VECTOR

ExecuPlan™

Reference Guide

E x e c u P l a n™

Executive Planning System

and

Interactive Electronic Worksheet

R e f e r e n c e   G u i d e

ExecuPlan Version 1.1
1/23/81

Reference Guide Revision B
1/23/81

7100-4400-10-01

Disclaimer

Revisions

The version of the product and reference guide are shown at the top of this page: A change in the guide itself is indicated by a letter and does not indicate a change in the product. A product change will change the product version and the guide revision.

Table of Contents

Section 2

Commands

Complete detailed descriptions of all commands and how they are used.

C - the CLEAR command

The clear command is used to clear certain locations in the array of the data and/or formulas associated with them. The various forms are as follows.

C - Clear current location. The data and formula will be cleared from the current location.

CF - Clear current formula. If there is a formula associated with the current location, it will be erased. The data in the current location will be left unchanged.

CR - Clear row. The data and formulas for all locations in the current row will be erased. Alternate form:

```
CR row
```

In this case, (row) specified will be cleared.

CC - Clear column. The data and formulas for all locations in the current column will be erased. Alternate form:

```
CC column
```

In this case, (column) specified will be cleared.

CA - Clear all. All data and formulas in the array will be erased.

D - the DISK command

The disk command is used to access disk directories to load, save, or erase files. It is also used to select the current disk to be used for subsequent disk operations.

D - Disk commands. Disk command mode will be entered, using the currently-selected disk.

D x - Disk select/commands. Disk x will be selected, then disk command mode will be entered. The value x must generally be in the range A to P, but specifically must be a valid drive on the system being used at the time.

Disk Command Mode

When disk command mode is entered, the directory of EPL files on the selected disk will be displayed at the top of the screen; the current disk and a list of available commands will be displayed at the bottom. Furthermore, the first file name will be highlighted.

At this point, typing any of the four arrow keys will move the highlight in the appropriate direction. Once the highlight is moved to the desired file, the disk commands below can be used. All disk commands reference the currently-highlighted file.

Disk Commands

L - Load a file. The highlighted file will be loaded into the array. Note that this erases the current array; therefore, if it is desired to save the current array, it should be updated or saved before executing the load command.

S - Save a file. The array will be saved to the highlighted file. The former contents of the file are lost. This command should be seldom used, and is included only for symmetry. Normally, the update command is used to save the array.

D - Delete a file. The highlighted file will be erased from the disk. This is equivalent to using the ERA command under CP/M.

<Esc> - Exit disk command mode. When <Esc> is typed, the user will be returned to the main system.

Each of the disk commands (except <Esc>) requires confirmation. When either L, S, or D is typed, one of the messages

Loading <file> - type Y to proceed -
Saving <file> - type Y to proceed -
Deleting <file> - type Y to proceed -

will be displayed. At this point, typing Y will cause the selected action to be carried out. Any other character, including <Return>, will cancel the action.

If when disk command mode is entered, there are no EPL files on the disk, the message

No files...

will be displayed. Obviously, there are no files to load or delete; therefore the only possible action is to save the array, which in this case MUST be done with the update command. Typing any character will return to the main system.

E - the ENTER command

The enter command is used to enter data or formulas into the array. It has four forms, for entering text, lines, values, or formulas. Since the four are drastically different, they will be explained separately.

ET - Enter Text

The enter text command will accept a single argument and write it into the current location. The general format is

```
ET text
```

The format for the ET command is somewhat precise. There must be one character after the ET, then whatever is after that is considered the text. For example, in the command

```
ET Hello There<                    (the < indicates the cursor)
```

will enter the text "Hello There". On the other hand, the command

```
ET  Income Type <
```

will enter the text " Income Type ". In other words, exactly what you type is what you get. Since whatever the user types is taken as the text, no multiple form of the ET command is possible (unlike the other enter commands).

EL - Enter Line

The enter line command is used to enter a line of data where the data is simply one character repeated. This is typically used for a dividing line of some sort, or perhaps the line after a column of numbers above the total, or something similar. There are two formats of the enter line command:

```
EL character
EL character ncolumns
```

The first type will create a line consisting of the (character) all the way across the array. The second will create a line only extending across a certain number of columns, specified by (ncolumns). In either case, the current location will be used as the first (leftmost) column; that is, the column in which the line begins.

The EL command operates much like the ET command. The data in the line is

simply treated by the system like any other text. The EL command automatically creates the sequence of characters to be the exact width of each column as it is entered. This can be used to create a "broken" line, also. For example, if the widths of all columns are set to 10, then the EL command is used, then the widths of all columns are set to 12, there will be a 2-character break in the line between each of the columns.

EV - Enter Value

The enter value command and its variations are used to enter numeric values into the current location and possibly adjacent locations in the array. The simplest form is

```
EV number
```

which will write the value (number) into the current location. Additionally, this form of the command can be "implied" by simply typing

```
number
```

when in command mode. That is, typing "123" and "EV 123" are the same.

The second form of the EV command is used to enter values into adjacent locations, either across a row or down a column. The forms are

```
EVR number number number number etc.
EVC number number number number etc.
```

The first will enter as many successive numbers as are typed into the current location and successive columns on the same row; the second will do the same, only the numbers will be entered into successive rows on the same column. For example, if the current location is [1,1], the command

```
EVC 1 2 3 4 5
```

will enter 1 into [1,1], 2 into [2,1], 3 into [3,1], etc.

The final form is the "repeat" form. Is is used to enter the SAME number into successive locations. The forms are

```
EVRR number count
EVCR number count
```

The first will enter the number (number) into successive columns on the same row for (count) columns; the second will do the same down a column. For example, if the current location is [1,1], the command

```
EVCR 100 5
```

will enter the value 100 into locations [1,1] through [5,1].

EF - Enter Formula

The enter formula command is used to enter formulas into the array. It has two forms, single and multiple. The single form is

```
EF formula
```

which will enter the (formula) into the formula table. It will set the value in the current location to 0, unless a "SC A" has been done (see the set command), in which case the formula will be evaluated and the result of the evaluation put into the current location.

The multiple form of the EF command is

```
EFM formula nrows ncolumns
```

where (nrows) and (ncolumns) are the number of successive rows and columns, respectively, to write the formula into. The array section which will get the formula may be thought of as a rectangle consisting of (nrows) rows and (ncolumns) columns, with the current location as the upper-left corner. For example, if the current location is [1,2], then the command

```
EFM 1.1*[.,.-1] 8 11
```

will write the formula "1.1*[.,.-1]" to 8 rows and 11 columns, or specifically, into locations [1,2] through [1,12], [2,2] through [2,12], and so on through [8,2] through [8,12].

F - the FORMAT command

The format command is used to choose the manner in which the data in the array will be displayed and printed. There are four forms of the command, but they differ only in the portion of the array that they affect. They are

F format
FR format
FC format
FA format

F means to format only the current location. FR means format the current row, FC means the current column. FA means format the entire array.

The (format) consists of zero or more individual format characters. These characters can be listed in any order, in a generally "free" fashion. Following are the various format characters.

$ - Dollar Sign. If $ is included, numbers will be printed with a leading dollar sign. For example, 123 will be displayed as $123.

, - Comma. If a comma is included, numbers will be printed with commas inserted every three digits to the left of the decimal point. That is, the value 123 would be unchanged, but the value 1234567 would be 1,234,567.

0-15 - Digits. Including a number in the range 0 through 15 will set the number of digits printed to the RIGHT of the decimal point. For example, the value 123 with a format of 4 would be printed as 123.0000.

% - Percent. The percent character indicates that the value is to be considered a percentage, and it will be printed with a percent sign following it. In addition, the number will be multiplied by 100 before being displayed. For example, the value .13 will be displayed as 13%.

R - Right Justify. For text, the R character causes right justification.

Each location in the array is either "formatted" or "unformatted". Typing a format command without any arguments (for example, "FC ") sets a location or locations to "unformatted". In this case, numbers and text will both be displayed left justified. Numbers will be displayed in a "general" format, meaning however necessary to express the value (For example, 100 will be 100, 3.14159 will be 3.14159).

The only format option for text is R, right justify. Its absence indicates left

justification.

For numbers, formatting is a little more complicated. ANY numeric format sets the location to "formatted". When a location is formatted, numbers are RIGHT justified. This right justification should not be confused with the R character for text.

Each time a F command is used, it overrides any previous F command. For example, if a "F 2" command is issued, the current location will be set to 2 places to the right of the decimal point, typically used for "dollars-and-cents" notation. If it is then desired to add the dollar sign, the command "F $" will NOT function as expected, since it cancels the effect of the "F 2" command. The proper command would be "F 2 $" since this combines the two commands. This brings up an important point: Leaving out the number-of-digits character in a format command is the same as using 0; that is, "F $" is identical to "F $ 0".

As was indicated above, the format characters are entered in a "free" fashion. Their order is unimportant. For example, the commands

```
F 2$,     F $,2     F ,$2     F  $ , 2     and  F  2  ,  $
```

are all identical. Format characters may be used in whatever combination desired, except that the combination "$%" will produce a meaningless figure; for example, 123.45 will be displayed as $12345%.

Since there is limited space available for format characters, the R and , characters are actually the same thing. That is, using a comma on text will right justify it, and using R on a number will insert commas. Since a location cannot contain both text and a number, this should not cause any problem.

H - the HELP command

The help command is used to access a screenful of assistance (commonly called a "help screen") for a particular command. There are actually three different forms of the help command.

```
H
H letter
?
```

Just typing H gives a help screen on typing commands, editing, and moving the cursor. Typing H followed by a letter gives a help screen on the command beginning with that letter. For example, HF gives help on the format command.

The help command reads the help screen from the file EPL.SYS on the currently-logged-in disk. That disk should not be confused with the currently-selected disk used for ExecuPlan! The logged-in disk is the disk that CP/M thinks is the current one. To be more specific, the disk that was in the prompt before ExecuPlan was executed. If it was A>, then the logged-in disk is drive A; if it was B>, then it was drive B, etc. If the file is not present on the disk, then the system will say "Help Unavailable".

The ? command is used to get a QUICK help screen. It tells how to get more help (via the H command) and gives a list of the command letters and their meanings. This screen is part of the program, not read in from the disk. Therefore, it is always available.

I - the INITIALIZE command

The initialize command is used to set everything back to the standard. Specifically, the command

Resets the array size to 20 x 20,
Resets the row and column titles back to 1 2 3 4 5 etc.,
Clears all main titles,
Clears the entire array,
Sets the current file to none,
and Resets the print blocks back to standard.

The format of the command is simply

```
I
```

It will display a warning message,

```
** Warning: Initialization erases ALL data - type Y to proceed -
```

at which point you may type Y to proceed with the initialization. Typing any other character (including <Return>) will cancel the command.

J - the JUMP command

The jump command is used as a quick way to move the screen around on the array, faster than using the arrow keys. There are three jump commands:

```
JB
JR row
JC column
```

The first, JB, simply jumps to the top-left corner of the array, which would be, [1,1] on an initialized array.

JC jumps to the specified column. That column will be the leftmost on the screen after execution.

JR jumps to the specified row. The row will be the top on the screen after the jump, unless the size of the array makes this impossible. If the jumped-to row is within 18 rows of the end, it will be somewhere in the middle of the screen.

K - the KILL command

The kill command is used to "kill", or remove, a row or column from the array. It does not, however, change the size of the array. Therefore, when it kills a specified row or column, it creates a new one at the end in order to keep the array the same size.

The format of the kill command is

```
    KR row new-row
    KC column new-column
```

where (row) or (column) is the row or column to kill, and (new-row) or (new-column) is the title to be assigned to the row or column created at the end of the array.

Example: If your array currently has 12 rows, numbered 1 through 12, and you execute the command "KR 6 13" then your resulting rows will be 1,2,3,4,5,7,8,9,10,11,12,13.

There is one VERY IMPORTANT thing to note about the kill command! Relative references in formulas which refer to or over the killed row or column will NOT be changed. In other words, they will be INCORRECT after the command is executed. In the example above, if you had a reference in row 7 which contained something like .-2, before you executed the kill command, that would have pointed to row 5; after the command, it will point to row 4.

L - the LIST command

The list command is used to produce a list on the printer of all of the formulas associated with the array. The list is printed in the order in which the formulas will be evaluated; that is, either in row-major or column-major order, depending on the current "SC R/C" setting. The format of the command is simply

```
L
```

The listing will have the heading at the top of each page

```
FILENAME      Formula list by row
                         (or column, if appropriate)
```

where FILENAME is the name of the file if one is assigned. The format of the listing is

```
[destination]      = formula
```

where [destination] is the destination of the formula, and (formula) is the text of the formula.

If during the listing it is desired to stop, typing <Esc> will cancel the command.

M - the MOVE command

The move command is used to move a row or column from one place in the array to another. The command format is

```
MR row dest-row
MC column dest-column
```

where (row) or (column) is the row or column to move, and (dest-row) or (dest-column) is the row or column to move it adjacent to.

Depending on which direction the row or column is moved, it will either be placed above/to the left of the destination, or below/to the right. Specifically: for a COLUMN, if it is being moved to the left, is will be placed to the left of the destination; if it is being moved to the right, it will be placed to the right of the destination. For a ROW, if it is being moved up, it will be placed above the destination; if it is being moved down, it will be placed below the destination.

For example, say you have the rows 1,2,3,4,5,6. If you execute the command "MR 5 3", the resulting sequence will be 1,2,5,3,4,6. If you had executed the command "MR 2 6", the resulting sequence would have been 1,3,4,5,6,2. This example is equally applicable to columns.

NOTE: Like the K command, the M command does not change relative references in formulas.

O - the OPEN command

The open command is used to open up a new row or column in the array. The command does NOT change the size of the array, therefore when a new row or column is created, the last row or column of the array is removed, and its contents lost.

The format of the open command is

```
OR ref-row row
OC ref-column column
```

where (row) or (column) is the title of the new row or column to create, and (ref-row) or (ref-column) is where to put it. If COLUMN, the new column will be to the LEFT of the ref-column; if ROW, the new row will be ABOVE the ref-row.

Example: If there are currently 10 rows, numbered 1 through 10, then executing the command "OR 7 NEW" will result in rows 1,2,3,4,5,6,NEW,7,8,9 with row 10 being lost.

NOTE: Like the K and M commands, the O command does not change relative references within formulas.

P - the PRINT command

The print command is used to print the array, or to cause what would be printed to be written into a disk file for editing with Scope or Memorite. Like the disk command, the print command is actually an entire command mode. It is invoked simply with

```
        P
```

Print Command Mode

When print mode is entered, the screen will be erased and replaced with what is called the "print screen". The screen is divided into 5 "blocks", each one controlling certain aspects of what will be printed.

Block 0 - Main titles. This block is used to select how the main titles will be printed. For each of the four main titles, which will be displayed, the choice may be made whether to R - right justify, L - left justify, C - center, or X - not to print at all.

Block 1 - Print bounds. This is used to select the portion of the array to be printed. Specifically, the starting row, starting column, ending row, and ending column are specified. By proper manipulating of these bounds, an array much larger than a piece of paper can be printed on several sheets, then the sheets rearranged to form a large sheet.

Block 2 - Paper size. This informs the system of the size of paper being used, in terms of number of characters per line and number of lines per page. The width is used only for centering the titles, but the length tells the system the maximum number of lines to print on one page before skipping to the next page.

Block 3 - Row/column titles. This block allows the user to choose whether or not to have the system print the row and column titles on the report.

Block 4 - Invisible. This allows the user to set rows and columns to "invisible", meaning that they will NOT be printed, even if they are within the print bounds selected by block 1. This is most often used to prevent the printing of some type of intermediate result column. There is also a provision for overriding the invisible function, that is, to go ahead and print the invisible rows and columns.

Print Commands

In addition to the blocks, the bottom of the screen will list the available commands. Following are the commands, and how to use the blocks.

P - Print the array. *Typing P will cause the array to be printed, using the settings of the blocks to define the bounds, titles, etc. If during printing you wish to stop, type <Esc>.*

D - Disk. *Typing D is just like P, only instead of printing the array, the data will be written to a disk file. The format of the data will, however, be identical to when it is printed.*

When D is typed, the message

Please enter file name:

will be displayed. Type the name of the file that you wish to write the data to, followed by <Return>. The file will be assumed to have .MEM as the extension, and must not already exist. If it does, an error will be displayed and the command cancelled. The current disk will be used for the file; to use a different disk, first select it with the disk command from the main system.

If when you are prompted for the file, you decide not to execute the command, simply type <Return> without typing the file name, and the command will be cancelled.

F - Formfeed printer. *Typing F will simply cause a formfeed character to be sent to the printer. This normally has the effect of rolling the paper up to the top of the next page.*

0 - Edit block 0. *Typing 0 will allow you to change the information in block 0. Note that only the justification character can be changed at this time. To change the text of the title, the TMx command must be used when under the main system.*

When 0 is typed, the justification character of the first main title will be highlighted. At this point, you have several options:

Type <down arrow> - this will move the highlight down to the next title. If you are already at the bottom (fourth) title, the highlight will be moved back to the first one. The justification character for the current title will not be changed.

Type <up arrow> - this is the opposite of <down arrow>. The highlight will be moved up to the previous title. If you are already at the top, it will be moved down the the bottom one. The justification character will not be changed.

Type <Esc> - this will leave the current character unchanged, and return to print command mode.

Type <Return> - this is identical to <down arrow>, except that if you are at the bottom title, it will stop editing block 0 and return to print command mode, similarly to <Esc> above.

Type a justification character - typing either L, R, C, or X will set the justification character for the current title to whatever is typed. The highlight will not be moved, so if the wrong thing is typed, you may simply retype the correct character.

Anything other than the above characters will simply be ignored.

1 - Edit block 1. Typing 1 will allow you to edit the information contained in block 1. When 1 is typed, a cursor will appear a little to the right of the first line in block 1. At this point, you are in a mode similar to block 0, but a little different. Essentially, while block 0 is an "instant" block, meaning that when you type a character, it immediately replaces the previous character, block 1 is an "updated" block, meaning that the new information appears to the right of the old information and is edited by itself, and only replaces the old information when you leave block 1 and return to the print command mode.

Of course, the data you type is not a justification character. Instead, the appropriate title is typed. The first time print command mode is entered, the bounds are set to the size of the entire array.

When typing the title, up to eight characters may be typed, terminated by either <Return> or <Esc> (the difference is explained below). In the process of typing the title, <Bs> may be typed to back up one character.

Additionally, there are two special characters allowed, if typed as the first character. Typing <Control-F> will display the "first" row or column of the array; typing <Control-L> will display the "last" row or column. This could be useful if you wish to, say, start the printing at the first row, but you're not sure what the title of the first row is.

The same editing characters are available for block 1 as for block 0 (<down arrow>, <up arrow>, <Esc>, <Return>), but the way they work is slightly different. When by some method the cursor comes to be on a line, any previously-typed data on that line is erased. To change a single item without having the cursor move down the next line (and consequently erase something that you might have typed there), <Esc> may be typed instead of <Return>, which will immediately return to print command mode.

When print command mode is reentered, the information in block 1 will be updated based on the new information typed. If a title typed does not exist, it will be highlighted and an error displayed. Any line which contains an error will not be updated.

Although as explained this probably seems very complicated, it is virtually self-explanatory when actually done.

2 - Edit block 2. *Typing 2 allows the user to modify the information under block 2, the paper size block. All aspects of block 2 are identical to block 1, except that instead of typing row or column titles, you type decimal numbers. For page width, you should type the number of characters per line. This is only used for centering the main titles. For page length, you should type the maximum number of lines you wish printed on a page. On a standard 11" page, a length of 56 lines allows reasonable margins on the top and bottom. For both length and width, the system will accept values in the range of 40 through 255.*

3 - Edit block 3. *Typing 3 allows the user to select whether row or column titles are to be printed as part of the report. If the appropriate line is Y, the titles will be printed; if N, they won't. Block 3 is similar to block 0, except that the items are only updated when print command mode is returned to.*

4 - Edit block 4. *Block 4 is used to determine which rows or columns, if any, are to be considered "invisible", meaning that they will not be printed.*

Block 4 is perhaps the most confusing block, because it includes two individual fields for each line. The first is the row/column indicator, one of the characters R or C. The second is the title of the row or column.

When block 4 in entered, a cursor will appear on the top line of block 4. At this point you have several options.

> *The <up arrow>, <down arrow>, and <Esc> keys function the same as block 0. The exception is that when you are on the bottom (tenth) line, the down arrow moves to the "Print Inv? " field. From there the <down arrow> will move back to the first line. The <up arrow> moves in the same manner, only up instead of down.*
>
> *Typing R or C will set the row/column indicator for the current line to whatever is typed.*
>
> *Typing <Space> will turn OFF the line; that is, when <Return> is typed after the <Space>, it will also remove the title on the current line.*
>
> *<Return> is typed to allow movement to the title field. For example, to*

set a line to Row 1, you would type "R" <Return> followed by "1" as explained below. If the row/column indicator already contained an R, you could simply type the <Return> and then proceed with the "1".

If <Return> was typed, the cursor will jump three spaces to the right. At this point the system is awaiting a title. Type the row or column title that you wish to set to invisible. If the title is terminated with <Return>, the cursor will proceed down to the next line. If it is terminated with <Esc>, block 4 will be updated and the print command mode will be returned to.

When the cursor is moved to the "Print Inv? " field, you may type Y or N, in the same manner as block 3. If N is typed, the invisible function will work, that is, the rows and columns indicated will not be printed. If Y is typed, the function will be effectively overridden, that is, the rows and columns listed will be printed anyway.

When print command mode is returned to, the rows and columns in block 4 are looked up. If any of them are not found, they will be highlighted and an error displayed. However, the information will be left in the block, and when the array is printed, the invalid entries will be ignored.

Characteristics of print command mode.

The information in the print blocks remains as set until changed. In addition, all of the information is saved with the file, so when a file is loaded, all of the information in the print blocks will be the same as when the file was saved.

Q - the QUIT command

The quit command is used to return to CP/M. There are three forms of the quit command:

```
Q
QY
QN
```

If QY is typed, the array will automatically be updated; if no current file exists, you will be asked for one. If QN is typed, the array will not be updated.

If just Q is typed, you will be prompted with

```
Exiting - Type Y to update -
```

If at this point you type Y, the file will be updated and the program exited. If you type N, the file will not be updated, and the program exited. If you type <Esc>, the system will cancel the command altogether. Anything else will be ignored.

R - the ROUND command

The round command is used to change the precision of numbers in the array. Specifically, all numbers in the array that are NOT calculated as the result of a formula will be changed to match their representation on the screen. For example, if the value 1.469 is in a certain location, but the format is 2, then the number is being displayed as 1.47. Therefore, when the round command is executed, the number will actually be changed to 1.47.

The round command is executed simply by typing

```
R
```

at which time the system will warn

```
** Caution: Data will be changed - type Y to proceed -
```

If you type Y, the action will be carried out. Typing anything else will cancel the command without having any data changed.

S - the SET command

The set command is used to set or change certain aspects of the system. There are four set commands; set size, calculation, printer, and disk. These are explained separately.

