bottom right corner is the available memory space remaining.

Following is a list of the twenty commands shown in the Command Line, with the forty sub-commands that apply to each.

ALPHA The first command given before entering text into the current cell location. All alpha-numerics can be used, but numbers will be treated as text, and can't be used as values for calculations.

BLANK Used to erase the contents of a specified cell or range of cells. Blanked cells retain their location and format.

COPY Allows you to duplicate any cell or cells in any number, including both cell format and content.

DELETE Completely erases a row or column.

EDIT Allows you to edit the contents of a cell, or the formula in that cell, without re-entering the data. Requires careful use of the EDIT keys.

FORMAT Defines all of the various parameters of cell width, content, and display of data.

GOTO Moves your cell pointer immediately to any cell, by giving the row/column or pre-defined name. Also used to move from one window to another.

HELP Calls up a detailed HelpFile (from disk) that covers the whole Multiplan software, including a command summary.

INSERT Inserts a blank row or column, formatted to DEFAULT settings.

LOCK Protects the cell, or formula, from accidental overwrite.

MOVE Moves a cell, or group of cells, to specified row/column, deleting the original.

NAME Allows you to give a name to a cell to aid in future references to that cell. "Total" or "Sales" are examples.

OPTIONS Covers global options such as RECALC, MUTE, & ITERATION. The most important of which is canceling RECALC, to avoid waiting for each entry to recalculate the entire worksheet.

PRINT Used to print the worksheet. Before printing, you must first define the extent of the field to be printed with MARGINS and OPTIONS, then select PRINTER to start output. PRINT FILE outputs to disk instead of the printer to be included into a Word Processor file, or other cases where you need the worksheet stored in ASCII format.

QUIT Self explanatory, provides a "safe" exit.

SORT Sorts entries in cells in a specified column, in either ascending or descending order.

TRANSFER Includes six sub-commands which are used: to LOAD, SAVE, RENAME, or CLEAR an active worksheet. Also, to DELETE a file from a disk, and an OPTIONS command to define disk filename and format.

VALUE Used to enter a numerical value or formula into a cell. This must be used for numbers that will be used in calculations.

WINDOW A window is used to overlay one or more portions of a worksheet with another. As an example, to hold the titles of columns fixed while the data scrolls underneath. The sub-commands define how the windows are opened, closed, or linked together. A border can be defined to offset it from the worksheet.

eXTERNAL Allows related worksheets on disk to be linked as a source of data for the active sheet. Any range of cells can be drawn up for reuse. Multiple worksheets can be linked relative to each other so as to work together.

Upon selecting a command, a command menu appears with a number of response fields shown. In each response field is a proposed response, which is the default selection unless you change it. To use a command, type it's key letter and fill in the response fields. To move through the fields, use the tab key until the cursor is highlighting the correct area. Type in your response, and either tab to the next field or hit <ENTER> to activate the command.

When the necessary response is a row/column cell reference, there are two ways to respond: Absolute and Relative. Absolute referencing is a numerical definition of the cell location, such as R5C10 (the intersection of ROW 5 and COLUMN 10). A group of cells, a RANGE, is called by giving the boundary intersections separated by a colon(:), such as: R2C1:R4C10 defines a 3-by-10 cell grid consisting of columns 1 through 10 on rows 2 through 4.

Relative referencing involves identifying the desired cell by displacement from another cell, usually the one the cell pointer is currently on. As an example, if you are on row 5, column 10, (R5C10) and you wish to refer to a cell two rows up and three columns over, (R3C13) you could type in R-2C+3 or use the cursor keys to move the cursor over R3C13. The relative address will automatically update as you move. When the cursor is in place, hit <ENTER> (or tab to the next field) and the reference is defined.

So far, I have covered what you see on the screen and in response to the various commands; what the commands and key functions are; and how to fill in response fields where needed. Before going on to building a worksheet, you'll need to know how to save what you're working on, and how to load it back in. Besides the fact that you'll want to take a break occasionally, it's nice to be able to experiment with the commands, "messing up" the worksheet, and then loading the "clean" version back in to continue.

The LOAD and SAVE commands are under the command TRANSFER (a lousy name). Hit "T" and the menu will be displayed. Since the first option is LOAD, hitting <ENTER> now will prompt for a filename. To select SAVE, (or another option) hit the first letter and <ENTER>, or tab through to the desired item and hit <ENTER>. When loading or saving, you'll be prompted for a filename the first time, which will become the default response.

Multiplan also incorporates an extensive helpfile contained on disk. When the command line is displayed, you select HELP with either the Help action key (<AID> or "?") or by typing "H". The worksheet will be replaced by the beginning of the helpfile. If a command has been selected, hitting the help key will display the specific section of the helpfile that pertains to the command you are using. The help menu allows you to RESUME (return to command menu), START at the top of the helpfile, or move to NEXT or PREVIOUS page of information.

The first step to creating a worksheet is to decide how many rows and columns you'll need, and how the data will be displayed. It is best to sketch this out on paper to get a feel for how it will look. Also, you'll need to decide what formulas will have to be created that use the data contained in the worksheet. Lastly, you will probably want to change the format of many of the cells, usually by rows or columns. Most often, the formatting required is for display purposes. Cell width, alignment of the data within the cells, etc.

Now that you know how everything will look, begin by formatting the cells. Upon start-up, the cells are set with a number of defaults. You may want to change the widths of some columns, to between 3 and 32 columns, to show all of the entry for the cells. If the data in a cell is too large to fit the width of the cell, it will be truncated to fit, unless it is a numerical entry, where it will be replaced by a sting of "#"'s.

FORMAT CELLS is used to set cell alignment and display format. A cell can be aligned to either center text for columnar headers, etc., or to align data displayed in tables. For instance, a table of dollar values could be shown with a "$" in front and decimal points aligned.

The display formats are used to show how the data appears in a cell. CONTinuous allows the text in a cell to run over the right boundary to the next cell. If all cells are made continuous, you have a word processor-type format. EXP displays numbers in scientific notation. Fixed Point rounds off decimals to a defined number. GENeral is as you see when starting up, values displayed as entered. INTeger rounds off all numbers to integers. "$" (Dollar) adds a dollar sign to numbers and rounds to two decimals. "%" Replaces the number with an equivalent number of asterisks, to use like a bar graph. "%" displays the number in percent form. Lastly, the "--" just leaves the setting at the previous option.

#####TO BE CONTINUED#####