The Set Size Command

The set size command is used to change the size of the array. The form of the command is

```
SS nrows ncolumns
```

where (nrows) is the number of rows to make the array, and (ncolumns) is the number of columns to make the array. When the system is initialized, the size is 20 x 20. To change to a 9 x 13 array, for example, the command would be

```
SS 9 13
```

If the number of rows or columns is being decreased, the message

```
** WARNING - Some rows containing data may be lost - type Y to
proceed -
```

will appear. If the number of columns is being decreased, the message will indicate columns instead of rows, of course. Typing Y will cause the action to be carried out; anything else will cancel the action. If you are decreasing both rows and columns, you will get both messages, one at a time. Responding Y to rows but something else to columns will result is the number of rows decreasing, but the number of columns remaining the same.

If is is desired to only change one dimension, the value 0 may be put in the other. For example, to change the array from 20 rows to 30, but not affect the number of columns, the command would be

```
SS 30 0
```

NOTE: There is a substantial amount of logic involved when shrinking the array, particularly when columns are involved. If an array with many rows has the number of columns lessened, several moments (read a minute or so) could elapse before the system completes the command. Therefore, do not fear system failure. SUGGESTION: When changing the size of the array, the rows are changed first. Therefore, say you're changing the 20 x 20 array to 10 x 100. The system will first create 100 rows, then shrink to 10 columns.

This will take quite a while, since the system is removing 10 columns of 100 rows each! To make the operation faster, FIRST change the number of columns (SS 0 10) THEN change the number of rows (SS 100 0). Using this method, only about 3 seconds will be used, a substantial increase in speed. If you are decreasing the number of rows and increasing the number of columns, FIRST decrease the rows, THEN increase the columns. The object is to make the number of rows as small as possible when creating or removing columns.

The Set Calculation Command

The set calc command is used to change the order and frequency of when the formulas are evaluated.

Formulas may be evaluated in either "row-major" or "column-major" order. If they are evaluated in row-major order, that means that they will be evaluated across each row, then down to the next row and across it, etc. In other words, the same way you read. In column-major order, they are evaluated down each column, then moved over to the next column and down it, etc.

The frequency may be set to either "manual" or "automatic". In manual mode, the formulas are only evaluated when the <Tab> key is pressed. In automatic mode, they are evaluated whenever a value or formula is entered.

The commands accepts the letters R, C, A, and M to represent each of the above options. Obviously, only two may be entered at once, but they can be entered in any order. For example, to set the calc order to column-major, and the frequency to automatic, the command would be

```
    SC C A          or          SC A C
```

To change only, say, the calculation order, this time to manual, the command would be

```
    SC M
```

While it will not produce an error to enter both R and C or both A and M at the same time, only the last one on the line will be used.

The Set Printer Command

The set printer command is used to send special characters to the printer. This should seldom be used, but is provided for special cases. The most common use would be so set a printer for some special mode.

The format of the command is

```
      SP byte byte byte etc.
```

where (byte) is a decimal number. As many or few bytes may be sent as needed. For the exact bytes required for particular printers, you will have to refer to the appropriate printer's manual.

For an example on what this might be used for, consider the Vector Matrix printer. If you typed the command

```
      SP 27 80
```

which corresponds to "ESC P" (the printer manual explains this), the printer would switch between 132 and 80 characters per line.

The Set Disk Command

The set disk command is used to reset the disk system in case you wish to change floppies. CP/M will normally not allow you to do this; it you do, you will get a BDOS read-only error, which can't be recovered from. Therefore, if you wish to change a floppy disk, put the new diskette into the drive and type

```
      SD
```

which will tell CP/M about the new disk and allow you to access it.

Foreword

Welcome to the world of ExecuPlan!

This reference guide is intended to provide all of the reference information needed to help you to use ExecuPlan to its fullest capability. Sections are provided covering how to bring up the system, read the screen, type and edit commands, enter data, and so forth. Complete explanations of all commands are provided. A section covers all of the errors that might occur and what they mean. A complete section is devoted to the math capabilities, entering formulas, how formulas are evaluated, functions, and so forth. Finally, some helpful information is provided for special circumstances, and the appendix has some annotated sample screens.

This guide is actually not intended for the first-time user. The accompanying guide, ExecuPlan Primer, is the most basic manual. It covers beginning concepts and moves step-by-step to more complex models.

It is recommended that first the primer be read and understood, then this reference guide read and understood. Once the concepts are understood, the reference guide will then be useful for looking up information as needed. This guide does cover most topics in more detail than the primer, but they are explained in a more direct manner, assuming that the reader has some familiarity with the system.

It cannot be stressed enough that you should READ this guide. There will be countless occasions where you aren't sure about something, or don't understand why what happens happens, or something else along that line. If this is the case, get out this reference guide, or the primer if necessary, and look it up. Almost always the information that you need will be found in moments.

To briefly preface the sections of this guide: Section 1 contains general information, such as bringing the system up, understanding the array, reading the screen, understanding titles and references, and typing and editing commands. Section 2 gives detailed explanations of all of the commands. Section 3 covers all errors. Section 4 explains the math package, how formulas are evaluated, how operators and functions work, and other math-related information. Section 5 just has some miscellaneous information.

Again, congratulations on your acquisition of ExecuPlan, and we hope that it will be beneficial to you in whatever applications you have in mind.

Acknowledgment

The ExecuPlan program and this reference guide were completely designed and written by Neale E. Brassell exclusively for Vector Graphic Inc.

The program was developed on a Vector Graphic VIP computer system with two drives, a Sprint 3 printer, and a Teletype model 40 line printer, utilizing the SCOPE editor, ZSM assembler, and RAID debugging system. The documentation was written with the MEMORITE III word processing system

The source program consists of nearly a quarter million bytes of source code, contained in fifteen program modules totaling about 170 pages. After assembly, the object program is about 20K long. Additionally, the help screens occupy another 54K data file.

The floating point math routines and the single-argument functions are derived from Microsoft BASIC version 5.1 which is licensed from Microsoft Inc. The balance of the math routines (parser, expression evaluator, and multi-argument functions) and the balance of the program are all original code.

The author would like to thank the software staff at Vector Graphic for their support and suggestions on the development of ExecuPlan. I am particularly grateful to Chris Cory, who contributed many significant ideas, was instrumental in the debugging of the program, and who wrote the primer and developed all of the sample applications that come with ExecuPlan.

STARTING UP

ExecuPlan is a CP/M-based program, supplied on a CP/M diskette. To execute the program, insert the diskette into the drive and type

```
A>EPL          (You type what's underlined)
```

After a few seconds the screen will display a large banner identifying ExecuPlan, and giving the revision and copyright notice. Type any character on the keyboard. The banner will be erased and an initialized array will be shown on the screen. You're up and running!

At the time EPL is loaded, you may also tell the system to automatically load or create a file on the disk. For example, to run ExecuPlan and load the file BUDGET81, you would type

```
A>EPL BUDGET81
```

After the banner is erased, the system would display the message

```
Loading BUDGET81
```

and proceed to load the program. After the program is loaded, the array containing the data will be displayed.

One other capability when EPL is executed is the ability to create a new file. The command is typed the same as to load a file. If the file does not exist, then the system will display the message

```
Creating BUDGET81
```

and proceed to create the file. The array displayed will be blank, but the file specified will be created, and when the array is updated it will be written into that file.

THE ARRAY

Concept

ExecuPlan is built around the concept of an electronic array. This array contains a number of rows and columns. Information may be put into each intersection of a row and a column. Then, information put into the array can be used to compute other information, which is also put into the array. Finally, the array may be printed in a report fashion.

Each intersection of a row and column is called a "coordinate" or "location". If, for example, the array contains 20 rows and 20 columns, then there would be 400 locations in the array.

References

Given all of these locations, a method must be used to refer to each of them. This is called a "reference", and has the following format:

    [Row,Column]

This is the scheme used throughout ExecuPlan to reference locations. For example, a reference to row 10 and column 8 would be

    [10,8]

Each row and column has a "title", which is used in a reference. In the preceding example, "10" is the row title, and "8" is the column title. Titles do not have to be numbers, however. A reference might be

    [INCOME,JANUARY]

which would refer to row INCOME and column JANUARY. A command is provided for changing the titles of rows and columns.

In the next section, which talks about the screen, the idea of a "current" location will be brought up. For the purpose of references, suffice it for the moment to say that there is a location in the array that is considered to be the current one. From this comes the idea of "relative references". A relative reference is one where rather than specifying the title of a row or column, the distance away from the current location is specified. The character period (.) is used to mean the current location. For example, the reference

    [.,.]

means "the current row and the current column", that is, the current location. The reference

```
[.,.-1]
```

means "the current row, and the column that is 1 to the left of the current". If the current location is [4,4], then the aforementioned reference would point to [4,3].

Following are some examples of relative references.

[.,.-5]	Current row, current column - 5
[7,.]	Row "7", current column
[.-1,JAN]	Current row - 1, column JAN

THE SCREEN

This section explains how the screen looks and what all of the things on the screen mean. It is suggested that you look at the picture of a sample screen in the appendix while reading this section.

Concept

Conceptually, the screen is a movable "window" over the array. When the system is initialized, that window is over the top-left corner of the array. Using the arrow keys, the window can be moved over any section of the array. The portion of the array that appears under the window is always 18 rows (unless there are less than that many) by however many columns will fit. Initially, columns have a width of 16 characters, so 4 will fit on the screen. Columns may have almost any width, however, so the number of columns displayed will vary from 1 to 34. If a column's full width will not fit on the screen, it will not be displayed.

Standard Information

At this point, let's take a look at the screen layout.

FIRST LINE: The first line (which is highlighted) is the "status" line, because it tells you the status of the system. Four things are displayed on the status line. 1: The current location. This is the location that the cursor is currently on (see below for additional explanation of the cursor). 2: Contents of the current location. This is simply whatever is in the current location, displayed according to the format of that location. 3: The current file, if any. This area might be blank. If it is not, then it contains the name of the current file, which is used if the file is updated. 4: The amount of space still available in memory. Initially, this is about 30,000 characters.

SECOND LINE: The second line is the "formula" line. If the current location contains a formula, that formula will be displayed on this line. If there is no formula associated with the current location, this line will be blank.

THIRD LINE: This is the "main title" line. Initially, this line contains a line of dashes. When main title #1 is set (see T command), then that title is centered in the field of dashes and displayed on this line.

BOTTOM LINE: This is the "command" line, where commands are typed. This will be explained in detail in the next section.

Data Area

The area between the main title line and the row of dashes above the command line is the data area of the screen. It, in turn, actually has two sections, the titles and the data.

The titles are displayed across the top of the data area and down the left side. The COLUMN titles are across the top, and the ROW titles down the left side.

The data itself fills the remainder of the area. If there is no data in the system, this area will be blank.

It is somewhat difficult to explain, but very easy to understand, the layout of the data in the data area. Essentially, each intersection of a row and a column (each LOCATION) can contain an item of data. If you visually trace a horizontal line across from the row title, and visually trace a vertical line down from the column title, then where those lines intersect will be where the data contained in that location will be displayed. For an easy example, the data contained in [1,1] will be the top-left corner; data contained in [18,4] (on an initialized system) will be in the bottom-right corner.

The Cursor

The "cursor" is the name given to the large white rectangle that is somewhere in the data area. The position of that cursor, taken as a reference, is called the "current location", a term that has been and will be used OFTEN. If the cursor in in the top-left corner (again, of an initialized system), then it is in location [1,1], and therefore [1,1] is said to be the current location. When the arrow keys are used (or a couple of other keys, all explained later), the cursor moves. If it is in the top-left corner, then pressing the <down arrow> key will move the cursor to the second row. Pressing it again will move to the third row, and so forth.

Moving the Cursor

As hinted at above, the cursor is moved around on the screen with the <arrow> keys. Here, we'll go into more detail on this.

The arrow keys move the cursor in the appropriate direction. When the cursor reaches the edge of the screen, then instead of moving the cursor, the whole screen is moved. More specifically, the window that the screen represents moves with respect to the entire array. If column 4 is the rightmost column on the screen and the <right arrow> key is pressed, the first column on the screen will be shifted off and the next column to the right will be shifted on. Then, if the <left arrow> key is pressed, only the cursor will move. Repeated pressings will move the cursor to the left until it

is on column 2, which at this point would be the leftmost. The next pressing would shift the screen left and move column 1 onto the screen.

When the cursor is at the extreme point in any direction and that arrow key is pressed, the system will simply ignore it.

In addition to the regular arrow keys, the <up arrow> and <down arrow> keys may be shifted to move the screen 18 rows (a screenful) at a time. For example, if the cursor is on row 1 and <shift down arrow> is pressed, the cursor will be on row 18, which would be the beginning of the second screenful of rows. The exception to this is if that action would result in less than 18 rows being displayed. In that case, the screen will be shifted as far as possible, but not so far as to result in less than 18 rows being displayed.

Finally, there are three other cursor-movement keys. Pressing <Lf> will move the cursor to the leftmost column of the current row, as if the <left arrow> key were pressed repeatedly. Typing <Control-T> will move the cursor to the top of the array and put it at the top line of the data display also. <Control-E> will move the cursor to the end of the array, and put the cursor on the bottom line also.

TYPING COMMANDS

In order to really DO anything with ExecuPlan, you must give it "commands", which are instructions telling it what to do.

There are two directions taken when it comes to giving a computer commands. One is the "language" method. In this method, you create a list of instructions, feed it to the computer, and get back the results. This method is interesting, but it really just amounts to simplified computer programming. This is not an acceptable method of getting results to someone who is not a programmer.

The other method is known as "interaction". In this method, you give the computer AN INSTRUCTION. The computer follows that instruction and displays the result, if any. You then give the computer another instruction, and it carrys that out. In other words, the computer interacts with the user. It is this method that is used by ExecuPlan.

Recall from the last section that the bottom line is the "command" line. It is on this line that commands are typed.

In the very bottom left corner of the screen are two things. These are the "command prompt" and the "command cursor". Before you begin to type a command, these look like ><, only highlighted. The left character, >, is the prompt. The right character, <, is the command cursor, NOT TO BE CONFUSED WITH THE CURSOR IN THE ARRAY. The command cursor tells you where on the line you are at. When you type a character, the character appears on the screen where the command cursor WAS, then the cursor reappears one character to the right.

Notice in the last sentence of that paragraph that the command cursor was refered to simply as the cursor. This might seem confusing, since the white rectangle on the screen in the current location is called the cursor. Well, they are SO conceptually different that you will have no difficulty ascertaining which one is meant when you see the word "cursor".

Characteristics of Command Mode

Yes, when the system is sitting waiting for you to type a command, that is called "command mode". When the system is in command mode, you may type commands (seems reasonable, doesn't it?).

As mentioned above, when a character is typed, it appears on the screen where the cursor was, then the cursor reappears one character to the right. This should not be new, since virtually all software operates in this manner.

At any time during the typing of a command, all of the capabilities of moving the screen (arrow keys, etc.) are available.

When a command has been typed satisfactorily and you wish the computer to carry out that command, the <Return> key must be pressed. As soon as that key is pressed, the system STOPS waiting for you to type, and begins to execute the command.

In addition to normal letters and numbers used when typing commands, certain other "special" characters can be used, which will be explained in the following paragraphs.

Finally, and for lack of a better place to explain it, it should be noted that upper case and lower case letters are fully interchangeable. That is, typing "EVC" is the same as "evc" or "Evc" or "evC" or whatever. Actually, this is true not just in command mode, but everywhere in the system. EXCEPT: Titles do not allow interchanging of cases. That is, "Jan" and "JAN" are different titles.

Special Characters

While typing commands, there are several special characters available. They are of two types. The first is editing, which means that they allow you to "edit", or change, what you've typed. The second is "inserting", meaning that you can "insert" certain things from the system without having to type them.

Each special character will be explained here.

***<Bs>** - Typing the <Bs> key (Backspace) will cause the cursor to "back up", effectively erasing the last character that was typed. For example, if you typed "ET HELLI" then realized that you meant to type "O", not "I", then you could hit the <Bs> key which would back up and erase the "I". Then, you'd type "O" and continue.*

***<Del>** - Typing <Del> simply erases the entire command that you've typed. It is equivalent to hitting <Bs> repeatedly until every character was erased.*

***<Control-K>** - For users with extensive Memorite experience, <Control-K> is the same as <Del>. (Memorite uses <Control-K> to erase the current line.)*

***<Esc>** - Hitting the <Esc> key enters "command edit mode". In this mode, more extensive editing of the line is possible. Command edit mode will be explained in following paragraphs.*

All of the above characters are of the editing type. The remainder are the inserting type.

<Control-A> - This will take the current location, make it a reference, and insert it into the command line. If the cursor is at [1,1], for example, then typing <control-A> would insert the characters "[1,1]" into the command line, as if you had typed it. This is useful for grabbing locations to be used in a formula. The rationale behind the character <Control-A> is that it inserts the location that the cursor is AT.

<Control-F> - This will insert the current FORMULA into the command line. Look at the second line on the screen, the formula line. Whatever is there will be inserted into the command if <control-F> is typed. If there is no formula associated with the current location, then nothing will be inserted.

<Control-C> - This will insert the CONTENTS of the current location into the command line. The data inserted will be exactly as it is displayed.

<Control-L> - This will simply insert the entire LAST command typed into the command line. This can be useful for repeating an operation, such as entering data, for several different locations. It also is useful, particularly, when a lengthy command is typed which contains an error. Rather than retyping the entire line, you can simply type <control-L> then use the editing commands, below, on the line.

Keep in mind that the above characters can be combined with the screen movement characters. For example, say you'd like to insert a formula into the command line. Not the current formula, but the formula that is associated with a particular location. You can simply move the cursor to that location, type <control-F>, then move the cursor back to where you want it!

Command Edit Mode

As mentioned above, typing <Esc> enters command edit mode. When in command edit mode, none of the above-mentioned characters work, and none of the screen-movement characters work. This mode is used when you've typed a long command and need to change it, but just don't want to retype the whole thing.

Command edit mode, or simply edit mode, is indicated by having a character of the command highlighted. At first, this will be the rightmost character of the command. The highlighted spot is itself like a cursor, insofar as that is the spot where whatever is done will take place.

While in edit mode, typing characters is just like normal command mode, except that whatever character is typed will replace the character under it.

Certain special characters exist in edit mode, and they are as follows.

<Left-arrow> - this will move the highlight one character to the left.

<Right-arrow> - this will move the highlight one character to the right. It will not move any further than the cursor.

<Lf> - this will move the highlight to the first character of the line.

<Control-V> - this will enter "insert mode". At this point, when normal characters are typed, they will not replace the character under the highlight. Instead, the remainder of the line will be shifted to the right and the character typed will be inserted. Typing <control-V> a second time will leave insert mode.

<Control-D> - this will delete the character under the highlight. The remainder of the line will be shifted to the left.

<Esc> - this will leave edit mode and return to the normal command mode.

<Return> - this is like typing <Esc> then typing <Return>. That is, it will leave edit mode, but then proceed as if you had typed <Return> in command mode and execute the command.

Forced References

Throughout this guide, the expression "current location" will be used. As previously explained, this means the location where the cursor is at.

There is a way to make the current location somewhere else, if desired. For example, the C command clears the current location. You might wish to clear location [20,5], but the current location is [1,1]. You could just move the cursor to the desired location and then clear it, but another method is provided.

This method is called "forcing" the current location. That is, you can make the system think that somewhere else is the current location. This is done by simply typing the reference before the command. For example, you could type

```
[5,20] C
```

Which would force [5,20] as the current location, then execute the C command which clears the current location, which would be [5,20]. Another example:

```
[INCOME,JAN] EVC 1000 1500 2000
```

This would force location [INCOME,JAN] and then execute the EVC command, *which will put the values into successive locations, starting at the current location. Of course, the current location would be what it was forced to be.*

It should be noted, though, that the forced reference is only applicable to the command with which it is typed. After execution of the command is complete, the spot where the cursor is at will again be the current location.

In general, anywhere in this guide where any of the expressions "current location", "current row", or "current column" are used, they refer to the current as defined, unless a forced reference is used, in which case they refer to the forced current location.

Manual Recalculation

The system has two modes for calculations, automatic and manual, which are selected with the SC command (which is explained in another section). When in automatic mode, entering any formula or value will cause the system to reevaluate the array. In manual mode, this will not happen. To cause a recalculation when in manual mode, press the <Tab> key. You must do this as the first character on the command line - if you are in the middle of a command, the system will ignore you. The message -Recalculating- will appear in the bottom right corner of the screen while the system is doing the calculations. When it is done, you will be returned to command mode.

Note that it is possible to have circular references such that it takes two or more recalculations before the array is fully evaluated. You can simply keep pressing the <Tab> key as often as needed.

Foreword

Welcome to the world of ExecuPlan!

This reference guide is intended to provide all of the reference information needed to help you to use ExecuPlan to its fullest capability. Sections are provided covering how to bring up the system, read the screen, type and edit commands, enter data, and so forth. Complete explanations of all commands are provided. A section covers all of the errors that might occur and what they mean. A complete section is devoted to the math capabilities, entering formulas, how formulas are evaluated, functions, and so forth. Finally, some helpful information is provided for special circumstances, and the appendix has some annotated sample screens.

This guide is actually not intended for the first-time user. The accompanying guide, ExecuPlan Primer, is the most basic manual. It covers beginning concepts and moves step-by-step to more complex models.

It is recommended that first the primer be read and understood, then this reference guide read and understood. Once the concepts are understood, the reference guide will then be useful for looking up information as needed. This guide does cover most topics in more detail than the primer, but they are explained in a more direct manner, assuming that the reader has some familiarity with the system.

It cannot be stressed enough that you should READ this guide. There will be countless occasions where you aren't sure about something, or don't understand why what happens happens, or something else along that line. If this is the case, get out this reference guide, or the primer if necessary, and look it up. Almost always the information that you need will be found in moments.

To briefly preface the sections of this guide: Section 1 contains general information, such as bringing the system up, understanding the array, reading the screen, understanding titles and references, and typing and editing commands. Section 2 gives detailed explanations of all of the commands. Section 3 covers all errors. Section 4 explains the math package, how formulas are evaluated, how operators and functions work, and other math-related information. Section 5 just has some miscellaneous information.

Again, congratulations on your acquisition of ExecuPlan, and we hope that it will be beneficial to you in whatever applications you have in mind.

Acknowledgment

The ExecuPlan program and this reference guide were completely designed and written by Neale E. Brassell exclusively for Vector Graphic Inc.

The program was developed on a Vector Graphic VIP computer system with two drives, a Sprint 3 printer, and a Teletype model 40 line printer, utilizing the SCOPE editor, ZSM assembler, and RAID debugging system. The documentation was written with the MEMORITE III word processing system

The source program consists of nearly a quarter million bytes of source code, contained in fifteen program modules totaling about 170 pages. After assembly, the object program is about 20K long. Additionally, the help screens occupy another 54K data file.

The floating point math routines and the single-argument functions are derived from Microsoft BASIC version 5.1 which is licensed from Microsoft Inc. The balance of the math routines (parser, expression evaluator, and multi-argument functions) and the balance of the program are all original code.

The author would like to thank the software staff at Vector Graphic for their support and suggestions on the development of ExecuPlan. I am particularly grateful to Chris Cory, who contributed many significant ideas, was instrumental in the debugging of the program, and who wrote the primer and developed all of the sample applications that come with ExecuPlan.

STARTING UP

ExecuPlan is a CP/M-based program, supplied on a CP/M diskette. To execute the program, insert the diskette into the drive and type

```
A>EPL          (You type what's underlined)
```

After a few seconds the screen will display a large banner identifying ExecuPlan, and giving the revision and copyright notice. Type any character on the keyboard. The banner will be erased and an initialized array will be shown on the screen. You're up and running!

At the time EPL is loaded, you may also tell the system to automatically load or create a file on the disk. For example, to run ExecuPlan and load the file BUDGET81, you would type

```
A>EPL BUDGET81
```

After the banner is erased, the system would display the message

```
Loading BUDGET81
```

and proceed to load the program. After the program is loaded, the array containing the data will be displayed.

One other capability when EPL is executed is the ability to create a new file. The command is typed the same as to load a file. If the file does not exist, then the system will display the message

```
Creating BUDGET81
```

and proceed to create the file. The array displayed will be blank, but the file specified will be created, and when the array is updated it will be written into that file.

THE ARRAY

Concept

ExecuPlan is built around the concept of an electronic array. This array contains a number of rows and columns. Information may be put into each intersection of a row and a column. Then, information put into the array can be used to compute other information, which is also put into the array. Finally, the array may be printed in a report fashion.

Each intersection of a row and column is called a "coordinate" or "location". If, for example, the array contains 20 rows and 20 columns, then there would be 400 locations in the array.

References

Given all of these locations, a method must be used to refer to each of them. This is called a "reference", and has the following format:

[Row,Column]

This is the scheme used throughout ExecuPlan to reference locations. For example, a reference to row 10 and column 8 would be

[10,8]

Each row and column has a "title", which is used in a reference. In the preceding example, "10" is the row title, and "8" is the column title. Titles do not have to be numbers, however. A reference might be

[INCOME,JANUARY]

which would refer to row INCOME and column JANUARY. A command is provided for changing the titles of rows and columns.

In the next section, which talks about the screen, the idea of a "current" location will be brought up. For the purpose of references, suffice it for the moment to say that there is a location in the array that is considered to be the current one. From this comes the idea of "relative references". A relative reference is one where rather than specifying the title of a row or column, the distance away from the current location is specified. The character period (.) is used to mean the current location. For example, the reference

[.,.]

means "the current row and the current column", that is, the current location. The reference

[.,.-1]

means "the current row, and the column that is 1 to the left of the current". If the current location is [4,4], then the aforementioned reference would point to [4,3].

Following are some examples of relative references.

[.,.-5]	*Current row, current column - 5*
[7,.]	*Row "7", current column*
[.-1,JAN]	*Current row - 1, column JAN*

THE SCREEN

This section explains how the screen looks and what all of the things on the screen mean. It is suggested that you look at the picture of a sample screen in the appendix while reading this section.

Concept

Conceptually, the screen is a movable "window" over the array. When the system is initialized, that window is over the top-left corner of the array. Using the arrow keys, the window can be moved over any section of the array. The portion of the array that appears under the window is always 18 rows (unless there are less than that many) by however many columns will fit. Initially, columns have a width of 16 characters, so 4 will fit on the screen. Columns may have almost any width, however, so the number of columns displayed will vary from 1 to 34. If a column's full width will not fit on the screen, it will not be displayed.

Standard Information

At this point, let's take a look at the screen layout.

FIRST LINE: The first line (which is highlighted) is the "status" line, because it tells you the status of the system. Four things are displayed on the status line. 1: The current location. This is the location that the cursor is currently on (see below for additional explanation of the cursor). 2: Contents of the current location. This is simply whatever is in the current location, displayed according to the format of that location. 3: The current file, if any. This area might be blank. If it is not, then it contains the name of the current file, which is used if the file is updated. 4: The amount of space still available in memory. Initially, this is about 30,000 characters.

SECOND LINE: The second line is the "formula" line. If the current location contains a formula, that formula will be displayed on this line. If there is no formula associated with the current location, this line will be blank.

THIRD LINE: This is the "main title" line. Initially, this line contains a line of dashes. When main title #1 is set (see T command), then that title is centered in the field of dashes and displayed on this line.

BOTTOM LINE: This is the "command" line, where commands are typed. This will be explained in detail in the next section.

Data Area

The area between the main title line and the row of dashes above the command line is the data area of the screen. It, in turn, actually has two sections, the titles and the data.

The titles are displayed across the top of the data area and down the left side. The COLUMN titles are across the top, and the ROW titles down the left side.

The data itself fills the remainder of the area. If there is no data in the system, this area will be blank.

It is somewhat difficult to explain, but very easy to understand, the layout of the data in the data area. Essentially, each intersection of a row and a column (each LOCATION) can contain an item of data. If you visually trace a horizontal line across from the row title, and visually trace a vertical line down from the column title, then where those lines intersect will be where the data contained in that location will be displayed. For an easy example, the data contained in [1,1] will be the top-left corner; data contained in [18,4] (on an initialized system) will be in the bottom-right corner.

The Cursor

The "cursor" is the name given to the large white rectangle that is somewhere in the data area. The position of that cursor, taken as a reference, is called the "current location", a term that has been and will be used OFTEN. If the cursor in in the top-left corner (again, of an initialized system), then it is in location [1,1], and therefore [1,1] is said to be the current location. When the arrow keys are used (or a couple of other keys, all explained later), the cursor moves. If it is in the top-left corner, then pressing the <down arrow> key will move the cursor to the second row. Pressing it again will move to the third row, and so forth.

Moving the Cursor

As hinted at above, the cursor is moved around on the screen with the <arrow> keys. Here, we'll go into more detail on this.

The arrow keys move the cursor in the appropriate direction. When the cursor reaches the edge of the screen, then instead of moving the cursor, the whole screen is moved. More specifically, the window that the screen represents moves with respect to the entire array. If column 4 is the rightmost column on the screen and the <right arrow> key is pressed, the first column on the screen will be shifted off and the next column to the right will be shifted on. Then, if the <left arrow> key is pressed, only the cursor will move. Repeated pressings will move the cursor to the left until it

is on column 2, which at this point would be the leftmost. The next pressing would shift the screen left and move column 1 onto the screen.

When the cursor is at the extreme point in any direction and that arrow key is pressed, the system will simply ignore it.

In addition to the regular arrow keys, the <up arrow> and <down arrow> keys may be shifted to move the screen 18 rows (a screenful) at a time. For example, if the cursor is on row 1 and <shift down arrow> is pressed, the cursor will be on row 18, which would be the beginning of the second screenful of rows. The exception to this is if that action would result in less than 18 rows being displayed. In that case, the screen will be shifted as far as possible, but not so far as to result in less than 18 rows being displayed.

Finally, there are three other cursor-movement keys. Pressing <Lf> will move the cursor to the leftmost column of the current row, as if the <left arrow> key were pressed repeatedly. Typing <Control-T> will move the cursor to the top of the array and put it at the top line of the data display also. <Control-E> will move the cursor to the end of the array, and put the cursor on the bottom line also.

TYPING COMMANDS

In order to really DO anything with ExecuPlan, you must give it "commands", which are instructions telling it what to do.

There are two directions taken when it comes to giving a computer commands. One is the "language" method. In this method, you create a list of instructions, feed it to the computer, and get back the results. This method is interesting, but it really just amounts to simplified computer programming. This is not an acceptable method of getting results to someone who is not a programmer.

The other method is known as "interaction". In this method, you give the computer AN INSTRUCTION. The computer follows that instruction and displays the result, if any. You then give the computer another instruction, and it carrys that out. In other words, the computer interacts with the user. It is this method that is used by ExecuPlan.

Recall from the last section that the bottom line is the "command" line. It is on this line that commands are typed.

In the very bottom left corner of the screen are two things. These are the "command prompt" and the "command cursor". Before you begin to type a command, these look like ><, only highlighted. The left character, >, is the prompt. The right character, <, is the command cursor, NOT TO BE CONFUSED WITH THE CURSOR IN THE ARRAY. The command cursor tells you where on the line you are at. When you type a character, the character appears on the screen where the command cursor WAS, then the cursor reappears one character to the right.

Notice in the last sentence of that paragraph that the command cursor was refered to simply as the cursor. This might seem confusing, since the white rectangle on the screen in the current location is called the cursor. Well, they are SO conceptually different that you will have no difficulty ascertaining which one is meant when you see the word "cursor".

Characteristics of Command Mode

Yes, when the system is sitting waiting for you to type a command, that is called "command mode". When the system is in command mode, you may type commands (seems reasonable, doesn't it?).

As mentioned above, when a character is typed, it appears on the screen where the cursor was, then the cursor reappears one character to the right. This should not be new, since virtually all software operates in this manner.

At any time during the typing of a command, all of the capabilities of moving the screen (arrow keys, etc.) are available.

When a command has been typed satisfactorily and you wish the computer to carry out that command, the <Return> key must be pressed. As soon as that key is pressed, the system STOPS waiting for you to type, and begins to execute the command.

In addition to normal letters and numbers used when typing commands, certain other "special" characters can be used, which will be explained in the following paragraphs.

Finally, and for lack of a better place to explain it, it should be noted that upper case and lower case letters are fully interchangeable. That is, typing "EVC" is the same as "evc" or "Evc" or "evC" or whatever. Actually, this is true not just in command mode, but everywhere in the system. EXCEPT: Titles do not allow interchanging of cases. That is, "Jan" and "JAN" are different titles.

Special Characters

While typing commands, there are several special characters available. They are of two types. The first is editing, which means that they allow you to "edit", or change, what you've typed. The second is "inserting", meaning that you can "insert" certain things from the system without having to type them.

Each special character will be explained here.

<Bs> - Typing the <Bs> key (Backspace) will cause the cursor to "back up", effectively erasing the last character that was typed. For example, if you typed "ET HELLI" then realized that you meant to type "O", not "I", then you could hit the <Bs> key which would back up and erase the "I". Then, you'd type "O" and continue.

<Del> - Typing <Del> simply erases the entire command that you've typed. It is equivalent to hitting <Bs> repeatedly until every character was erased.

<Control-K> - For users with extensive Memorite experience, <Control-K> is the same as <Del>. (Memorite uses <Control-K> to erase the current line.)

<Esc> - Hitting the <Esc> key enters "command edit mode". In this mode, more extensive editing of the line is possible. Command edit mode will be explained in following paragraphs.

All of the above characters are of the editing type. The remainder are the inserting type.

<Control-A> - This will take the current location, make it a reference, and insert it into the command line. If the cursor is at [1,1], for example, then typing <control-A> would insert the characters "[1,1]" into the command line, as if you had typed it. This is useful for grabbing locations to be used in a formula. The rationale behind the character <Control-A> is that it inserts the location that the cursor is AT.

<Control-F> - This will insert the current FORMULA into the command line. Look at the second line on the screen, the formula line. Whatever is there will be inserted into the command if <control-F> is typed. If there is no formula associated with the current location, then nothing will be inserted.

<Control-C> - This will insert the CONTENTS of the current location into the command line. The data inserted will be exactly as it is displayed.

<Control-L> - This will simply insert the entire LAST command typed into the command line. This can be useful for repeating an operation, such as entering data, for several different locations. It also is useful, particularly, when a lengthy command is typed which contains an error. Rather than retyping the entire line, you can simply type <control-L> then use the editing commands, below, on the line.

Keep in mind that the above characters can be combined with the screen movement characters. For example, say you'd like to insert a formula into the command line. Not the current formula, but the formula that is associated with a particular location. You can simply move the cursor to that location, type <control-F>, then move the cursor back to where you want it!

Command Edit Mode

As mentioned above, typing <Esc> enters command edit mode. When in command edit mode, none of the above-mentioned characters work, and none of the screen-movement characters work. This mode is used when you've typed a long command and need to change it, but just don't want to retype the whole thing.

Command edit mode, or simply edit mode, is indicated by having a character of the command highlighted. At first, this will be the rightmost character of the command. The highlighted spot is itself like a cursor, insofar as that is the spot where whatever is done will take place.

While in edit mode, typing characters is just like normal command mode, except that whatever character is typed will replace the character under it.

Certain special characters exist in edit mode, and they are as follows.

<Left-arrow> - this will move the highlight one character to the left.

<Right-arrow> - this will move the highlight one character to the right. It will not move any further than the cursor.

<Lf> - this will move the highlight to the first character of the line.

<Control-V> - this will enter "insert mode". At this point, when normal characters are typed, they will not replace the character under the highlight. Instead, the remainder of the line will be shifted to the right and the character typed will be inserted. Typing <control-V> a second time will leave insert mode.

<Control-D> - this will delete the character under the highlight. The remainder of the line will be shifted to the left.

<Esc> - this will leave edit mode and return to the normal command mode.

<Return> - this is like typing <Esc> then typing <Return>. That is, it will leave edit mode, but then proceed as if you had typed <Return> in command mode and execute the command.

Forced References

Throughout this guide, the expression "current location" will be used. As previously explained, this means the location where the cursor is at.

There is a way to make the current location somewhere else, if desired. For example, the C command clears the current location. You might wish to clear location [20,5], but the current location is [1,1]. You could just move the cursor to the desired location and then clear it, but another method is provided.

This method is called "forcing" the current location. That is, you can make the system think that somewhere else is the current location. This is done by simply typing the reference before the command. For example, you could type

```
[5,20] C
```

Which would force [5,20] as the current location, then execute the C command which clears the current location, which would be [5,20]. Another example:

```
[INCOME,JAN] EVC 1000 1500 2000
```

This would force location [INCOME,JAN] and then execute the EVC command, which will put the values into successive locations, starting at the current location. Of course, the current location would be what it was forced to be.

It should be noted, though, that the forced reference is only applicable to the command with which it is typed. After execution of the command is complete, the spot where the cursor is at will again be the current location.

In general, anywhere in this guide where any of the expressions "current location", "current row", or "current column" are used, they refer to the current as defined, unless a forced reference is used, in which case they refer to the forced current location.

Manual Recalculation

The system has two modes for calculations, automatic and manual, which are selected with the SC command (which is explained in another section). When in automatic mode, entering any formula or value will cause the system to reevaluate the array. In manual mode, this will not happen. To cause a recalculation when in manual mode, press the <Tab> key. You must do this as the first character on the command line - if you are in the middle of a command, the system will ignore you. The message -Recalculating- will appear in the bottom right corner of the screen while the system is doing the calculations. When it is done, you will be returned to command mode.

Note that it is possible to have circular references such that it takes two or more recalculations before the array is fully evaluated. You can simply keep pressing the <Tab> key as often as needed.

Section 2

Commands

Complete detailed descriptions of all commands and how they are used.

C - the CLEAR command

The clear command is used to clear certain locations in the array of the data and/or formulas associated with them. The various forms are as follows.

C - Clear current location. The data and formula will be cleared from the current location.

CF - Clear current formula. If there is a formula associated with the current location, it will be erased. The data in the current location will be left unchanged.

CR - Clear row. The data and formulas for all locations in the current row will be erased. Alternate form:

    CR row

In this case, (row) specified will be cleared.

CC - Clear column. The data and formulas for all locations in the current column will be erased. Alternate form:

    CC column

In this case, (column) specified will be cleared.

CA - Clear all. All data and formulas in the array will be erased.

D - the DISK command

The disk command is used to access disk directories to load, save, or erase files. It is also used to select the current disk to be used for subsequent disk operations.

D - Disk commands. Disk command mode will be entered, using the currently-selected disk.

D x - Disk select/commands. Disk x will be selected, then disk command mode will be entered. The value x must generally be in the range A to P, but specifically must be a valid drive on the system being used at the time.

Disk Command Mode

When disk command mode is entered, the directory of EPL files on the selected disk will be displayed at the top of the screen; the current disk and a list of available commands will be displayed at the bottom. Furthermore, the first file name will be highlighted.

At this point, typing any of the four arrow keys will move the highlight in the appropriate direction. Once the highlight is moved to the desired file, the disk commands below can be used. All disk commands reference the currently-highlighted file.

Disk Commands

L - Load a file. The highlighted file will be loaded into the array. Note that this erases the current array; therefore, if it is desired to save the current array, it should be updated or saved before executing the load command.

S - Save a file. The array will be saved to the highlighted file. The former contents of the file are lost. This command should be seldom used, and is included only for symmetry. Normally, the update command is used to save the array.

D - Delete a file. The highlighted file will be erased from the disk. This is equivalent to using the ERA command under CP/M.

<Esc> - Exit disk command mode. When <Esc> is typed, the user will be returned to the main system.

Each of the disk commands (except <Esc>) requires confirmation. When either L, S, or D is typed, one of the messages

```
Loading <file> - type Y to proceed -
Saving <file> - type Y to proceed -
Deleting <file> - type Y to proceed -
```

will be displayed. At this point, typing Y will cause the selected action to be carried out. Any other character, including <Return>, will cancel the action.

If when disk command mode is entered, there are no EPL files on the disk, the message

```
No files...
```

will be displayed. Obviously, there are no files to load or delete; therefore the only possible action is to save the array, which in this case MUST be done with the update command. Typing any character will return to the main system.

E - the ENTER command

The enter command is used to enter data or formulas into the array. It has four forms, for entering text, lines, values, or formulas. Since the four are drastically different, they will be explained separately.

ET - Enter Text

The enter text command will accept a single argument and write it into the current location. The general format is

```
ET text
```

The format for the ET command is somewhat precise. There must be one character after the ET, then whatever is after that is considered the text. For example, in the command

```
ET Hello There<                    (the < indicates the cursor)
```

will enter the text "Hello There". On the other hand, the command

```
ET  Income Type <
```

will enter the text " Income Type ". In other words, exactly what you type is what you get. Since whatever the user types is taken as the text, no multiple form of the ET command is possible (unlike the other enter commands).

EL - Enter Line

The enter line command is used to enter a line of data where the data is simply one character repeated. This is typically used for a dividing line of some sort, or perhaps the line after a column of numbers above the total, or something similar. There are two formats of the enter line command:

```
EL character
EL character ncolumns
```

The first type will create a line consisting of the (character) all the way across the array. The second will create a line only extending across a certain number of columns, specified by (ncolumns). In either case, the current location will be used as the first (leftmost) column; that is, the column in which the line begins.

The EL command operates much like the ET command. The data in the line is

simply treated by the system like any other text. The EL command automatically creates the sequence of characters to be the exact width of each column as it is entered. This can be used to create a "broken" line, also. For example, if the widths of all columns are set to 10, then the EL command is used, then the widths of all columns are set to 12, there will be a 2-character break in the line between each of the columns.

EV - Enter Value

The enter value command and its variations are used to enter numeric values into the current location and possibly adjacent locations in the array. The simplest form is

```
EV number
```

which will write the value (number) into the current location. Additionally, this form of the command can be "implied" by simply typing

```
number
```

when in command mode. That is, typing "123" and "EV 123" are the same.

The second form of the EV command is used to enter values into adjacent locations, either across a row or down a column. The forms are

```
EVR number number number number etc.
EVC number number number number etc.
```

The first will enter as many successive numbers as are typed into the current location and successive columns on the same row; the second will do the same, only the numbers will be entered into successive rows on the same column. For example, if the current location is [1,1], the command

```
EVC 1 2 3 4 5
```

will enter 1 into [1,1], 2 into [2,1], 3 into [3,1], etc.

The final form is the "repeat" form. Is is used to enter the SAME number into successive locations. The forms are

```
EVRR number count
EVCR number count
```

The first will enter the number (number) into successive columns on the same row for (count) columns; the second will do the same down a column. For example, if the current location is [1,1], the command

```
EVCR 100 5
```

will enter the value 100 into locations [1,1] through [5,1].

EF - Enter Formula

The enter formula command is used to enter formulas into the array. It has two forms, single and multiple. The single form is

```
EF formula
```

which will enter the (formula) into the formula table. It will set the value in the current location to 0, unless a "SC A" has been done (see the set command), in which case the formula will be evaluated and the result of the evaluation put into the current location.

The multiple form of the EF command is

```
EFM formula nrows ncolumns
```

where (nrows) and (ncolumns) are the number of successive rows and columns, respectively, to write the formula into. The array section which will get the formula may be thought of as a rectangle consisting of (nrows) rows and (ncolumns) columns, with the current location as the upper-left corner. For example, if the current location is [1,2], then the command

```
EFM 1.1*[.,.-1] 8 11
```

will write the formula "1.1[.,.-1]" to 8 rows and 11 columns, or specifically, into locations [1,2] through [1,12], [2,2] through [2,12], and so on through [8,2] through [8,12].*

F - the FORMAT command

The format command is used to choose the manner in which the data in the array will be displayed and printed. There are four forms of the command, but they differ only in the portion of the array that they affect. They are

F format
FR format
FC format
FA format

F means to format only the current location. FR means format the current row, FC means the current column. FA means format the entire array.

The (format) consists of zero or more individual format characters. These characters can be listed in any order, in a generally "free" fashion. Following are the various format characters.

$ - Dollar Sign. If $ is included, numbers will be printed with a leading dollar sign. For example, 123 will be displayed as $123.

, - Comma. If a comma is included, numbers will be printed with commas inserted every three digits to the left of the decimal point. That is, the value 123 would be unchanged, but the value 1234567 would be 1,234,567.

0-15 - Digits. Including a number in the range 0 through 15 will set the number of digits printed to the RIGHT of the decimal point. For example, the value 123 with a format of 4 would be printed as 123.0000.

% - Percent. The percent character indicates that the value is to be considered a percentage, and it will be printed with a percent sign following it. In addition, the number will be multiplied by 100 before being displayed. For example, the value .13 will be displayed as 13%.

R - Right Justify. For text, the R character causes right justification.

Each location in the array is either "formatted" or "unformatted". Typing a format command without any arguments (for example, "FC ") sets a location or locations to "unformatted". In this case, numbers and text will both be displayed left justified. Numbers will be displayed in a "general" format, meaning however necessary to express the value (For example, 100 will be 100, 3.14159 will be 3.14159).

The only format option for text is R, right justify. Its absence indicates left

justification.

For numbers, formatting is a little more complicated. ANY numeric format sets the location to "formatted". When a location is formatted, numbers are RIGHT justified. This right justification should not be confused with the R character for text.

Each time a F command is used, it overrides any previous F command. For example, if a "F 2" command is issued, the current location will be set to 2 places to the right of the decimal point, typically used for "dollars-and-cents" notation. If it is then desired to add the dollar sign, the command "F $" will NOT function as expected, since it cancels the effect of the "F 2" command. The proper command would be "F 2 $" since this combines the two commands. This brings up an important point: Leaving out the number-of-digits character in a format command is the same as using 0; that is, "F $" is identical to "F $ 0".

As was indicated above, the format characters are entered in a "free" fashion. Their order is unimportant. For example, the commands

```
F 2$,     F $,2     F ,$2     F $ , 2     and F 2 , $
```

are all identical. Format characters may be used in whatever combination desired, except that the combination "$%" will produce a meaningless figure; for example, 123.45 will be displayed as $12345%.

Since there is limited space available for format characters, the R and , characters are actually the same thing. That is, using a comma on text will right justify it, and using R on a number will insert commas. Since a location cannot contain both text and a number, this should not cause any problem.

H - the HELP command

The help command is used to access a screenful of assistance (commonly called a "help screen") for a particular command. There are actually three different forms of the help command.

```
H
H letter
?
```

Just typing H gives a help screen on typing commands, editing, and moving the cursor. Typing H followed by a letter gives a help screen on the command beginning with that letter. For example, HF gives help on the format command.

The help command reads the help screen from the file EPL.SYS on the currently-logged-in disk. That disk should not be confused with the currently-selected disk used for ExecuPlan! The logged-in disk is the disk that CP/M thinks is the current one. To be more specific, the disk that was in the prompt before ExecuPlan was executed. If it was A>, then the logged-in disk is drive A; if it was B>, then it was drive B, etc. If the file is not present on the disk, then the system will say "Help Unavailable".

The ? command is used to get a QUICK help screen. It tells how to get more help (via the H command) and gives a list of the command letters and their meanings. This screen is part of the program, not read in from the disk. Therefore, it is always available.

I - the INITIALIZE command

The initialize command is used to set everything back to the standard. Specifically, the command

Resets the array size to 20 x 20,
Resets the row and column titles back to 1 2 3 4 5 etc.,
Clears all main titles,
Clears the entire array,
Sets the current file to none,
and Resets the print blocks back to standard.

The format of the command is simply

```
I
```

It will display a warning message,

```
** Warning: Initialization erases ALL data - type Y to proceed -
```

at which point you may type Y to proceed with the initialization. Typing any other character (including <Return>) will cancel the command.

J - the JUMP command

The jump command is used as a quick way to move the screen around on the array, faster than using the arrow keys. There are three jump commands:

```
JB
JR row
JC column
```

The first, JB, simply jumps to the top-left corner of the array, which would be, [1,1] on an initialized array.

JC jumps to the specified column. That column will be the leftmost on the screen after execution.

JR jumps to the specified row. The row will be the top on the screen after the jump, unless the size of the array makes this impossible. If the jumped-to row is within 18 rows of the end, it will be somewhere in the middle of the screen.

K - the KILL command

The kill command is used to "kill", or remove, a row or column from the array. It does not, however, change the size of the array. Therefore, when it kills a specified row or column, it creates a new one at the end in order to keep the array the same size.

The format of the kill command is

```
KR row new-row
KC column new-column
```

where (row) or (column) is the row or column to kill, and (new-row) or (new-column) is the title to be assigned to the row or column created at the end of the array.

Example: If your array currently has 12 rows, numbered 1 through 12, and you execute the command "KR 6 13" then your resulting rows will be 1,2,3,4,5,7,8,9,10,11,12,13.

There is one VERY IMPORTANT thing to note about the kill command! Relative references in formulas which refer to or over the killed row or column will NOT be changed. In other words, they will be INCORRECT after the command is executed. In the example above, if you had a reference in row 7 which contained something like .-2, before you executed the kill command, that would have pointed to row 5; after the command, it will point to row 4.

L - the LIST command

The list command is used to produce a list on the printer of all of the formulas associated with the array. The list is printed in the order in which the formulas will be evaluated; that is, either in row-major or column-major order, depending on the current "SC R/C" setting. The format of the command is simply

L

The listing will have the heading at the top of each page

```
FILENAME      Formula list by row
                            (or column, if appropriate)
```

where FILENAME is the name of the file if one is assigned. The format of the listing is

```
[destination]      = formula
```

where [destination] is the destination of the formula, and (formula) is the text of the formula.

If during the listing it is desired to stop, typing <Esc> will cancel the command.

M - the MOVE command

The move command is used to move a row or column from one place in the array to another. The command format is

```
MR row dest-row
MC column dest-column
```

where (row) or (column) is the row or column to move, and (dest-row) or (dest-column) is the row or column to move it adjacent to.

Depending on which direction the row or column is moved, it will either be placed above/to the left of the destination, or below/to the right. Specifically: for a COLUMN, if it is being moved to the left, is will be placed to the left of the destination; if it is being moved to the right, it will be placed to the right of the destination. For a ROW, if it is being moved up, it will be placed above the destination; if it is being moved down, it will be placed below the destination.

For example, say you have the rows 1,2,3,4,5,6. If you execute the command "MR 5 3", the resulting sequence will be 1,2,5,3,4,6. If you had executed the command "MR 2 6", the resulting sequence would have been 1,3,4,5,6,2. This example is equally applicable to columns.

NOTE: Like the K command, the M command does not change relative references in formulas.

O - the OPEN command

The open command is used to open up a new row or column in the array. The command does NOT change the size of the array, therefore when a new row or column is created, the last row or column of the array is removed, and its contents lost.

The format of the open command is

```
OR ref-row row
OC ref-column column
```

where (row) or (column) is the title of the new row or column to create, and (ref-row) or (ref-column) is where to put it. If COLUMN, the new column will be to the LEFT of the ref-column; if ROW, the new row will be ABOVE the ref-row.

Example: If there are currently 10 rows, numbered 1 through 10, then executing the command "OR 7 NEW" will result in rows 1,2,3,4,5,6,NEW,7,8,9 with row 10 being lost.

NOTE: Like the K and M commands, the O command does not change relative references within formulas.

P - the PRINT command

The print command is used to print the array, or to cause what would be printed to be written into a disk file for editing with Scope or Memorite. Like the disk command, the print command is actually an entire command mode. It is invoked simply with

```
P
```

Print Command Mode

When print mode is entered, the screen will be erased and replaced with what is called the "print screen". The screen is divided into 5 "blocks", each one controlling certain aspects of what will be printed.

Block 0 - Main titles. This block is used to select how the main titles will be printed. For each of the four main titles, which will be displayed, the choice may be made whether to R - right justify, L - left justify, C - center, or X - not to print at all.

Block 1 - Print bounds. This is used to select the portion of the array to be printed. Specifically, the starting row, starting column, ending row, and ending column are specified. By proper manipulating of these bounds, an array much larger than a piece of paper can be printed on several sheets, then the sheets rearranged to form a large sheet.

Block 2 - Paper size. This informs the system of the size of paper being used, in terms of number of characters per line and number of lines per page. The width is used only for centering the titles, but the length tells the system the maximum number of lines to print on one page before skipping to the next page.

Block 3 - Row/column titles. This block allows the user to choose whether or not to have the system print the row and column titles on the report.

Block 4 - Invisible. This allows the user to set rows and columns to "invisible", meaning that they will NOT be printed, even if they are within the print bounds selected by block 1. This is most often used to prevent the printing of some type of intermediate result column. There is also a provision for overriding the invisible function, that is, to go ahead and print the invisible rows and columns.

Print Commands

In addition to the blocks, the bottom of the screen will list the available commands. Following are the commands, and how to use the blocks.

P - Print the array. Typing P will cause the array to be printed, using the settings of the blocks to define the bounds, titles, etc. If during printing you wish to stop, type <Esc>.

D - Disk. Typing D is just like P, only instead of printing the array, the data will be written to a disk file. The format of the data will, however, be identical to when it is printed.

When D is typed, the message

Please enter file name:

will be displayed. Type the name of the file that you wish to write the data to, followed by <Return>. The file will be assumed to have .MEM as the extension, and must not already exist. If it does, an error will be displayed and the command cancelled. The current disk will be used for the file; to use a different disk, first select it with the disk command from the main system.

If when you are prompted for the file, you decide not to execute the command, simply type <Return> without typing the file name, and the command will be cancelled.

F - Formfeed printer. Typing F will simply cause a formfeed character to be sent to the printer. This normally has the effect of rolling the paper up to the top of the next page.

0 - Edit block 0. Typing 0 will allow you to change the information in block 0. Note that only the justification character can be changed at this time. To change the text of the title, the TMx command must be used when under the main system.

When 0 is typed, the justification character of the first main title will be highlighted. At this point, you have several options:

Type <down arrow> - this will move the highlight down to the next title. If you are already at the bottom (fourth) title, the highlight will be moved back to the first one. The justification character for the current title will not be changed.

Type <up arrow> - this is the opposite of <down arrow>. The highlight will be moved up to the previous title. If you are already at the top, it will be moved down the the bottom one. The justification

character will not be changed.

Type <Esc> - this will leave the current character unchanged, and return to print command mode.

Type <Return> - this is identical to <down arrow>, except that if you are at the bottom title, it will stop editing block 0 and return to print command mode, similarly to <Esc> above.

Type a justification character - typing either L, R, C, or X will set the justification character for the current title to whatever is typed. The highlight will not be moved, so if the wrong thing is typed, you may simply retype the correct character.

Anything other than the above characters will simply be ignored.

1 - Edit block 1. *Typing 1 will allow you to edit the information contained in block 1. When 1 is typed, a cursor will appear a little to the right of the first line in block 1. At this point, you are in a mode similar to block 0, but a little different. Essentially, while block 0 is an "instant" block, meaning that when you type a character, it immediately replaces the previous character, block 1 is an "updated" block, meaning that the new information appears to the right of the old information and is edited by itself, and only replaces the old information when you leave block 1 and return to the print command mode.*

Of course, the data you type is not a justification character. Instead, the appropriate title is typed. The first time print command mode is entered, the bounds are set to the size of the entire array.

When typing the title, up to eight characters may be typed, terminated by either <Return> or <Esc> (the difference is explained below). In the process of typing the title, <Bs> may be typed to back up one character.

Additionally, there are two special characters allowed, if typed as the first character. Typing <Control-F> will display the "first" row or column of the array; typing <Control-L> will display the "last" row or column. This could be useful if you wish to, say, start the printing at the first row, but you're not sure what the title of the first row is.

The same editing characters are available for block 1 as for block 0 (<down arrow>, <up arrow>, <Esc>, <Return>), but the way they work is slightly different. When by some method the cursor comes to be on a line, any previously-typed data on that line is erased. To change a single item without having the cursor move down the next line (and consequently erase something that you might have typed there), <Esc> may be typed instead of <Return>, which will immediately return to print command mode.

When print command mode is reentered, the information in block 1 will be updated based on the new information typed. If a title typed does not exist, it will be highlighted and an error displayed. Any line which contains an error will not be updated.

Although as explained this probably seems very complicated, it is virtually self-explanatory when actually done.

2 - Edit block 2. Typing 2 allows the user to modify the information under block 2, the paper size block. All aspects of block 2 are identical to block 1, except that instead of typing row or column titles, you type decimal numbers. For page width, you should type the number of characters per line. This is only used for centering the main titles. For page length, you should type the maximum number of lines you wish printed on a page. On a standard 11" page, a length of 56 lines allows reasonable margins on the top and bottom. For both length and width, the system will accept values in the range of 40 through 255.

3 - Edit block 3. Typing 3 allows the user to select whether row or column titles are to be printed as part of the report. If the appropriate line is Y, the titles will be printed; if N, they won't. Block 3 is similar to block 0, except that the items are only updated when print command mode is returned to.

4 - Edit block 4. Block 4 is used to determine which rows or columns, if any, are to be considered "invisible", meaning that they will not be printed.

Block 4 is perhaps the most confusing block, because it includes two individual fields for each line. The first is the row/column indicator, one of the characters R or C. The second is the title of the row or column.

When block 4 in entered, a cursor will appear on the top line of block 4. At this point you have several options.

> The <up arrow>, <down arrow>, and <Esc> keys function the same as block 0. The exception is that when you are on the bottom (tenth) line, the down arrow moves to the "Print Inv? " field. From there the <down arrow> will move back to the first line. The <up arrow> moves in the same manner, only up instead of down.
>
> Typing R or C will set the row/column indicator for the current line to whatever is typed.
>
> Typing <Space> will turn OFF the line; that is, when <Return> is typed after the <Space>, it will also remove the title on the current line.
>
> <Return> is typed to allow movement to the title field. For example, to

set a line to Row 1, you would type "R" <Return> followed by "1" as explained below. If the row/column indicator already contained an R, you could simply type the <Return> and then proceed with the "1".

If <Return> was typed, the cursor will jump three spaces to the right. At this point the system is awaiting a title. Type the row or column title that you wish to set to invisible. If the title is terminated with <Return>, the cursor will proceed down to the next line. If it is terminated with <Esc>, block 4 will be updated and the print command mode will be returned to.

When the cursor is moved to the "Print Inv? " field, you may type Y or N, in the same manner as block 3. If N is typed, the invisible function will work, that is, the rows and columns indicated will not be printed. If Y is typed, the function will be effectively overridden, that is, the rows and columns listed will be printed anyway.

When print command mode is returned to, the rows and columns in block 4 are looked up. If any of them are not found, they will be highlighted and an error displayed. However, the information will be left in the block, and when the array is printed, the invalid entries will be ignored.

Characteristics of print command mode.

The information in the print blocks remains as set until changed. In addition, all of the information is saved with the file, so when a file is loaded, all of the information in the print blocks will be the same as when the file was saved.

Q - the QUIT command

The quit command is used to return to CP/M. There are three forms of the quit command:

```
Q
QY
QN
```

If QY is typed, the array will automatically be updated; if no current file exists, you will be asked for one. If QN is typed, the array will not be updated.

If just Q is typed, you will be prompted with

```
Exiting - Type Y to update -
```

If at this point you type Y, the file will be updated and the program exited. If you type N, the file will not be updated, and the program exited. If you type <Esc>, the system will cancel the command altogether. Anything else will be ignored.

R - the ROUND command

The round command is used to change the precision of numbers in the array. Specifically, all numbers in the array that are NOT calculated as the result of a formula will be changed to match their representation on the screen. For example, if the value 1.469 is in a certain location, but the format is 2, then the number is being displayed as 1.47. Therefore, when the round command is executed, the number will actually be changed to 1.47.

The round command is executed simply by typing

```
R
```

at which time the system will warn

```
** Caution: Data will be changed - type Y to proceed -
```

If you type Y, the action will be carried out. Typing anything else will cancel the command without having any data changed.

S - the SET command

The set command is used to set or change certain aspects of the system. There are four set commands; set size, calculation, printer, and disk. These are explained separately.

The Set Size Command

The set size command is used to change the size of the array. The form of the command is

```
SS nrows ncolumns
```

where (nrows) is the number of rows to make the array, and (ncolumns) is the number of columns to make the array. When the system is initialized, the size is 20 x 20. To change to a 9 x 13 array, for example, the command would be

```
SS 9 13
```

If the number of rows or columns is being decreased, the message

```
** WARNING - Some rows containing data may be lost - type Y to
proceed -
```

will appear. If the number of columns is being decreased, the message will indicate columns instead of rows, of course. Typing Y will cause the action to be carried out; anything else will cancel the action. If you are decreasing both rows and columns, you will get both messages, one at a time. Responding Y to rows but something else to columns will result is the number of rows decreasing, but the number of columns remaining the same.

If is is desired to only change one dimension, the value 0 may be put in the other. For example, to change the array from 20 rows to 30, but not affect the number of columns, the command would be

```
SS 30 0
```

NOTE: There is a substantial amount of logic involved when shrinking the array, particularly when columns are involved. If an array with many rows has the number of columns lessened, several moments (read a minute or so) could elapse before the system completes the command. Therefore, do not fear system failure. SUGGESTION: When changing the size of the array, the rows are changed first. Therefore, say you're changing the 20 x 20 array to 10 x 100. The system will first create 100 rows, then shrink to 10 columns.

This will take quite a while, since the system is removing 10 columns of 100 rows each! To make the operation faster, FIRST change the number of columns (SS 0 10) THEN change the number of rows (SS 100 0). Using this method, only about 3 seconds will be used, a substantial increase in speed. If you are decreasing the number of rows and increasing the number of columns, FIRST decrease the rows, THEN increase the columns. The object is to make the number of rows as small as possible when creating or removing columns.

The Set Calculation Command

The set calc command is used to change the order and frequency of when the formulas are evaluated.

Formulas may be evaluated in either "row-major" or "column-major" order. If they are evaluated in row-major order, that means that they will be evaluated across each row, then down to the next row and across it, etc. In other words, the same way you read. In column-major order, they are evaluated down each column, then moved over to the next column and down it, etc.

The frequency may be set to either "manual" or "automatic". In manual mode, the formulas are only evaluated when the <Tab> key is pressed. In automatic mode, they are evaluated whenever a value or formula is entered.

The commands accepts the letters R, C, A, and M to represent each of the above options. Obviously, only two may be entered at once, but they can be entered in any order. For example, to set the calc order to column-major, and the frequency to automatic, the command would be

```
SC C A          or          SC A C
```

To change only, say, the calculation order, this time to manual, the command would be

```
SC M
```

While it will not produce an error to enter both R and C or both A and M at the same time, only the last one on the line will be used.

The Set Printer Command

The set printer command is used to send special characters to the printer. This should seldom be used, but is provided for special cases. The most common use would be so set a printer for some special mode.

The format of the command is

```
        SP byte byte byte etc.
```

where (byte) is a decimal number. As many or few bytes may be sent as needed. For the exact bytes required for particular printers, you will have to refer to the appropriate printer's manual.

For an example on what this might be used for, consider the Vector Matrix printer. If you typed the command

```
        SP 27 80
```

which corresponds to "ESC P" (the printer manual explains this), the printer would switch between 132 and 80 characters per line.

The Set Disk Command

The set disk command is used to reset the disk system in case you wish to change floppies. CP/M will normally not allow you to do this; it you do, you will get a BDOS read-only error, which can't be recovered from. Therefore, if you wish to change a floppy disk, put the new diskette into the drive and type

```
        SD
```

which will tell CP/M about the new disk and allow you to access it.

T - the TITLE command

The title command is used to set and delete the main titles for the array, and to change the row and column titles. The commands for the row and column titles are substantially different from the commands for the main titles, therefore they will be explained separately.

Main Titles

The first two forms of the title command are for handling the main titles. They are as follows.

```
TMx text
TDx
```

where (x) denotes the number of the referenced main title. Since there are up to four main titles, (x) must be in the range of 1 to 4. If you are referencing the first main title, the 1 may be skipped. That is,

```
TM text     is the same as TM1, and

TD          is the same as TD1.
```

The (text) specified in the TM command refers to the text of the title, that is, what you wish the title to be. For example, the command

```
TM Budget Forecast
```

will set the first main title to "Budget Forecast".

The TD command is used to delete a main title, which is to say, make it blank.

The first main title is displayed on the screen, centered in a field of dashes. The remainder of the main titles can be viewed by using the P command, where they will be displayed in block 0 (see the P command). Note that when a main title is entered (or changed), the justification character is set to L. When a title is deleted, the justification character will be changed to X. It is possible to make a "comment" by typing a main title, then changing its justification character to X, thus keeping it from being printed.

Row and Column Titles

The second pair of title commands are used to change the row and column titles. The commands are

```
TR old-title new-title
TC old-title new-title
```

The TR is used to title a row, and TC is used to title a column. The (old-title) denotes the current title of the row or column you wish to change, and (new-title) denotes the new title you wish to assign to the row or column. For example, to change the title of row 1 to INCOME, the command would be

```
TR 1 INCOME
```

If you then changed you mind, and wanted to make the title SALES, the command would be

```
TR INCOME SALES
```

Note that the second time, the previously-assigned new title, INCOME, was used to refer to the row.

It is acceptable to use the relative method of referencing when specifying the old title. For example, to change the title of the current column to "JAN", you could use the command

```
TC . JAN
```

Titles for rows and columns may contain any characters, but must be only one word with a maximum of 8 characters, and must not start with a period (.). This is to prevent confusion with relative references.

U - the UPDATE Command

The update command is used to update the disk with the current contents of the array. Assuming there is a "current" file, all that need be typed to update the disk is

```
U
```

and the update will occur. If there is no current file, the system will say

```
No "current" file to update; Please type NEW filename:
```

and await your response. Type the name of a new file and <Return>. The system will create the file and save the array into it. In addition, it will be made the current file. If the file already exists, an error will occur. When asked for the file name, just typing <Return> will cancel the command.

Under certain circumstances, it may be desirable to save the array to disk, but NOT under the current file. This can be done by typing

```
UN
```

which stands for "Update New". The system will prompt with

```
Please type NEW filename:
```

and the proceed as above for a new file.

V - the VERIFY command

The verify command is used to verify the size of the array, or determine what the current actual cursor position is.

TAKE NOTE: Say for example that you have rows titled

HDGS, MONTHS, -, 1, 2, 3, 4, and 5.

Now you wish to remove ONE row. Your inclination will be to type the command "SS 4 0". This is not correct, however! Although the last row's title is 5, it is not the fifth row! If you typed that command, you would lose rows 2, 3, 4, and 5, which was certainly not your intention!

To eliminate the problem, two commands are provided. They are

```
VS
VC
```

The first, VS, tells the size of the array. The second, VC, tells the current cursor position on the array, in terms of absolute position. The VS command causes the message

```
Current size is x rows by y columns.
```

to be printed, where (x) is the number of rows and (y) is the number of columns. The VC command causes the message

```
Cursor is at row x and column y.
```

to be printed, where (x) and (y) are the row and column, respectively, where the cursor is at. In the example above, if the cursor was in the first column and on the row titled "5", then the VC command would result in the message

```
Cursor is at row 8 and column 1.
```

Typically, you might move the cursor to the first row (or column) that you wish to remove, then use the VC command, which will tell you the actual number of that row or column. If, on the other hand, you know that you wish to decrease the size of the array by a certain number of rows or columns, the VS command would be most useful.

By utilization of these commands, the possibility of accidentally destroying data with the SS command should be reduced.

W - the WIDTH command

The width command is used to set the widths of columns in the array. There are two forms provided.

```
W column width width width etc.
WA width
```

The first form allows you to specify (column), which is the column to start with, and a many (width)s as desired. Each (width) typed will be assigned to the following column. For example, assume that columns are titled JAN, FEB, MAR, and so forth. The command

```
W APR 20
```

will set the width of column APR to 20 characters. The command

```
W JAN 15 15 25 25 7
```

will set the width of columns JAN and FEB to 15, columns MAR and APR to 25, and column MAY to 7.

The second form will assign all columns the (width) specified.

Columns widths may be in the range of 2 to 64 characters, although columns narrower than about 6 characters begin to get useless.

X - the EXCHANGE command

The exchange command is used to exchange, or swap, two rows or columns. The commands have the form

```
XR row-1 row-2
XC column-1 column-2
```

where (row-1) and (row-2) or (column-1) and (column-2) are the rows or columns to be exchanged. All aspects of the rows or columns are exchanged - the data, the formulas, the titles, and if columns, the widths.

NOTE: As with the K, M, and O commands, relative references in formulas affected by the execution of the command will not be changed. References over an exchanged row or column will not be bothered. References TO the row or column will simply get the new data instead of the old. However, references CONTAINED in the row or column will now be evaluated relative to the new position, whereas they were entered relative to the old position. This could potentially result in incorrect calcalations, so beware.

Section 3

Errors

- *All errors, what they mean, what causes them, and how they can be avoided.*

ERRORS

From time to time, something will be done wrong. ExecuPlan has a vast number of error messages to help you figure out what was haywire.

When an error occurs, the error message is displayed in the far bottom-right corner of the screen. Normally, a character or word somewhere on the screen is also highlighted. That character or word is the source of the error. Not all errors, however, have this feature.

After the error is displayed, the system just stops and waits for the user to type something. As soon as a character is typed, the system proceeds. Most errors result in the system returning to command mode and awaiting another command. Some, however, have other results. Certain math errors, for example, simply warn you; when you type a character, processing continues.

If an error occurs while formulas are being evaluated, then an additional message is displayed on the command line telling you where the formula is that caused the problem. Also, the formula itself will be displayed on the formula line of the display.

Following are explanations of all of the errors and their causes.

- COMMAND ERROR -

The command error indicates that the command typed is invalid. If the first character is highlighted, then that is the invalid command. If the second character is highlighted, then the first character is valid, but the second one is not.

- SYNTAX ERROR -

Syntax error indicates one of several things. If the first character following the command is not a space, then a syntax error will result. If an invalid character is encountered while a decimal number is being read, that will also cause a syntax error. There are also a couple of other obscure conditions that will cause a syntax error. The character highlighted will normally be the character that was undigestable.

- MISSING ARGUMENT -

This indicates that something was expected, but nothing was found. For example, typing a title command but leaving out one of the titles will cause a missing argument error. A character is not always highlighted, but if one is, it is at that point that another argument was expected.

- TITLE TOO LONG -

This indicates that a title was being read, but more than eight characters were found in the title. The ninth character will be highlighted.

- TITLE NOT FOUND -

This should be pretty obvious. A title was read, but there is no row or column with that title. The entire title that was not found will be highlighted. Also, typing a relative reference that refers to someplace off the array will cause this error.

- INVALID RELATIVE -

When a title is being read, if the first character is a period (.), then it assumes that a relative reference is in the works. If the character(s) following the period do not make sense, then this error will result.

- DUPLICATE TITLE -

Duplicate title indicates that a title was entered that should not already exist, but it does. An example would be the second argument in a TR command. This error is also used when a disk file name is typed for a new file, and the file already exists. The entire title (or file name) will be highlighted.

- OUT OF RANGE -

Certain commands expect a number within a certain acceptable range. If the number typed in not within that range, the out of range error will be the result. For example, typing a column width less than 2 or greater than 64 would cause this error. The number will be highlighted.

- BAD FORMAT CHAR -

A character is encountered in a F command that is not acceptable. The character that was unacceptable will be highlighted.

- DECIMALS > 15 -

This is similar to the above in that it means that something is wrong in a F command. This error, however, indicates that a number for the decimal count is too large. Fifteen is the maximum number of decimal places that may be specified. The offending number will be highlighted.

- BAD CALC ORDER -

During the reading of a SC command, a character other than R, C, M, or A was reached. The character is highlighted.

- INPUT ERROR -

An input error occurs when an EV command is being executed and something wrong is reached. Normally, this is a decimal number containing some garbage characters. The number being read is highlighted.

- EXCESS INPUT -

This is reached during one of the multiple forms of the EV or EF commands. For example, if the command "EVCR 45 100" is given, but there are only 50 rows, an excess input error will be generated. Whatever portion of the command that caused too much input to occur will be highlighted. Under the EV command, the extra will be ignored. If the command is EFM, the whole command will be cancelled.

- INVALID DRIVE -

When a D command is executed and a drive is specified, that drive must be in the range A to P. Anything outside of that range will cause an invalid drive error. The offending character will be highlighted. Note that even something within the range A to P may be invalid, since few systems have 16 disks! However, the system really doesn't know that, hence the extended range. If you type a drive within the range that doesn't exist, then CP/M will get into the act and give a BDOS error. These are unrecoverable!

- WRITE PROTECTED -

Under CP/M version 2, files may be set to "read-only" status. Trying to write to or erase such a file will result in this error. Nothing is highlighted.

- DISK I/O ERROR -

This error means one of several things. One possibility is that there is a physical error on the disk. Another is that an attempt is being made to write to the disk, but there is no more room. Finally, an attempt may be being made to read a file which is goofed up somehow. Nothing is highlighted by this error.

- FORMULA ERROR -

This is actually a rather general error. What it means is that there is something wrong in a formula, such as a non-existant function, an improperly-typed number, an invalid operator, or something else along that line. The system will try to highlight the character that caused the problem, but depending on the cause, that character might not actually be the source of the error.

- BAD RANGE BOUNDS -

When evaluating a multi-argument function, the arguments were invalid. There are actually two separate things that could be wrong. First, the arguments aren't references at all; second, they could be references, but define an invalid range for the function. The arguments for such a function must be, respectively, the top-left and bottom-right corners of a rectangle. The rectangle may in fact be a line, or even a point. However, the second reference can't have a row or column that is less than the row or column in the first reference. Nothing will be highlighted.

- MATH IMPOSSIBLE -

Certain things just can't be done with real numbers, and things like logarithms or square roots of negative numbers are such things. Nothing will be highlighted.

- DIVISION BY ZERO -

The cause of this is quite apparent. The particular thing about this error is that the character typed to recover from the error condition determines what will be used as the result of the operation (that caused the error). If the character "0" is typed, then zero will be used as the result. If the character <Esc> is pressed, then the system will abort the operation and return to command mode. Any other character will cause the value 9.99999999999999 times 10 to the 35th power to be used as the result of the division.

- OVERFLOW -

Some math operation resulted in a number that is just too big. The largest possible number or something near it will be used instead. Nothing will be highlighted.

- OUT OF MEMORY -

This indicates that there is too little memory available to carry out the operation requested. Normally, there is around 30K of free space to start with. With gobs of text and formulas in memory, this can disappear quickly. When an operation would result with less than about 100 bytes (characters) of free space, this error is caused. The padding is allowed because certain operations use some memory during their execution. Note that if this error is given during an EFM command with large arguments, there might really be enough room. The system allows for maximum tolerences when calculating the space available. If you think there is enough space, try reentering the formula, but in smaller multiples. Nothing is highlighted by this error.

- HELP UNAVAILABLE -

This indicates that the help command was used, but the help file was not found. The help file is called EPL.SYS, and must reside on the logged-in disk under CP/M. Either the disk containing EPL.SYS was not in the drive, or the user is assuming the wrong drive is the logged-in one. Nothing is highlighted.

Section 4

Math Capabilities

How the math package is used, how formulas are formed, and explanations of all operators, functions, and special capabilities.

MATH CAPABILITIES

ExecuPlan has a very powerful math package incorporated into it. Virtually all operators and functions necessary for any type of calculations are provided. Furthermore, their usage is in a simple, algebraic format.

The EF command is used to enter formulas into the system. For example, the command

```
EF AVG([1,1],[5,1])
```

would enter a formula which would compute the average of locations [1,1] through [5,1].

Characteristics of Formulas

This section will be devoted to explaining exactly how formulas are formed. If you, the reader, are familiar with the programming language BASIC, then suffice it to say that ExecuPlan handles formulas the same way. Assuming that you're not, then read on.

Formulas are essentially a list of items, where each item is either data of some type, or an operator. Things like numbers or functions are data; plus, minus, and so forth are operators.

Data and operators are simply strung together to form an algebraic expression. For example,

```
3+4+9
```

is a valid expression, containing numeric data and the operator "+". Under ExecuPlan, most formulas will "reference" locations in the array. A sample formula with a reference would be

```
5*[1,1]
```

which means "take the number 5 and multiply it by the contents of location [1,1]".

One of the more powerful features is that of functions. These can be confusing, because while they perform an operation like an operator, they are treated as data, because when evaluated, a function is a value. A simple function might be

```
SQR(15)
```

which means "take the square root of 15". However, when contained in an expression, such as

```
[3,5]-SQR(15)
```

note that it is treated like data. Note also the syntax of a function: the function itself, followed by a left parenthesis, then the data the function is to be performed upon (the argument), then a right parenthesis. Within the parentheses can be another expression, such as

```
SQR(45-[3,5]*12)
```

which will take the square root of the result of the expression which is its argument.

Parentheses may also be used as part of a formula, besides being used to enclose the argument for a function. They are used just as in algebra, to represent a partial result. For example

```
[1,5]/(3-4*[2,5])
```

which means to take the contents of [1,5] and divide it by the result of the parenthesized expression. Unlike most systems, ExecuPlan will not complain if there are not a matching number of left and right parentheses. Instead, it will just ignore the extras. Also, parentheses may be nested to any level, that is, you may have as many as you need to properly represent your expression.

There are really only two rules with regard to formulas. First, they CANNOT contain ANY SPACES (blank characters). The first blank encountered is considered the end of the formula, and the extra past it will either be ignored or cause an error. Second, a formula can only be as long as you can type, which limits it to about 74 characters.

Precedence of Operators

Given an expression, the question arises as to in what order to evaluate the operators and functions. There are two normal ways to do this. One is called "left-to-right", and means that the operators are evaluated in the order they are encountered. The order is called "precedence", which means that they are evaluated in a specific order with certain operators first, regardless of the order they're in.

It has been said that business people use the l-to-r method, and scientific people use the precedence method, and that since computer programmers are scientific types, that's why computers always use the precedence method. Well, that may be true, but the programmer of this system is just as much a business type as scientific type. The reason that precedence was used is *simply that it is more powerful; that is, certain operations cannot be done as easily with the l-to-r method. Besides, by now most business types are so used to precedence that it would cause even more confusion to have a program use l-to-r!*

Before getting to precedence, though, it might be a good idea to mention what the operators are! There are five of them:

+	*addition*
-	*subtraction*
*	*multiplication*
/	*division*
^	*exponentiation (that is, raise to power. 2^3 is 8.)*

So much for the operators. Now, basically their precedence is as follows:

1. *parenthesized expressions*
2. *functions (remember that functions are evaluated, then treated as a value from that point on)*
3. *exponentiation*
4. *multiplication and division*
5. *addition and subtraction*

When operators of equal precedence are met, then those operators are evaluated left-to-right.

Examples:

*3+4*5*	*4 * 5 first, then add 3*
3(3+5)*	*3 + 5 first, because it's parenthesized*
*12/int(3*11)*	*3 * 11 first, then function "int", then division.*
*4+4-2*7*	*take 4, add 4, then subtract the product 2*7 (hence the result is -6)*

NOTE: Often it will be desired to use a negative number, for example -3, in an expression. Therefore, it should be explained how it will be handled by the formula evaluator.

Essentially, whenever two consecutive operators are encountered, the program inserts a 0 between them. Thus, the sequence 2++4 would result in 2+0+4, which would give the presumed correct answer. In some cases, however, an incorrect answer might be arrived at. For example, the sequence 4*-3, which should evaluate to -12, will evaluate to 4*0-3, which is -3.

Normally, two consecutive operators should never be used. The example above, however, is a valid possibility. It is an example of the unary "negative" operation, which is the only usual possibility.

To eliminate the problem, use parentheses around such an operation when it in used in an expression. For example, -SQR(2) would not need parentheses, but 4*-3 would, so you'd enter 4*(-3).

Defined Constants

There is one other nice little feature of the program, defined constants. These are simply a couple of numbers that may or may not be used very much, but will save some typing when they are. The defined constants are

#PI	The value PI, 3.1415926 etc.
#E	The value e, 2.718281 etc.
#RND	A random number in the range 1 to 2.

Defined constants can be used wherever a number would normally be used.

Functions

At this point, we'll take a look at the functions provided in ExecuPlan. There are basically two types of functions, single-argument and multi-argument. A single-arg function is something like square root; a multi-arg function would be something like standard deviation.

These two types of functions are slightly different, beyond the obvious fact that they take a different number of arguments. A single-arg function can take anything as an argument - a number, defined constant, another function, a reference, even a whole expression.

A multi-arg function requires that the arguments be references. Specifically, these references represent the top-left and bottom-right corners of a

rectangular portion of the array. The rectangle may actually be a line, or even a single location, but in a manner of speaking, these are still rectangles.

For example, the arguments ([1,1],[5,5]) define a 5 x 5 rectangle. The arguments ([1,1],[5,1]) define a vertical line; the arguments ([1,1],[1,5]) define a horizontal line. The arguments ([1,1],[1,1]) define a point. Nevertheless, they would all be acceptable. The arguments ([3,3],[4,2]) would, however, not be allowed, since the second reference is to the LEFT of the first. It would also not be allowed if it were ABOVE the first.

The reason for this restriction should be apparent. The functions which take multiple arguments operate on a range, that is, a group of values. Only by specifying the bounds of the range, as references, can the function possibly know what numbers to use.

Naturally, there is always an odd case. Here, it is the net present value function, which requires both a range and numeric arguments. The exact format of this function will be explained when the function is explained.

Following are explanations of all of the functions, how they're used, what they do, and which type they are.

ABS **Absolute Value**

Type: single-arg

The absolute value function returns the absolute value if its argument. In other words, if the argument is positive, it is returned unchanged. If it is negative, it will be made positive. Example:

ABS(3) = 3 ABS(-4) = 4.

INT **Integer**

Type: single-arg

The integer function returns the greatest integer less than or equal to the argument. Example:

INT(3) = 3 INT(#PI) = 3 INT(5.9) = 5 INT(-1.1) = -2.

SIN	**Sine**
COS	**Cosine**
TAN	**Tangent**
ATN	**Arctangent**

Type: single-arg

These functions return the result of the appropriate trigonometric function. The argument is expected to be in radians (with the exception of ATN, which returns its result in radians).

LN	**Natural Logarithm**
LOG	**Decimal Logarithm**

Type: single-arg

These functions return the appropriate log of the argument. LN is the natural, or naperian, log (base e), while LOG is the decimal (base 10) log. Example:

LN(#E) = 1   LOG(100) = 2.

EXP **Exponent**

Type: single-arg

This function also commonly called antilog. It returns the natural (base e) antilogarithm of the argument. Example:

EXP(1) = e

SQR **Square Root**

Type: single-arg

The square root function returns.... the square root of its argument. Bet you would have never guessed.

SUM	**Summation**

Type: multi-arg

The sum function returns the total of all of the numbers in the range specified. For example, if [1,1] through [5,1] contain the values 1, 2, 3, 4, and 5, then

SUM([1,1],[5,1]) *returns the value 15.*

MIN	**Minimum**
MAX	**Maximum**

Type: multi-arg

The MIN and MAX functions return the smallest or largest, respectively, number is the range. Assuming the conditions above (in the SUM explanation), MIN would return 1 and MAX would return 5.

AVG	**Average**
MEAN	**Mean**

Type: multi-arg

The AVG and MEAN functions are the same thing - the average of the numbers in the range specified. Both are provided so that whichever term is preferred by the user may be used.

VAR	**Variance**
SD	**Standard Deviation**

Type: multi-arg

The VAR and SD functions compute the variance and standard deviation, respectively, of the argument range.

COUNT *Counter*

Type: multi-arg

The COUNT function will simply return the number of items in the argument range. This function is used by the average, variance, standard deviation, and net present value functions. IMPORTANT NOTE: This function, and therefore all of the functions that use it, react in a certain way to invalid contents of locations in the array. That is, when a certain location within the argument range does not contain a number (instead, it contains nothing or text), the value zero will be used instead. The location will still be counted! Therefore, any of the above-mentioned functions could return an invalid result if any locations in the argument range are invalid.

NPV *Net Present Value*

Type: special multi-arg

This function returns the computed NPV of the argument range, using additional numbers specified in the arguments. The standard formula for net present value is

$$\sum_{t=1}^{N} \frac{F_t}{(1+k)^t} - I$$

where F(1), F(2), through F(n) are cash returns for years 1 through n, k is the interest rate, and I is the initial cost. The format for the NPV function is

NPV([bnd-1],[bnd-2],k,I)

where k and I correspond to the same variables in the formula. The value for n is computed as the COUNT of the locations in the range bounded by [bnd-1] and [bnd-2].

Section 5

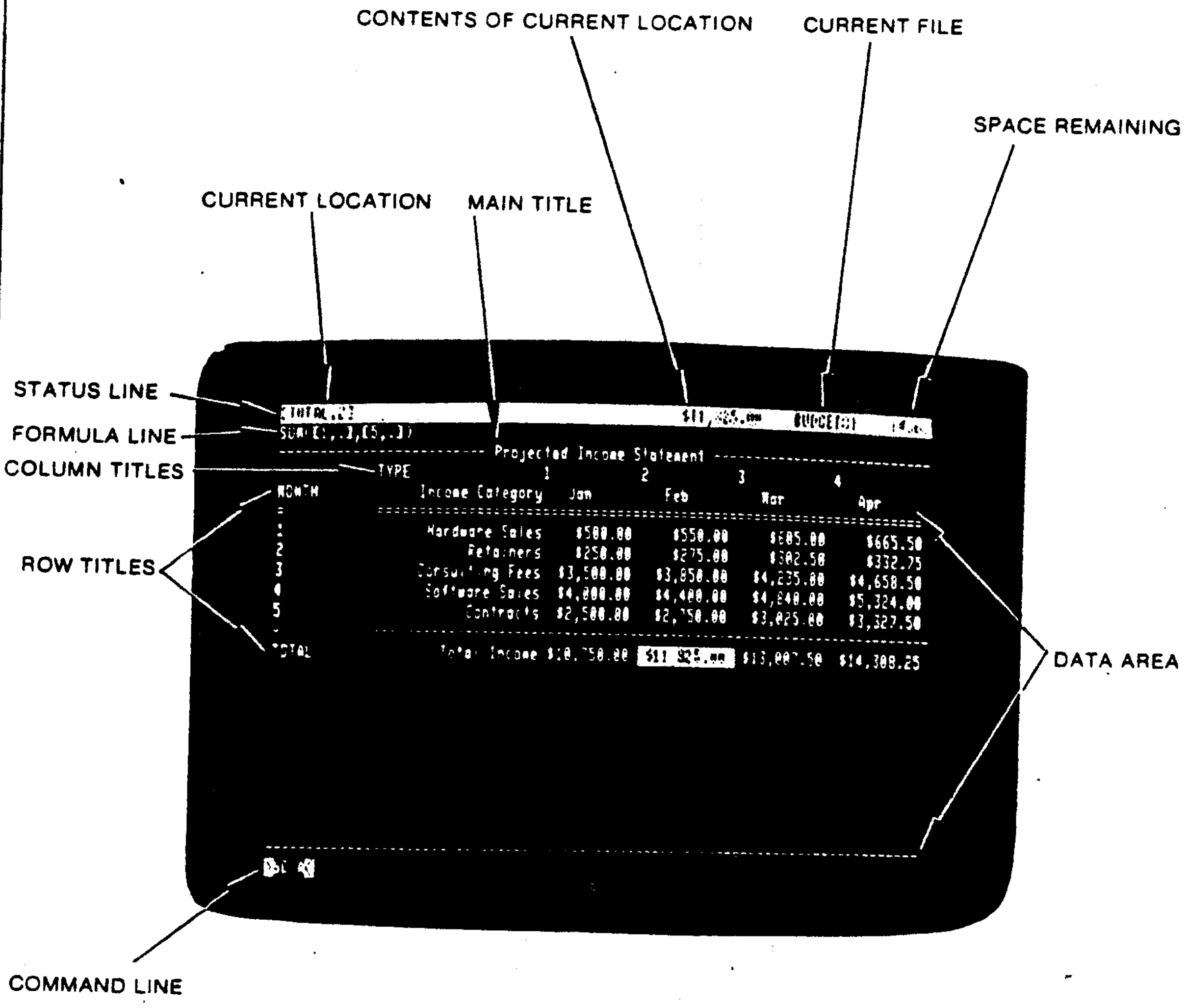
Miscellaneous

A miscellaneous collection of information that may help the user from time to time.

Appendix

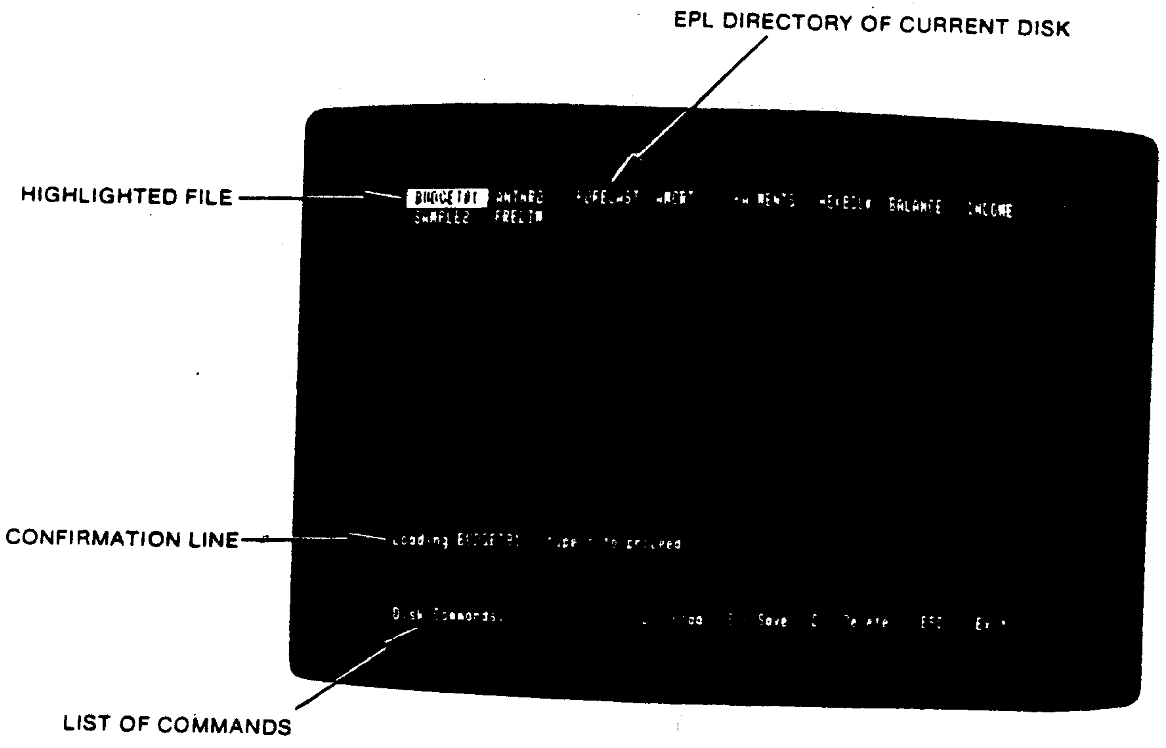
Sample Screens

Some sample screens to assist in understanding how various information is displayed and where it is displayed at.



```
[TOTAL,2]                                          $11,825.00   BUDGET[1]    [illegible]
SUM([1,.],[5,.])
---------------------------- Projected Income Statement ----------------------------
        TYPE                      1            2            3            4
MONTH         Income Category   Jan          Feb          Mar          Apr
=       ====================================================================
1               Hardware Sales    $500.00      $550.00      $605.00      $665.50
2                    Retainers    $250.00      $275.00      $302.50      $332.75
3              Consulting Fees  $3,500.00    $3,850.00    $4,235.00    $4,658.50
4               Software Sales  $4,000.00    $4,400.00    $4,840.00    $5,324.00
5                    Contracts  $2,500.00    $2,750.00    $3,025.00    $3,327.50
-       --------------------------------------------------------------------
TOTAL             Total Income $10,750.00   $11,825.00   $13,007.50   $14,308.25
```

Figure 1: Sample Display

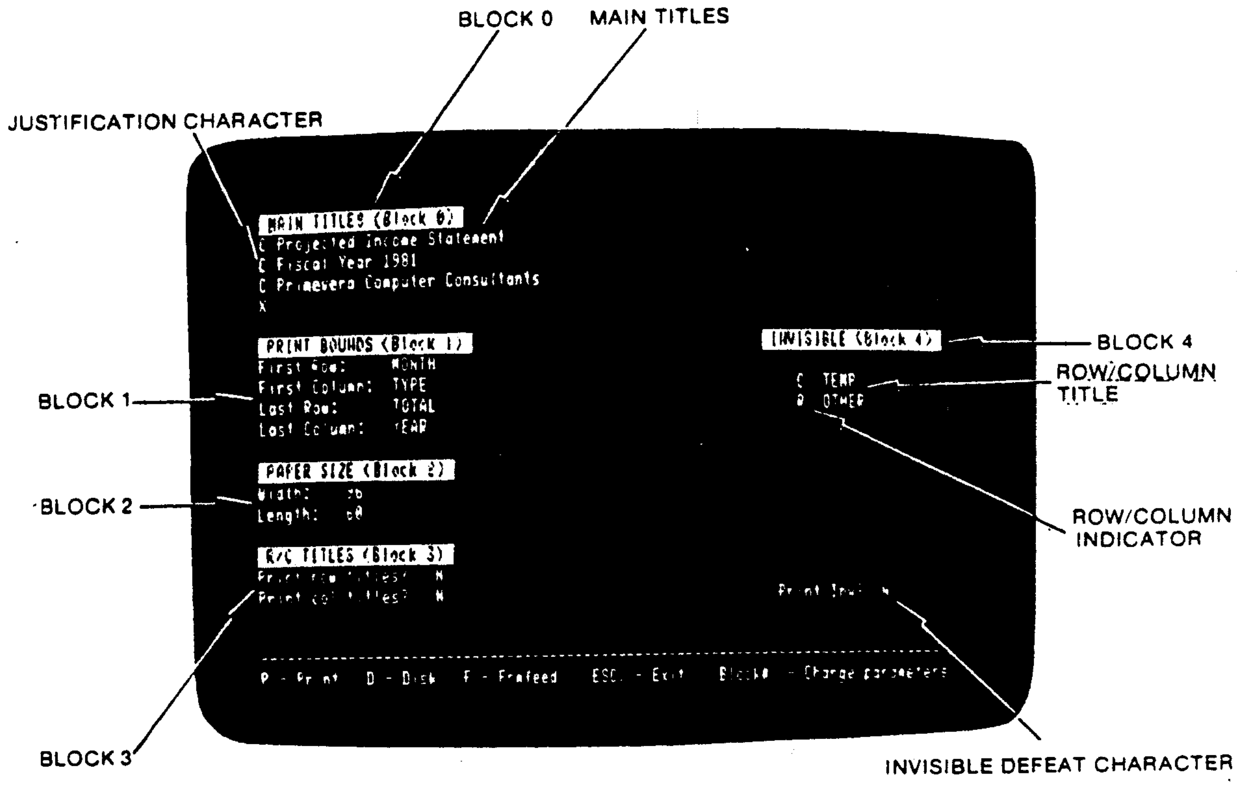


```
BUDGET01 ANTHRO   FORECAST [illegible]   [illegible]   [illegible] BALANCE   INCOME
SAMPLE2  PRELIM

Loading BUDGET01   Type [illegible] to proceed

Disk Commands:              L  Load   S  Save   D  Delete   ESC   Exit
```

Figure 2: Sample Disk Screen



```
MAIN TITLES (Block 0)
C Projected Income Statement
C Fiscal Year 1981
C Primevera Computer Consultants
X

PRINT BOUNDS (Block 1)                              INVISIBLE (Block 4)
First Row:      MONTH
First Column:   TYPE                                   C  TEMP
Last Row:       TOTAL                                  @  OTHER
Last Column:    YEAR

PAPER SIZE (Block 2)
Width:    96
Length:   60

R/C TITLES (Block 3)
Print row titles?   N
Print col titles?   N                              Print Inv?  N

-------------------------------------------------------------------------
P - Print   D - Disk   F - Frmfeed   ESC - Exit   Block# - Change parameters
```

Figure 3: Sample Print Screen

INDEX

Memory Utilization

This information is provided so that the user may have some idea of how memory is used under ExecuPlan. This information is somewhat advanced, and if you don't understand it, don't worry, it doesn't matter.

In order, ExecuPlan keeps the following tables: Column widths, Row titles, Column titles, Primary addresses, Numbers, Formulas, and Strings. All of the tables start from the end of ExecuPlan and build up, except for the strings, which start at the end of memory (actually the base of the BDOS in CP/M) and build down.

The column width table takes one byte per column, that byte being the width of the column.

The row and column title tables take eight bytes per title.

The primary address table, which is used as a giant reference table for the array, takes three bytes per location on the array. The first two bytes are a relative pointer to the actual data in memory, the last byte is the format byte for that location.

The numeric table holds all of the numbers. This table is dynamic, that is, only as many numbers as are actually in the system are kept. Numbers take eight bytes each, and are stored in Microsoft double precision floating point format, which yields 16 digits of precision.

The formula table holds all of the formulas. Unlike the numeric table which is pointed to, the formula table is fully independent. Only the beginning is pointed to. Each formula has a length byte, a destination row and column (taking two bytes), the text of the formula, and then a termination byte. Therefore, formulas take up the number of bytes in the text, plus four.

Text, known to a computer as strings, is stored in the string table. The string table is simply sequentially allocated, down from the top of memory. The strings are stored in reverse order since the table builds down. There are no overhead bytes with strings (the end is indicated by bit 7 on, and the beginning is pointed to from the primary address table), strings take only as many bytes as the string is long.

System Recovery

From time to time, some type of error might occur that will result in the user being dropped out of ExecuPlan into CP/M or the Monitor.

The most common possibility is that you might accidentally try to access a disk drive that does not exist. CP/M will respond with a message like

```
BDOS Error on D: Bad Sector
```

or something like that. This type of error is called "fatal", because there is no direct way to recover from it. Another possibility is that you might accidentally hit the reset button.

In any case, the probability is good that there was something you were working on that you don't want to lose. Therefore, it is nice to be able to recover from these conditions.

Monitor

From the Monitor, type "G 0100" and see what happens. Chances are, you should be right back in ExecuPlan. You might have to type "JB" to clean up the screen.

It is suggested that you dismount the disk just in case. There could be a possibility that the memory image is goofed up, and that might cause crazy things to happen. Better safe than sorry.

CP/M

More common is the case where you get dropped back into CP/M because of a disk error or read-only error. When you get the message mentioned above, or one like it, hit <Return>. You will then be back in CP/M. Now, type the following command:

```
A>SAVE 0 HOPE.COM
```

What this does is to create an empty file on the disk, without disturbing the memory image. If you already have a file called HOPE.COM on the disk, use another name. Now type "HOPE". With any luck, you will be back in ExecuPlan and can continue. If this does not work, reset the computer and proceed as explained in the above section. If that doesn't work, there is probably no recovery possible.

Foreword

Welcome to the world of ExecuPlan!

This reference guide is intended to provide all of the reference information needed to help you to use ExecuPlan to its fullest capability. Sections are provided covering how to bring up the system, read the screen, type and edit commands, enter data, and so forth. Complete explanations of all commands are provided. A section covers all of the errors that might occur and what they mean. A complete section is devoted to the math capabilities, entering formulas, how formulas are evaluated, functions, and so forth. Finally, some helpful information is provided for special circumstances, and the appendix has some annotated sample screens.

This guide is actually not intended for the first-time user. The accompanying guide, ExecuPlan Primer, is the most basic manual. It covers beginning concepts and moves step-by-step to more complex models.

It is recommended that first the primer be read and understood, then this reference guide read and understood. Once the concepts are understood, the reference guide will then be useful for looking up information as needed. This guide does cover most topics in more detail than the primer, but they are explained in a more direct manner, assuming that the reader has some familiarity with the system.

It cannot be stressed enough that you should READ this guide. There will be countless occasions where you aren't sure about something, or don't understand why what happens happens, or something else along that line. If this is the case, get out this reference guide, or the primer if necessary, and look it up. Almost always the information that you need will be found in moments.

To briefly preface the sections of this guide: Section 1 contains general information, such as bringing the system up, understanding the array, reading the screen, understanding titles and references, and typing and editing commands. Section 2 gives detailed explanations of all of the commands. Section 3 covers all errors. Section 4 explains the math package, how formulas are evaluated, how operators and functions work, and other math-related information. Section 5 just has some miscellaneous information.

Again, congratulations on your acquisition of ExecuPlan, and we hope that it will be beneficial to you in whatever applications you have in mind.

Acknowledgment

The ExecuPlan program and this reference guide were completely designed and written by Neale E. Brassell exclusively for Vector Graphic Inc.

The program was developed on a Vector Graphic VIP computer system with two drives, a Sprint 3 printer, and a Teletype model 40 line printer, utilizing the SCOPE editor, ZSM assembler, and RAID debugging system. The documentation was written with the MEMORITE III word processing system

The source program consists of nearly a quarter million bytes of source code, contained in fifteen program modules totaling about 170 pages. After assembly, the object program is about 20K long. Additionally, the help screens occupy another 54K data file.

The floating point math routines and the single-argument functions are derived from Microsoft BASIC version 5.1 which is licensed from Microsoft Inc. The balance of the math routines (parser, expression evaluator, and multi-argument functions) and the balance of the program are all original code.

The author would like to thank the software staff at Vector Graphic for their support and suggestions on the development of ExecuPlan. I am particularly grateful to Chris Cory, who contributed many significant ideas, was instrumental in the debugging of the program, and who wrote the primer and developed all of the sample applications that come with ExecuPlan.

STARTING UP

ExecuPlan is a CP/M-based program, supplied on a CP/M diskette. To execute the program, insert the diskette into the drive and type

```
A>EPL          (You type what's underlined)
```

After a few seconds the screen will display a large banner identifying ExecuPlan, and giving the revision and copyright notice. Type any character on the keyboard. The banner will be erased and an initialized array will be shown on the screen. You're up and running!

At the time EPL is loaded, you may also tell the system to automatically load or create a file on the disk. For example, to run ExecuPlan and load the file BUDGET81, you would type

```
A>EPL BUDGET81
```

After the banner is erased, the system would display the message

```
Loading BUDGET81
```

and proceed to load the program. After the program is loaded, the array containing the data will be displayed.

One other capability when EPL is executed is the ability to create a new file. The command is typed the same as to load a file. If the file does not exist, then the system will display the message

```
Creating BUDGET81
```

and proceed to create the file. The array displayed will be blank, but the file specified will be created, and when the array is updated it will be written into that file.

THE ARRAY

Concept

ExecuPlan is built around the concept of an electronic array. This array contains a number of rows and columns. Information may be put into each intersection of a row and a column. Then, information put into the array can be used to compute other information, which is also put into the array. Finally, the array may be printed in a report fashion.

Each intersection of a row and column is called a "coordinate" or "location". If, for example, the array contains 20 rows and 20 columns, then there would be 400 locations in the array.

References

Given all of these locations, a method must be used to refer to each of them. This is called a "reference", and has the following format:

[Row,Column]

This is the scheme used throughout ExecuPlan to reference locations. For example, a reference to row 10 and column 8 would be

[10,8]

Each row and column has a "title", which is used in a reference. In the preceding example, "10" is the row title, and "8" is the column title. Titles do not have to be numbers, however. A reference might be

[INCOME,JANUARY]

which would refer to row INCOME and column JANUARY. A command is provided for changing the titles of rows and columns.

In the next section, which talks about the screen, the idea of a "current" location will be brought up. For the purpose of references, suffice it for the moment to say that there is a location in the array that is considered to be the current one. From this comes the idea of "relative references". A relative reference is one where rather than specifying the title of a row or column, the distance away from the current location is specified. The character period (.) is used to mean the current location. For example, the reference

[.,.]

means "the current row and the current column", that is, the current location. The reference

```
[.,.-1]
```

means "the current row, and the column that is 1 to the left of the current". If the current location is [4,4], then the aforementioned reference would point to [4,3].

Following are some examples of relative references.

[.,.-5]	Current row, current column - 5
[7,.]	Row "7", current column
[.-1,JAN]	Current row - 1, column JAN

THE SCREEN

This section explains how the screen looks and what all of the things on the screen mean. It is suggested that you look at the picture of a sample screen in the appendix while reading this section.

Concept

Conceptually, the screen is a movable "window" over the array. When the system is initialized, that window is over the top-left corner of the array. Using the arrow keys, the window can be moved over any section of the array. The portion of the array that appears under the window is always 18 rows (unless there are less than that many) by however many columns will fit. Initially, columns have a width of 16 characters, so 4 will fit on the screen. Columns may have almost any width, however, so the number of columns displayed will vary from 1 to 34. If a column's full width will not fit on the screen, it will not be displayed.

Standard Information

At this point, let's take a look at the screen layout.

FIRST LINE: The first line (which is highlighted) is the "status" line, because it tells you the status of the system. Four things are displayed on the status line. 1: The current location. This is the location that the cursor is currently on (see below for additional explanation of the cursor). 2: Contents of the current location. This is simply whatever is in the current location, displayed according to the format of that location. 3: The current file, if any. This area might be blank. If it is not, then it contains the name of the current file, which is used if the file is updated. 4: The amount of space still available in memory. Initially, this is about 30,000 characters.

SECOND LINE: The second line is the "formula" line. If the current location contains a formula, that formula will be displayed on this line. If there is no formula associated with the current location, this line will be blank.

THIRD LINE: This is the "main title" line. Initially, this line contains a line of dashes. When main title #1 is set (see T command), then that title is centered in the field of dashes and displayed on this line.

BOTTOM LINE: This is the "command" line, where commands are typed. This will be explained in detail in the next section.

Data Area

The area between the main title line and the row of dashes above the command line is the data area of the screen. It, in turn, actually has two sections, the titles and the data.

The titles are displayed across the top of the data area and down the left side. The COLUMN titles are across the top, and the ROW titles down the left side.

The data itself fills the remainder of the area. If there is no data in the system, this area will be blank.

It is somewhat difficult to explain, but very easy to understand, the layout of the data in the data area. Essentially, each intersection of a row and a column (each LOCATION) can contain an item of data. If you visually trace a horizontal line across from the row title, and visually trace a vertical line down from the column title, then where those lines intersect will be where the data contained in that location will be displayed. For an easy example, the data contained in [1,1] will be the top-left corner; data contained in [18,4] (on an initialized system) will be in the bottom-right corner.

The Cursor

The "cursor" is the name given to the large white rectangle that is somewhere in the data area. The position of that cursor, taken as a reference, is called the "current location", a term that has been and will be used OFTEN. If the cursor in in the top-left corner (again, of an initialized system), then it is in location [1,1], and therefore [1,1] is said to be the current location. When the arrow keys are used (or a couple of other keys, all explained later), the cursor moves. If it is in the top-left corner, then pressing the <down arrow> key will move the cursor to the second row. Pressing it again will move to the third row, and so forth.

Moving the Cursor

As hinted at above, the cursor is moved around on the screen with the <arrow> keys. Here, we'll go into more detail on this.

The arrow keys move the cursor in the appropriate direction. When the cursor reaches the edge of the screen, then instead of moving the cursor, the whole screen is moved. More specifically, the window that the screen represents moves with respect to the entire array. If column 4 is the rightmost column on the screen and the <right arrow> key is pressed, the first column on the screen will be shifted off and the next column to the right will be shifted on. Then, if the <left arrow> key is pressed, only the cursor will move. Repeated pressings will move the cursor to the left until it

is on column 2, which at this point would be the leftmost. The next pressing would shift the screen left and move column 1 onto the screen.

When the cursor is at the extreme point in any direction and that arrow key is pressed, the system will simply ignore it.

In addition to the regular arrow keys, the <up arrow> and <down arrow> keys may be shifted to move the screen 18 rows (a screenful) at a time. For example, if the cursor is on row 1 and <shift down arrow> is pressed, the cursor will be on row 18, which would be the beginning of the second screenful of rows. The exception to this is if that action would result in less than 18 rows being displayed. In that case, the screen will be shifted as far as possible, but not so far as to result in less than 18 rows being displayed.

Finally, there are three other cursor-movement keys. Pressing <Lf> will move the cursor to the leftmost column of the current row, as if the <left arrow> key were pressed repeatedly. Typing <Control-T> will move the cursor to the top of the array and put it at the top line of the data display also. <Control-E> will move the cursor to the end of the array, and put the cursor on the bottom line also.

TYPING COMMANDS

In order to really DO anything with ExecuPlan, you must give it "commands", which are instructions telling it what to do.

There are two directions taken when it comes to giving a computer commands. One is the "language" method. In this method, you create a list of instructions, feed it to the computer, and get back the results. This method is interesting, but it really just amounts to simplified computer programming. This is not an acceptable method of getting results to someone who is not a programmer.

The other method is known as "interaction". In this method, you give the computer AN INSTRUCTION. The computer follows that instruction and displays the result, if any. You then give the computer another instruction, and it carrys that out. In other words, the computer interacts with the user. It is this method that is used by ExecuPlan.

Recall from the last section that the bottom line is the "command" line. It is on this line that commands are typed.

In the very bottom left corner of the screen are two things. These are the "command prompt" and the "command cursor". Before you begin to type a command, these look like ><, only highlighted. The left character, >, is the prompt. The right character, <, is the command cursor, NOT TO BE CONFUSED WITH THE CURSOR IN THE ARRAY. The command cursor tells you where on the line you are at. When you type a character, the character appears on the screen where the command cursor WAS, then the cursor reappears one character to the right.

Notice in the last sentence of that paragraph that the command cursor was refered to simply as the cursor. This might seem confusing, since the white rectangle on the screen in the current location is called the cursor. Well, they are SO conceptually different that you will have no difficulty ascertaining which one is meant when you see the word "cursor".

Characteristics of Command Mode

Yes, when the system is sitting waiting for you to type a command, that is called "command mode". When the system is in command mode, you may type commands (seems reasonable, doesn't it?).

As mentioned above, when a character is typed, it appears on the screen where the cursor was, then the cursor reappears one character to the right. This should not be new, since virtually all software operates in this manner.

At any time during the typing of a command, all of the capabilities of moving the screen (arrow keys, etc.) are available.

When a command has been typed satisfactorily and you wish the computer to carry out that command, the <Return> key must be pressed. As soon as that key is pressed, the system STOPS waiting for you to type, and begins to execute the command.

In addition to normal letters and numbers used when typing commands, certain other "special" characters can be used, which will be explained in the following paragraphs.

Finally, and for lack of a better place to explain it, it should be noted that upper case and lower case letters are fully interchangeable. That is, typing "EVC" is the same as "evc" or "Evc" or "evC" or whatever. Actually, this is true not just in command mode, but everywhere in the system. EXCEPT: Titles do not allow interchanging of cases. That is, "Jan" and "JAN" are different titles.

Special Characters

While typing commands, there are several special characters available. They are of two types. The first is editing, which means that they allow you to "edit", or change, what you've typed. The second is "inserting", meaning that you can "insert" certain things from the system without having to type them.

Each special character will be explained here.

<Bs> *- Typing the <Bs> key (Backspace) will cause the cursor to "back up", effectively erasing the last character that was typed. For example, if you typed "ET HELLI" then realized that you meant to type "O", not "I", then you could hit the <Bs> key which would back up and erase the "I". Then, you'd type "O" and continue.*

<Del> *- Typing <Del> simply erases the entire command that you've typed. It is equivalent to hitting <Bs> repeatedly until every character was erased.*

<Control-K> *- For users with extensive Memorite experience, <Control-K> is the same as <Del>. (Memorite uses <Control-K> to erase the current line.)*

<Esc> *- Hitting the <Esc> key enters "command edit mode". In this mode, more extensive editing of the line is possible. Command edit mode will be explained in following paragraphs.*

All of the above characters are of the editing type. The remainder are the inserting type.

<Control-A> - This will take the current location, make it a reference, and insert it into the command line. If the cursor is at [1,1], for example, then typing <control-A> would insert the characters "[1,1]" into the command line, as if you had typed it. This is useful for grabbing locations to be used in a formula. The rationale behind the character <Control-A> is that it inserts the location that the cursor is AT.

<Control-F> - This will insert the current FORMULA into the command line. Look at the second line on the screen, the formula line. Whatever is there will be inserted into the command if <control-F> is typed. If there is no formula associated with the current location, then nothing will be inserted.

<Control-C> - This will insert the CONTENTS of the current location into the command line. The data inserted will be exactly as it is displayed.

<Control-L> - This will simply insert the entire LAST command typed into the command line. This can be useful for repeating an operation, such as entering data, for several different locations. It also is useful, particularly, when a lengthy command is typed which contains an error. Rather than retyping the entire line, you can simply type <control-L> then use the editing commands, below, on the line.

Keep in mind that the above characters can be combined with the screen movement characters. For example, say you'd like to insert a formula into the command line. Not the current formula, but the formula that is associated with a particular location. You can simply move the cursor to that location, type <control-F>, then move the cursor back to where you want it!

Command Edit Mode

As mentioned above, typing <Esc> enters command edit mode. When in command edit mode, none of the above-mentioned characters work, and none of the screen-movement characters work. This mode is used when you've typed a long command and need to change it, but just don't want to retype the whole thing.

Command edit mode, or simply edit mode, is indicated by having a character of the command highlighted. At first, this will be the rightmost character of the command. The highlighted spot is itself like a cursor, insofar as that is the spot where whatever is done will take place.

While in edit mode, typing characters is just like normal command mode, except that whatever character is typed will replace the character under it.

Certain special characters exist in edit mode, and they are as follows.

<Left-arrow> - this will move the highlight one character to the left.

<Right-arrow> - this will move the highlight one character to the right. It will not move any further than the cursor.

<Lf> - this will move the highlight to the first character of the line.

<Control-V> - this will enter "insert mode". At this point, when normal characters are typed, they will not replace the character under the highlight. Instead, the remainder of the line will be shifted to the right and the character typed will be inserted. Typing <control-V> a second time will leave insert mode.

<Control-D> - this will delete the character under the highlight. The remainder of the line will be shifted to the left.

<Esc> - this will leave edit mode and return to the normal command mode.

<Return> - this is like typing <Esc> then typing <Return>. That is, it will leave edit mode, but then proceed as if you had typed <Return> in command mode and execute the command.

Forced References

Throughout this guide, the expression "current location" will be used. As previously explained, this means the location where the cursor is at.

There is a way to make the current location somewhere else, if desired. For example, the C command clears the current location. You might wish to clear location [20,5], but the current location is [1,1]. You could just move the cursor to the desired location and then clear it, but another method is provided.

This method is called "forcing" the current location. That is, you can make the system think that somewhere else is the current location. This is done by simply typing the reference before the command. For example, you could type

```
[5,20] C
```

Which would force [5,20] as the current location, then execute the C command which clears the current location, which would be [5,20]. Another example:

```
[INCOME,JAN] EVC 1000 1500 2000
```

This would force location [INCOME,JAN] and then execute the EVC command, which will put the values into successive locations, starting at the current location. Of course, the current location would be what it was forced to be.

It should be noted, though, that the forced reference is only applicable to the command with which it is typed. After execution of the command is complete, the spot where the cursor is at will again be the current location.

In general, anywhere in this guide where any of the expressions "current location", "current row", or "current column" are used, they refer to the current as defined, unless a forced reference is used, in which case they refer to the forced current location.

Manual Recalculation

The system has two modes for calculations, automatic and manual, which are selected with the SC command (which is explained in another section). When in automatic mode, entering any formula or value will cause the system to reevaluate the array. In manual mode, this will not happen. To cause a recalculation when in manual mode, press the <Tab> key. You must do this as the first character on the command line - if you are in the middle of a command, the system will ignore you. The message -Recalculating- will appear in the bottom right corner of the screen while the system is doing the calculations. When it is done, you will be returned to command mode.

Note that it is possible to have circular references such that it takes two or more recalculations before the array is fully evaluated. You can simply keep pressing the <Tab> key as often as needed.

Section 2

Commands

Complete detailed descriptions of all commands and how they are used.

C - the CLEAR command

The clear command is used to clear certain locations in the array of the data and/or formulas associated with them. The various forms are as follows.

C - Clear current location. The data and formula will be cleared from the current location.

CF - Clear current formula. If there is a formula associated with the current location, it will be erased. The data in the current location will be left unchanged.

CR - Clear row. The data and formulas for all locations in the current row will be erased. Alternate form:

```
CR row
```

In this case, (row) specified will be cleared.

CC - Clear column. The data and formulas for all locations in the current column will be erased. Alternate form:

```
CC column
```

In this case, (column) specified will be cleared.

CA - Clear all. All data and formulas in the array will be erased.

D - the DISK command

The disk command is used to access disk directories to load, save, or erase files. It is also used to select the current disk to be used for subsequent disk operations.

D - Disk commands. Disk command mode will be entered, using the currently-selected disk.

D x - Disk select/commands. Disk x will be selected, then disk command mode will be entered. The value x must generally be in the range A to P, but specifically must be a valid drive on the system being used at the time.

Disk Command Mode

When disk command mode is entered, the directory of EPL files on the selected disk will be displayed at the top of the screen; the current disk and a list of available commands will be displayed at the bottom. Furthermore, the first file name will be highlighted.

At this point, typing any of the four arrow keys will move the highlight in the appropriate direction. Once the highlight is moved to the desired file, the disk commands below can be used. All disk commands reference the currently-highlighted file.

Disk Commands

L - Load a file. The highlighted file will be loaded into the array. Note that this erases the current array; therefore, if it is desired to save the current array, it should be updated or saved before executing the load command.

S - Save a file. The array will be saved to the highlighted file. The former contents of the file are lost. This command should be seldom used, and is included only for symmetry. Normally, the update command is used to save the array.

D - Delete a file. The highlighted file will be erased from the disk. This is equivalent to using the ERA command under CP/M.

<Esc> - Exit disk command mode. When <Esc> is typed, the user will be returned to the main system.

Each of the disk commands (except <Esc>) requires confirmation. When either L, S, or D is typed, one of the messages

```
Loading <file> - type Y to proceed -
Saving <file> - type Y to proceed -
Deleting <file> - type Y to proceed -
```

will be displayed. At this point, typing Y will cause the selected action to be carried out. Any other character, including <Return>, will cancel the action.

If when disk command mode is entered, there are no EPL files on the disk, the message

```
No files...
```

will be displayed. Obviously, there are no files to load or delete; therefore the only possible action is to save the array, which in this case MUST be done with the update command. Typing any character will return to the main system.

E - the ENTER command

The enter command is used to enter data or formulas into the array. It has four forms, for entering text, lines, values, or formulas. Since the four are drastically different, they will be explained separately.

ET - Enter Text

The enter text command will accept a single argument and write it into the current location. The general format is

```
ET text
```

The format for the ET command is somewhat precise. There must be one character after the ET, then whatever is after that is considered the text. For example, in the command

```
ET Hello There<                 (the < indicates the cursor)
```

will enter the text "Hello There". On the other hand, the command

```
ET  Income Type <
```

will enter the text " Income Type ". In other words, exactly what you type is what you get. Since whatever the user types is taken as the text, no multiple form of the ET command is possible (unlike the other enter commands).

EL - Enter Line

The enter line command is used to enter a line of data where the data is simply one character repeated. This is typically used for a dividing line of some sort, or perhaps the line after a column of numbers above the total, or something similar. There are two formats of the enter line command:

```
EL character
EL character ncolumns
```

The first type will create a line consisting of the (character) all the way across the array. The second will create a line only extending across a certain number of columns, specified by (ncolumns). In either case, the current location will be used as the first (leftmost) column; that is, the column in which the line begins.

The EL command operates much like the ET command. The data in the line is *simply treated by the system like any other text. The EL command automatically creates the sequence of characters to be the exact width of each column as it is entered. This can be used to create a "broken" line, also. For example, if the widths of all columns are set to 10, then the EL command is used, then the widths of all columns are set to 12, there will be a 2-character break in the line between each of the columns.*

EV - Enter Value

The enter value command and its variations are used to enter numeric values into the current location and possibly adjacent locations in the array. The simplest form is

```
EV number
```

which will write the value (number) into the current location. Additionally, this form of the command can be "implied" by simply typing

```
number
```

when in command mode. That is, typing "123" and "EV 123" are the same.

The second form of the EV command is used to enter values into adjacent locations, either across a row or down a column. The forms are

```
EVR number number number number etc.
EVC number number number number etc.
```

The first will enter as many successive numbers as are typed into the current location and successive columns on the same row; the second will do the same, only the numbers will be entered into successive rows on the same column. For example, if the current location is [1,1], the command

```
EVC 1 2 3 4 5
```

will enter 1 into [1,1], 2 into [2,1], 3 into [3,1], etc.

The final form is the "repeat" form. Is is used to enter the SAME number into successive locations. The forms are

```
EVRR number count
EVCR number count
```

The first will enter the number (number) into successive columns on the same row for (count) columns; the second will do the same down a column. For example, if the current location is [1,1], the command

```
EVCR 100 5
```

will enter the value 100 into locations [1,1] through [5,1].

EF - Enter Formula

The enter formula command is used to enter formulas into the array. It has two forms, single and multiple. The single form is

```
EF formula
```

which will enter the (formula) into the formula table. It will set the value in the current location to 0, unless a "SC A" has been done (see the set command), in which case the formula will be evaluated and the result of the evaluation put into the current location.

The multiple form of the EF command is

```
EFM formula nrows ncolumns
```

where (nrows) and (ncolumns) are the number of successive rows and columns, respectively, to write the formula into. The array section which will get the formula may be thought of as a rectangle consisting of (nrows) rows and (ncolumns) columns, with the current location as the upper-left corner. For example, if the current location is [1,2], then the command

```
EFM 1.1*[.,.-1] 8 11
```

will write the formula "1.1*[.,.-1]" to 8 rows and 11 columns, or specifically, into locations [1,2] through [1,12], [2,2] through [2,12], and so on through [8,2] through [8,12].

F - the FORMAT command

The format command is used to choose the manner in which the data in the array will be displayed and printed. There are four forms of the command, but they differ only in the portion of the array that they affect. They are

F format
FR format
FC format
FA format

F means to format only the current location. FR means format the current row, FC means the current column. FA means format the entire array.

The (format) consists of zero or more individual format characters. These characters can be listed in any order, in a generally "free" fashion. Following are the various format characters.

$ - Dollar Sign. If $ is included, numbers will be printed with a leading dollar sign. For example, 123 will be displayed as $123.

, - Comma. If a comma is included, numbers will be printed with commas inserted every three digits to the left of the decimal point. That is, the value 123 would be unchanged, but the value 1234567 would be 1,234,567.

0-15 - Digits. Including a number in the range 0 through 15 will set the number of digits printed to the RIGHT of the decimal point. For example, the value 123 with a format of 4 would be printed as 123.0000.

% - Percent. The percent character indicates that the value is to be considered a percentage, and it will be printed with a percent sign following it. In addition, the number will be multiplied by 100 before being displayed. For example, the value .13 will be displayed as 13%.

R - Right Justify. For text, the R character causes right justification.

Each location in the array is either "formatted" or "unformatted". Typing a format command without any arguments (for example, "FC ") sets a location or locations to "unformatted". In this case, numbers and text will both be displayed left justified. Numbers will be displayed in a "general" format, meaning however necessary to express the value (For example, 100 will be 100, 3.14159 will be 3.14159).

The only format option for text is R, right justify. Its absence indicates left *justification.*

For numbers, formatting is a little more complicated. ANY numeric format sets the location to "formatted". When a location is formatted, numbers are RIGHT justified. This right justification should not be confused with the R character for text.

Each time a F command is used, it overrides any previous F command. For example, if a "F 2" command is issued, the current location will be set to 2 places to the right of the decimal point, typically used for "dollars-and-cents" notation. If it is then desired to add the dollar sign, the command "F $" will NOT function as expected, since it cancels the effect of the "F 2" command. The proper command would be "F 2 $" since this combines the two commands. This brings up an important point: Leaving out the number-of-digits character in a format command is the same as using 0; that is, "F $" is identical to "F $ 0".

As was indicated above, the format characters are entered in a "free" fashion. Their order is unimportant. For example, the commands

F 2$,   F $,2   F ,$2   F $, 2   and F 2 , $

are all identical. Format characters may be used in whatever combination desired, except that the combination "$%" will produce a meaningless figure; for example, 123.45 will be displayed as $12345%.

Since there is limited space available for format characters, the R and , characters are actually the same thing. That is, using a comma on text will right justify it, and using R on a number will insert commas. Since a location cannot contain both text and a number, this should not cause any problem.

H - the HELP command

The help command is used to access a screenful of assistance (commonly called a "help screen") for a particular command. There are actually three different forms of the help command.

```
H
H letter
?
```

Just typing H gives a help screen on typing commands, editing, and moving the cursor. Typing H followed by a letter gives a help screen on the command beginning with that letter. For example, HF gives help on the format command.

The help command reads the help screen from the file EPL.SYS on the currently-logged-in disk. That disk should not be confused with the currently-selected disk used for ExecuPlan! The logged-in disk is the disk that CP/M thinks is the current one. To be more specific, the disk that was in the prompt before ExecuPlan was executed. If it was A>, then the logged-in disk is drive A; if it was B>, then it was drive B, etc. If the file is not present on the disk, then the system will say "Help Unavailable".

The ? command is used to get a QUICK help screen. It tells how to get more help (via the H command) and gives a list of the command letters and their meanings. This screen is part of the program, not read in from the disk. Therefore, it is always available.

I - the INITIALIZE command

The initialize command is used to set everything back to the standard. Specifically, the command

Resets the array size to 20 x 20,
Resets the row and column titles back to 1 2 3 4 5 etc.,
Clears all main titles,
Clears the entire array,
Sets the current file to none,
and Resets the print blocks back to standard.

The format of the command is simply

```
I
```

It will display a warning message,

```
** Warning: Initialization erases ALL data - type Y to proceed -
```

at which point you may type Y to proceed with the initialization. Typing any other character (including <Return>) will cancel the command.

J - the JUMP command

The jump command is used as a quick way to move the screen around on the array, faster than using the arrow keys. There are three jump commands:

```
JB
JR row
JC column
```

The first, JB, simply jumps to the top-left corner of the array, which would be, [1,1] on an initialized array.

JC jumps to the specified column. That column will be the leftmost on the screen after execution.

JR jumps to the specified row. The row will be the top on the screen after the jump, unless the size of the array makes this impossible. If the jumped-to row is within 18 rows of the end, it will be somewhere in the middle of the screen.

K - the KILL command

The kill command is used to "kill", or remove, a row or column from the array. It does not, however, change the size of the array. Therefore, when it kills a specified row or column, it creates a new one at the end in order to keep the array the same size.

The format of the kill command is

```
KR row new-row
KC column new-column
```

where (row) or (column) is the row or column to kill, and (new-row) or (new-column) is the title to be assigned to the row or column created at the end of the array.

Example: If your array currently has 12 rows, numbered 1 through 12, and you execute the command "KR 6 13" then your resulting rows will be 1,2,3,4,5,7,8,9,10,11,12,13.

There is one VERY IMPORTANT thing to note about the kill command! Relative references in formulas which refer to or over the killed row or column will NOT be changed. In other words, they will be INCORRECT after the command is executed. In the example above, if you had a reference in row 7 which contained something like .-2, before you executed the kill command, that would have pointed to row 5; after the command, it will point to row 4.

L - the LIST command

The list command is used to produce a list on the printer of all of the formulas associated with the array. The list is printed in the order in which the formulas will be evaluated; that is, either in row-major or column-major order, depending on the current "SC R/C" setting. The format of the command is simply

```
L
```

The listing will have the heading at the top of each page

```
FILENAME      Formula list by row
                         (or column, if appropriate)
```

where FILENAME is the name of the file if one is assigned. The format of the listing is

```
[destination]      = formula
```

where [destination] is the destination of the formula, and (formula) is the text of the formula.

If during the listing it is desired to stop, typing <Esc> will cancel the command.

M - the MOVE command

The move command is used to move a row or column from one place in the array to another. The command format is

```
MR row dest-row
MC column dest-column
```

where (row) or (column) is the row or column to move, and (dest-row) or (dest-column) is the row or column to move it adjacent to.

Depending on which direction the row or column is moved, it will either be placed above/to the left of the destination, or below/to the right. Specifically: for a COLUMN, if it is being moved to the left, is will be placed to the left of the destination; if it is being moved to the right, it will be placed to the right of the destination. For a ROW, if it is being moved up, it will be placed above the destination; if it is being moved down, it will be placed below the destination.

For example, say you have the rows 1,2,3,4,5,6. If you execute the command "MR 5 3", the resulting sequence will be 1,2,5,3,4,6. If you had executed the command "MR 2 6", the resulting sequence would have been 1,3,4,5,6,2. This example is equally applicable to columns.

NOTE: Like the K command, the M command does not change relative references in formulas.

O - the OPEN command

The open command is used to open up a new row or column in the array. The command does NOT change the size of the array, therefore when a new row or column is created, the last row or column of the array is removed, and its contents lost.

The format of the open command is

```
OR ref-row row
OC ref-column column
```

where (row) or (column) is the title of the new row or column to create, and (ref-row) or (ref-column) is where to put it. If COLUMN, the new column will be to the LEFT of the ref-column; if ROW, the new row will be ABOVE the ref-row.

Example: If there are currently 10 rows, numbered 1 through 10, then executing the command "OR 7 NEW" will result in rows 1,2,3,4,5,6,NEW,7,8,9 with row 10 being lost.

NOTE: Like the K and M commands, the O command does not change relative references within formulas.

P - the PRINT command

The print command is used to print the array, or to cause what would be printed to be written into a disk file for editing with Scope or Memorite. Like the disk command, the print command is actually an entire command mode. It is invoked simply with

P

Print Command Mode

When print mode is entered, the screen will be erased and replaced with what is called the "print screen". The screen is divided into 5 "blocks", each one controlling certain aspects of what will be printed.

Block 0 - Main titles. This block is used to select how the main titles will be printed. For each of the four main titles, which will be displayed, the choice may be made whether to R - right justify, L - left justify, C - center, or X - not to print at all.

Block 1 - Print bounds. This is used to select the portion of the array to be printed. Specifically, the starting row, starting column, ending row, and ending column are specified. By proper manipulating of these bounds, an array much larger than a piece of paper can be printed on several sheets, then the sheets rearranged to form a large sheet.

Block 2 - Paper size. This informs the system of the size of paper being used, in terms of number of characters per line and number of lines per page. The width is used only for centering the titles, but the length tells the system the maximum number of lines to print on one page before skipping to the next page.

Block 3 - Row/column titles. This block allows the user to choose whether or not to have the system print the row and column titles on the report.

Block 4 - Invisible. This allows the user to set rows and columns to "invisible", meaning that they will NOT be printed, even if they are within the print bounds selected by block 1. This is most often used to prevent the printing of some type of intermediate result column. There is also a provision for overriding the invisible function, that is, to go ahead and print the invisible rows and columns.

Print Commands

In addition to the blocks, the bottom of the screen will list the available commands. Following are the commands, and how to use the blocks.

P - Print the array. *Typing P will cause the array to be printed, using the settings of the blocks to define the bounds, titles, etc. If during printing you wish to stop, type <Esc>.*

D - Disk. *Typing D is just like P, only instead of printing the array, the data will be written to a disk file. The format of the data will, however, be identical to when it is printed.*

When D is typed, the message

> *Please enter file name:*

will be displayed. Type the name of the file that you wish to write the data to, followed by <Return>. The file will be assumed to have .MEM as the extension, and must not already exist. If it does, an error will be displayed and the command cancelled. The current disk will be used for the file; to use a different disk, first select it with the disk command from the main system.

If when you are prompted for the file, you decide not to execute the command, simply type <Return> without typing the file name, and the command will be cancelled.

F - Formfeed printer. *Typing F will simply cause a formfeed character to be sent to the printer. This normally has the effect of rolling the paper up to the top of the next page.*

0 - Edit block 0. *Typing 0 will allow you to change the information in block 0. Note that only the justification character can be changed at this time. To change the text of the title, the TMx command must be used when under the main system.*

When 0 is typed, the justification character of the first main title will be highlighted. At this point, you have several options:

> *Type <down arrow> - this will move the highlight down to the next title. If you are already at the bottom (fourth) title, the highlight will be moved back to the first one. The justification character for the current title will not be changed.*
>
> *Type <up arrow> - this is the opposite of <down arrow>. The highlight will be moved up to the previous title. If you are already at the top, it will be moved down the the bottom one. The justification*

character will not be changed.

Type <Esc> - this will leave the current character unchanged, and return to print command mode.

Type <Return> - this is identical to <down arrow>, except that if you are at the bottom title, it will stop editing block 0 and return to print command mode, similarly to <Esc> above.

Type a justification character - typing either L, R, C, or X will set the justification character for the current title to whatever is typed. The highlight will not be moved, so if the wrong thing is typed, you may simply retype the correct character.

Anything other than the above characters will simply be ignored.

1 - Edit block 1. *Typing 1 will allow you to edit the information contained in block 1. When 1 is typed, a cursor will appear a little to the right of the first line in block 1. At this point, you are in a mode similar to block 0, but a little different. Essentially, while block 0 is an "instant" block, meaning that when you type a character, it immediately replaces the previous character, block 1 is an "updated" block, meaning that the new information appears to the right of the old information and is edited by itself, and only replaces the old information when you leave block 1 and return to the print command mode.*

Of course, the data you type is not a justification character. Instead, the appropriate title is typed. The first time print command mode is entered, the bounds are set to the size of the entire array.

When typing the title, up to eight characters may be typed, terminated by either <Return> or <Esc> (the difference is explained below). In the process of typing the title, <Bs> may be typed to back up one character.

Additionally, there are two special characters allowed, if typed as the first character. Typing <Control-F> will display the "first" row or column of the array; typing <Control-L> will display the "last" row or column. This could be useful if you wish to, say, start the printing at the first row, but you're not sure what the title of the first row is.

The same editing characters are available for block 1 as for block 0 (<down arrow>, <up arrow>, <Esc>, <Return>), but the way they work is slightly different. When by some method the cursor comes to be on a line, any previously-typed data on that line is erased. To change a single item without having the cursor move down the next line (and consequently erase something that you might have typed there), <Esc> may be typed instead of <Return>, which will immediately return to print command mode.

When print command mode is reentered, the information in block 1 will be updated based on the new information typed. If a title typed does not exist, it will be highlighted and an error displayed. Any line which contains an error will not be updated.

Although as explained this probably seems very complicated, it is virtually self-explanatory when actually done.

2 - Edit block 2. *Typing 2 allows the user to modify the information under block 2, the paper size block. All aspects of block 2 are identical to block 1, except that instead of typing row or column titles, you type decimal numbers. For page width, you should type the number of characters per line. This is only used for centering the main titles. For page length, you should type the maximum number of lines you wish printed on a page. On a standard 11" page, a length of 56 lines allows reasonable margins on the top and bottom. For both length and width, the system will accept values in the range of 40 through 255.*

3 - Edit block 3. *Typing 3 allows the user to select whether row or column titles are to be printed as part of the report. If the appropriate line is Y, the titles will be printed; if N, they won't. Block 3 is similar to block 0, except that the items are only updated when print command mode is returned to.*

4 - Edit block 4. *Block 4 is used to determine which rows or columns, if any, are to be considered "invisible", meaning that they will not be printed.*

Block 4 is perhaps the most confusing block, because it includes two individual fields for each line. The first is the row/column indicator, one of the characters R or C. The second is the title of the row or column.

When block 4 in entered, a cursor will appear on the top line of block 4. At this point you have several options.

> *The <up arrow>, <down arrow>, and <Esc> keys function the same as block 0. The exception is that when you are on the bottom (tenth) line, the down arrow moves to the "Print Inv? " field. From there the <down arrow> will move back to the first line. The <up arrow> moves in the same manner, only up instead of down.*
>
> *Typing R or C will set the row/column indicator for the current line to whatever is typed.*
>
> *Typing <Space> will turn OFF the line; that is, when <Return> is typed after the <Space>, it will also remove the title on the current line.*
>
> *<Return> is typed to allow movement to the title field. For example, to*

set a line to Row 1, you would type "R" <Return> followed by "1" as explained below. If the row/column indicator already contained an R, you could simply type the <Return> and then proceed with the "1".

If <Return> was typed, the cursor will jump three spaces to the right. At this point the system is awaiting a title. Type the row or column title that you wish to set to invisible. If the title is terminated with <Return>, the cursor will proceed down to the next line. If it is terminated with <Esc>, block 4 will be updated and the print command mode will be returned to.

When the cursor is moved to the "Print Inv? " field, you may type Y or N, in the same manner as block 3. If N is typed, the invisible function will work, that is, the rows and columns indicated will not be printed. If Y is typed, the function will be effectively overridden, that is, the rows and columns listed will be printed anyway.

When print command mode is returned to, the rows and columns in block 4 are looked up. If any of them are not found, they will be highlighted and an error displayed. However, the information will be left in the block, and when the array is printed, the invalid entries will be ignored.

Characteristics of print command mode.

The information in the print blocks remains as set until changed. In addition, all of the information is saved with the file, so when a file is loaded, all of the information in the print blocks will be the same as when the file was saved.

Q - the QUIT command

The quit command is used to return to CP/M. There are three forms of the quit command:

```
Q
QY
QN
```

If QY is typed, the array will automatically be updated; if no current file exists, you will be asked for one. If QN is typed, the array will not be updated.

If just Q is typed, you will be prompted with

```
Exiting - Type Y to update -
```

If at this point you type Y, the file will be updated and the program exited. If you type N, the file will not be updated, and the program exited. If you type <Esc>, the system will cancel the command altogether. Anything else will be ignored.

R - the ROUND command

The round command is used to change the precision of numbers in the array. Specifically, all numbers in the array that are NOT calculated as the result of a formula will be changed to match their representation on the screen. For example, if the value 1.469 is in a certain location, but the format is 2, then the number is being displayed as 1.47. Therefore, when the round command is executed, the number will actually be changed to 1.47.

The round command is executed simply by typing

```
R
```

at which time the system will warn

```
** Caution: Data will be changed - type Y to proceed -
```

If you type Y, the action will be carried out. Typing anything else will cancel the command without having any data changed.

S - the SET command

The set command is used to set or change certain aspects of the system. There are four set commands; set size, calculation, printer, and disk. These are explained separately.

The Set Size Command

The set size command is used to change the size of the array. The form of the command is

```
SS nrows ncolumns
```

where (nrows) is the number of rows to make the array, and (ncolumns) is the number of columns to make the array. When the system is initialized, the size is 20 x 20. To change to a 9 x 13 array, for example, the command would be

```
SS 9 13
```

If the number of rows or columns is being decreased, the message

```
** WARNING - Some rows containing data may be lost - type Y to
proceed -
```

will appear. If the number of columns is being decreased, the message will indicate columns instead of rows, of course. Typing Y will cause the action to be carried out; anything else will cancel the action. If you are decreasing both rows and columns, you will get both messages, one at a time. Responding Y to rows but something else to columns will result is the number of rows decreasing, but the number of columns remaining the same.

If is is desired to only change one dimension, the value 0 may be put in the other. For example, to change the array from 20 rows to 30, but not affect the number of columns, the command would be

```
SS 30 0
```

NOTE: There is a substantial amount of logic involved when shrinking the array, particularly when columns are involved. If an array with many rows has the number of columns lessened, several moments (read a minute or so) could elapse before the system completes the command. Therefore, do not fear system failure. SUGGESTION: When changing the size of the array, the rows are changed first. Therefore, say you're changing the 20 x 20 array to 10 x 100. The system will first create 100 rows, then shrink to 10 columns.

This will take quite a while, since the system is removing 10 columns of 100 rows each! To make the operation faster, FIRST change the number of columns (SS 0 10) THEN change the number of rows (SS 100 0). Using this method, only about 3 seconds will be used, a substantial increase in speed. If you are decreasing the number of rows and increasing the number of columns, FIRST decrease the rows, THEN increase the columns. The object is to make the number of rows as small as possible when creating or removing columns.

The Set Calculation Command

The set calc command is used to change the order and frequency of when the formulas are evaluated.

Formulas may be evaluated in either "row-major" or "column-major" order. If they are evaluated in row-major order, that means that they will be evaluated across each row, then down to the next row and across it, etc. In other words, the same way you read. In column-major order, they are evaluated down each column, then moved over to the next column and down it, etc.

The frequency may be set to either "manual" or "automatic". In manual mode, the formulas are only evaluated when the <Tab> key is pressed. In automatic mode, they are evaluated whenever a value or formula is entered.

The commands accepts the letters R, C, A, and M to represent each of the above options. Obviously, only two may be entered at once, but they can be entered in any order. For example, to set the calc order to column-major, and the frequency to automatic, the command would be

```
    SC C A          or          SC A C
```

To change only, say, the calculation order, this time to manual, the command would be

```
    SC M
```

While it will not produce an error to enter both R and C or both A and M at the same time, only the last one on the line will be used.

The Set Printer Command

The set printer command is used to send special characters to the printer. This should seldom be used, but is provided for special cases. The most common use would be so set a printer for some special mode.

The format of the command is

```
SP byte byte byte etc.
```

where (byte) is a decimal number. As many or few bytes may be sent as needed. For the exact bytes required for particular printers, you will have to refer to the appropriate printer's manual.

For an example on what this might be used for, consider the Vector Matrix printer. If you typed the command

```
SP 27 80
```

which corresponds to "ESC P" (the printer manual explains this), the printer would switch between 132 and 80 characters per line.

The Set Disk Command

The set disk command is used to reset the disk system in case you wish to change floppies. CP/M will normally not allow you to do this; it you do, you will get a BDOS read-only error, which can't be recovered from. Therefore, if you wish to change a floppy disk, put the new diskette into the drive and type

```
SD
```

which will tell CP/M about the new disk and allow you to access it.

T - the TITLE command

The title command is used to set and delete the main titles for the array, and to change the row and column titles. The commands for the row and column titles are substantially different from the commands for the main titles, therefore they will be explained separately.

Main Titles

The first two forms of the title command are for handling the main titles. They are as follows.

```
TMx text
TDx
```

where (x) denotes the number of the referenced main title. Since there are up to four main titles, (x) must be in the range of 1 to 4. If you are referencing the first main title, the 1 may be skipped. That is,

```
TM text     is the same as TM1, and

TD          is the same as TD1.
```

The (text) specified in the TM command refers to the text of the title, that is, what you wish the title to be. For example, the command

```
TM Budget Forecast
```

will set the first main title to "Budget Forecast".

The TD command is used to delete a main title, which is to say, make it blank.

The first main title is displayed on the screen, centered in a field of dashes. The remainder of the main titles can be viewed by using the P command, where they will be displayed in block 0 (see the P command). Note that when a main title is entered (or changed), the justification character is set to L. When a title is deleted, the justification character will be changed to X. It is possible to make a "comment" by typing a main title, then changing its justification character to X, thus keeping it from being printed.

Row and Column Titles

The second pair of title commands are used to change the row and column titles. The commands are

```
TR old-title new-title
TC old-title new-title
```

The TR is used to title a row, and TC is used to title a column. The (old-title) denotes the current title of the row or column you wish to change, and (new-title) denotes the new title you wish to assign to the row or column. For example, to change the title of row 1 to INCOME, the command would be

```
TR 1 INCOME
```

If you then changed you mind, and wanted to make the title SALES, the command would be

```
TR INCOME SALES
```

Note that the second time, the previously-assigned new title, INCOME, was used to refer to the row.

It is acceptable to use the relative method of referencing when specifying the old title. For example, to change the title of the current column to "JAN", you could use the command

```
TC . JAN
```

Titles for rows and columns may contain any characters, but must be only one word with a maximum of 8 characters, and must not start with a period (.). This is to prevent confusion with relative references.

U - the UPDATE Command

The update command is used to update the disk with the current contents of the array. Assuming there is a "current" file, all that need be typed to update the disk is

U

and the update will occur. If there is no current file, the system will say

No "current" file to update; Please type NEW filename:

and await your response. Type the name of a new file and <Return>. The system will create the file and save the array into it. In addition, it will be made the current file. If the file already exists, an error will occur. When asked for the file name, just typing <Return> will cancel the command.

Under certain circumstances, it may be desirable to save the array to disk, but NOT under the current file. This can be done by typing

UN

which stands for "Update New". The system will prompt with

Please type NEW filename:

and the proceed as above for a new file.

V - the VERIFY command

The verify command is used to verify the size of the array, or determine what the current actual cursor position is.

TAKE NOTE: Say for example that you have rows titled

HDGS, MONTHS, -, 1, 2, 3, 4, and 5.

Now you wish to remove ONE row. Your inclination will be to type the command "SS 4 0". This is not correct, however! Although the last row's title is 5, it is not the fifth row! If you typed that command, you would lose rows 2, 3, 4, and 5, which was certainly not your intention!

To eliminate the problem, two commands are provided. They are

```
VS
VC
```

The first, VS, tells the size of the array. The second, VC, tells the current cursor position on the array, in terms of absolute position. The VS command causes the message

```
Current size is x rows by y columns.
```

to be printed, where (x) is the number of rows and (y) is the number of columns. The VC command causes the message

```
Cursor is at row x and column y.
```

to be printed, where (x) and (y) are the row and column, respectively, where the cursor is at. In the example above, if the cursor was in the first column and on the row titled "5", then the VC command would result in the message

```
Cursor is at row 8 and column 1.
```

Typically, you might move the cursor to the first row (or column) that you wish to remove, then use the VC command, which will tell you the actual number of that row or column. If, on the other hand, you know that you wish to decrease the size of the array by a certain number of rows or columns, the VS command would be most useful.

By utilization of these commands, the possibility of accidentally destroying data with the SS command should be reduced.

W - the WIDTH command

The width command is used to set the widths of columns in the array. There are two forms provided.

```
W column width width width etc.
WA width
```

The first form allows you to specify (column), which is the column to start with, and a many (width)s as desired. Each (width) typed will be assigned to the following column. For example, assume that columns are titled JAN, FEB, MAR, and so forth. The command

```
W APR 20
```

will set the width of column APR to 20 characters. The command

```
W JAN 15 15 25 25 7
```

will set the width of columns JAN and FEB to 15, columns MAR and APR to 25, and column MAY to 7.

The second form will assign all columns the (width) specified.

Columns widths may be in the range of 2 to 64 characters, although columns narrower than about 6 characters begin to get useless.

X - the EXCHANGE command

The exchange command is used to exchange, or swap, two rows or columns. The commands have the form

```
XR row-1 row-2
XC column-1 column-2
```

where (row-1) and (row-2) or (column-1) and (column-2) are the rows or columns to be exchanged. All aspects of the rows or columns are exchanged - the data, the formulas, the titles, and if columns, the widths.

NOTE: As with the K, M, and O commands, relative references in formulas affected by the execution of the command will not be changed. References over an exchanged row or column will not be bothered. References TO the row or column will simply get the new data instead of the old. However, references CONTAINED in the row or column will now be evaluated relative to the new position, whereas they were entered relative to the old position. This could potentially result in incorrect calcalations, so beware.

Section 3

Errors

All errors, what they mean, what causes them, and how they can be avoided.

ERRORS

From time to time, something will be done wrong. ExecuPlan has a vast number of error messages to help you figure out what was haywire.

When an error occurs, the error message is displayed in the far bottom-right corner of the screen. Normally, a character or word somewhere on the screen is also highlighted. That character or word is the source of the error. Not all errors, however, have this feature.

After the error is displayed, the system just stops and waits for the user to type something. As soon as a character is typed, the system proceeds. Most errors result in the system returning to command mode and awaiting another command. Some, however, have other results. Certain math errors, for example, simply warn you; when you type a character, processing continues.

If an error occurs while formulas are being evaluated, then an additional message is displayed on the command line telling you where the formula is that caused the problem. Also, the formula itself will be displayed on the formula line of the display.

Following are explanations of all of the errors and their causes.

- COMMAND ERROR -

The command error indicates that the command typed is invalid. If the first character is highlighted, then that is the invalid command. If the second character is highlighted, then the first character is valid, but the second one is not.

- SYNTAX ERROR -

Syntax error indicates one of several things. If the first character following the command is not a space, then a syntax error will result. If an invalid character is encountered while a decimal number is being read, that will also cause a syntax error. There are also a couple of other obscure conditions that will cause a syntax error. The character highlighted will normally be the character that was undigestable.

- MISSING ARGUMENT -

This indicates that something was expected, but nothing was found. For example, typing a title command but leaving out one of the titles will cause a missing argument error. A character is not always highlighted, but if one is, it is at that point that another argument was expected.

- TITLE TOO LONG -

This indicates that a title was being read, but more than eight characters were found in the title. The ninth character will be highlighted.

- TITLE NOT FOUND -

This should be pretty obvious. A title was read, but there is no row or column with that title. The entire title that was not found will be highlighted. Also, typing a relative reference that refers to someplace off the array will cause this error.

- INVALID RELATIVE -

When a title is being read, if the first character is a period (.), then it assumes that a relative reference is in the works. If the character(s) following the period do not make sense, then this error will result.

- DUPLICATE TITLE -

Duplicate title indicates that a title was entered that should not already exist, but it does. An example would be the second argument in a TR command. This error is also used when a disk file name is typed for a new file, and the file already exists. The entire title (or file name) will be highlighted.

- OUT OF RANGE -

Certain commands expect a number within a certain acceptable range. If the number typed in not within that range, the out of range error will be the result. For example, typing a column width less than 2 or greater than 64 would cause this error. The number will be highlighted.

- BAD FORMAT CHAR -

A character is encountered in a F command that is not acceptable. The character that was unacceptable will be highlighted.

- DECIMALS > 15 -

This is similar to the above in that it means that something is wrong in a F command. This error, however, indicates that a number for the decimal count is too large. Fifteen is the maximum number of decimal places that may be specified. The offending number will be highlighted.

- BAD CALC ORDER -

During the reading of a SC command, a character other than R, C, M, or A was reached. The character is highlighted.

- INPUT ERROR -

An input error occurs when an EV command is being executed and something wrong is reached. Normally, this is a decimal number containing some garbage characters. The number being read is highlighted.

- EXCESS INPUT -

This is reached during one of the multiple forms of the EV or EF commands. For example, if the command "EVCR 45 100" is given, but there are only 50 rows, an excess input error will be generated. Whatever portion of the command that caused too much input to occur will be highlighted. Under the EV command, the extra will be ignored. If the command is EFM, the whole command will be cancelled.

- INVALID DRIVE -

When a D command is executed and a drive is specified, that drive must be in the range A to P. Anything outside of that range will cause an invalid drive error. The offending character will be highlighted. Note that even something within the range A to P may be invalid, since few systems have 16 disks! However, the system reqlly doesn't know that, hence the extended range. If you type a drive within the range that doesn't exist, then CP/M will get into the act and give a BDOS error. These are unrecoverable!

- WRITE PROTECTED -

Under CP/M version 2, files may be set to "read-only" status. Trying to write to or erase such a file will result in this error. Nothing is highlighted.

- DISK I/O ERROR -

This error means one of several things. One possibility is that there is a physical error on the disk. Another is that an attempt is being made to write to the disk, but there is no more room. Finally, an attempt may be being made to read a file which is goofed up somehow. Nothing is highlighted by this error.

- FORMULA ERROR -

This is actually a rather general error. What it means is that there is something wrong in a formula, such as a non-existant function, an improperly-typed number, an invalid operator, or something else along that line. The system will try to highlight the character that caused the problem, but depending on the cause, that character might not actually be the source of the error.

- BAD RANGE BOUNDS -

When evaluating a multi-argument function, the arguments were invalid. There are actually two separate things that could be wrong. First, the arguments aren't references at all; second, they could be references, but define an invalid range for the function. The arguments for such a function must be, respectively, the top-left and bottom-right corners of a rectangle. The rectangle may in fact be a line, or even a point. However, the second reference can't have a row or column that is less than the row or column in the first reference. Nothing will be highlighted.

- MATH IMPOSSIBLE -

Certain things just can't be done with real numbers, and things like logarithms or square roots of negative numbers are such things. Nothing will be highlighted.

- DIVISION BY ZERO -

The cause of this is quite apparent. The particular thing about this error is that the character typed to recover from the error condition determines what will be used as the result of the operation (that caused the error). If the character "0" is typed, then zero will be used as the result. If the character <Esc> is pressed, then the system will abort the operation and return to command mode. Any other character will cause the value 9.99999999999999 times 10 to the 35th power to be used as the result of the division.

- OVERFLOW -

Some math operation resulted in a number that is just too big. The largest possible number or something near it will be used instead. Nothing will be highlighted.

- OUT OF MEMORY -

This indicates that there is too little memory available to carry out the operation requested. Normally, there is around 30K of free space to start with. With gobs of text and formulas in memory, this can disappear quickly. When an operation would result with less than about 100 bytes (characters) of free space, this error is caused. The padding is allowed because certain operations use some memory during their execution. Note that if this error is given during an EFM command with large arguments, there might really be enough room. The system allows for maximum tolerences when calculating the space available. If you think there is enough space, try reentering the formula, but in smaller multiples. Nothing is highlighted by this error.

- HELP UNAVAILABLE -

This indicates that the help command was used, but the help file was not found. The help file is called EPL.SYS, and must reside on the logged-in disk under CP/M. Either the disk containing EPL.SYS was not in the drive, or the user is assuming the wrong drive is the logged-in one. Nothing is highlighted.

Section 4

Math Capabilities

How the math package is used, how formulas are formed, and explanations of all operators, functions, and special capabilities.

MATH CAPABILITIES

ExecuPlan has a very powerful math package incorporated into it. Virtually all operators and functions necessary for any type of calculations are provided. Furthermore, their usage is in a simple, algebraic format.

The EF command is used to enter formulas into the system. For example, the command

```
EF AVG([1,1],[5,1])
```

would enter a formula which would compute the average of locations [1,1] through [5,1].

Characteristics of Formulas

This section will be devoted to explaining exactly how formulas are formed. If you, the reader, are familiar with the programming language BASIC, then suffice it to say that ExecuPlan handles formulas the same way. Assuming that you're not, then read on.

Formulas are essentially a list of items, where each item is either data of some type, or an operator. Things like numbers or functions are data; plus, minus, and so forth are operators.

Data and operators are simply strung together to form an algebraic expression. For example,

```
3+4+9
```

is a valid expression, containing numeric data and the operator "+". Under ExecuPlan, most formulas will "reference" locations in the array. A sample formula with a reference would be

```
5*[1,1]
```

which means "take the number 5 and multiply it by the contents of location [1,1]".

One of the more powerful features is that of functions. These can be confusing, because while they perform an operation like an operator, they are treated as data, because when evaluated, a function is a value. A simple function might be

```
SQR(15)
```

which means "take the square root of 15". However, when contained in an expression, such as

    [3,5]-SQR(15)

note that it is treated like data. Note also the syntax of a function: the function itself, followed by a left parenthesis, then the data the function is to be performed upon (the argument), then a right parenthesis. Within the parentheses can be another expression, such as

    SQR(45-[3,5]*12)

which will take the square root of the result of the expression which is its argument.

Parentheses may also be used as part of a formula, besides being used to enclose the argument for a function. They are used just as in algebra, to represent a partial result. For example

    [1,5]/(3-4*[2,5])

which means to take the contents of [1,5] and divide it by the result of the parenthesized expression. Unlike most systems, ExecuPlan will not complain if there are not a matching number of left and right parentheses. Instead, it will just ignore the extras. Also, parentheses may be nested to any level, that is, you may have as many as you need to properly represent your expression.

There are really only two rules with regard to formulas. First, they CANNOT contain ANY SPACES (blank characters). The first blank encountered is considered the end of the formula, and the extra past it will either be ignored or cause an error. Second, a formula can only be as long as you can type, which limits it to about 74 characters.

Precedence of Operators

Given an expression, the question arises as to in what order to evaluate the operators and functions. There are two normal ways to do this. One is called "left-to-right", and means that the operators are evaluated in the order they are encountered. The order is called "precedence", which means that they are evaluated in a specific order with certain operators first, regardless of the order they're in.

It has been said that business people use the l-to-r method, and scientific people use the precedence method, and that since computer programmers are scientific types, that's why computers always use the precedence method. Well, that may be true, but the programmer of this system is just as much a business type as scientific type. The reason that precedence was used is simply that it is more powerful; that is, certain operations cannot be done as easily with the l-to-r method. Besides, by now most business types are so used to precedence that it would cause even more confusion to have a program use l-to-r!

Before getting to precedence, though, it might be a good idea to mention what the operators are! There are five of them:

+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation (that is, raise to power. 2^3 is 8.)

So much for the operators. Now, basically their precedence is as follows:

1. parenthesized expressions
2. functions (remember that functions are evaluated, then treated as a value from that point on)
3. exponentiation
4. multiplication and division
5. addition and subtraction

When operators of equal precedence are met, then those operators are evaluated left-to-right.

Examples:

3+4*5	4 * 5 first, then add 3
3*(3+5)	3 + 5 first, because it's parenthesized
12/int(3*11)	3 * 11 first, then function "int", then division.
4+4-2*7	take 4, add 4, then subtract the product 2*7 (hence the result is -6)

NOTE: Often it will be desired to use a negative number, for example -3, in an expression. Therefore, it should be explained how it will be handled by the formula evaluator.

Essentially, whenever two consecutive operators are encountered, the program inserts a 0 between them. Thus, the sequence 2++4 would result in 2+0+4, which would give the presumed correct answer. In some cases, however, an incorrect answer might be arrived at. For example, the sequence 4*-3, which should evaluate to -12, will evaluate to 4*0-3, which is -3.

Normally, two consecutive operators should never be used. The example above, however, is a valid possibility. It is an example of the unary "negative" operation, which is the only usual possibility.

To eliminate the problem, use parentheses around such an operation when it in used in an expression. For example, -SQR(2) would not need parentheses, but 4*-3 would, so you'd enter 4*(-3).

Defined Constants

There is one other nice little feature of the program, defined constants. These are simply a couple of numbers that may or may not be used very much, but will save some typing when they are. The defined constants are

#PI	The value PI, 3.1415926 etc.
#E	The value e, 2.718281 etc.
#RND	A random number in the range 1 to 2.

Defined constants can be used wherever a number would normally be used.

Functions

At this point, we'll take a look at the functions provided in ExecuPlan. There are basically two types of functions, single-argument and multi-argument. A single-arg function is something like square root; a multi-arg function would be something like standard deviation.

These two types of functions are slightly different, beyond the obvious fact that they take a different number of arguments. A single-arg function can take anything as an argument - a number, defined constant, another function, a reference, even a whole expression.

A multi-arg function requires that the arguments be references. Specifically, these references represent the top-left and bottom-right corners of a

rectangular portion of the array. The rectangle may actually be a line, or even a single location, but in a manner of speaking, these are still rectangles.

For example, the arguments ([1,1],[5,5]) define a 5 x 5 rectangle. The arguments ([1,1],[5,1]) define a vertical line; the arguments ([1,1],[1,5]) define a horizontal line. The arguments ([1,1],[1,1]) define a point. Nevertheless, they would all be acceptable. The arguments ([3,3],[4,2]) would, however, not be allowed, since the second reference is to the LEFT of the first. It would also not be allowed if it were ABOVE the first.

The reason for this restriction should be apparent. The functions which take multiple arguments operate on a range, that is, a group of values. Only by specifying the bounds of the range, as references, can the function possibly know what numbers to use.

Naturally, there is always an odd case. Here, it is the net present value function, which requires both a range and numeric arguments. The exact format of this function will be explained when the function is explained.

Following are explanations of all of the functions, how they're used, what they do, and which type they are.

ABS Absolute Value

Type: single-arg

The absolute value function returns the absolute value if its argument. In other words, if the argument is positive, it is returned unchanged. If it is negative, it will be made positive. Example:

ABS(3) = 3 ABS(-4) = 4.

INT Integer

Type: single-arg

The integer function returns the greatest integer less than or equal to the argument. Example:

INT(3) = 3 INT(#PI) = 3 INT(5.9) = 5 INT(-1.1) = -2.

SIN	***Sine***
COS	***Cosine***
TAN	***Tangent***
ATN	***Arctangent***

Type: single-arg

These functions return the result of the appropriate trigonometric function. The argument is expected to be in radians (with the exception of ATN, which returns its result in radians).

LN	***Natural Logarithm***
LOG	***Decimal Logarithm***

Type: single-arg

These functions return the appropriate log of the argument. LN is the natural, or naperian, log (base e), while LOG is the decimal (base 10) log. Example:

LN(#E) = 1   LOG(100) = 2.

EXP	***Exponent***

Type: single-arg

This function also commonly called antilog. It returns the natural (base e) antilogarithm of the argument. Example:

EXP(1) = e

SQR	***Square Root***

Type: single-arg

The square root function returns.... the square root of its argument. Bet you would have never guessed.

SUM	**Summation**

Type: multi-arg

The sum function returns the total of all of the numbers in the range specified. For example, if [1,1] through [5,1] contain the values 1, 2, 3, 4, and 5, then

SUM([1,1],[5,1]) *returns the value 15.*

MIN	**Minimum**
MAX	**Maximum**

Type: multi-arg

The MIN and MAX functions return the smallest or largest, respectively, number is the range. Assuming the conditions above (in the SUM explanation), MIN would return 1 and MAX would return 5.

AVG	**Average**
MEAN	**Mean**

Type: multi-arg

The AVG and MEAN functions are the same thing - the average of the numbers in the range specified. Both are provided so that whichever term is preferred by the user may be used.

VAR	**Variance**
SD	**Standard Deviation**

Type: multi-arg

The VAR and SD functions compute the variance and standard deviation, respectively, of the argument range.

COUNT Counter

Type: multi-arg

The COUNT function will simply return the number of items in the argument range. This function is used by the average, variance, standard deviation, and net present value functions. IMPORTANT NOTE: This function, and therefore all of the functions that use it, react in a certain way to invalid contents of locations in the array. That is, when a certain location within the argument range does not contain a number (instead, it contains nothing or text), the value zero will be used instead. The location will still be counted! Therefore, any of the above-mentioned functions could return an invalid result if any locations in the argument range are invalid.

NPV Net Present Value

Type: special multi-arg

This function returns the computed NPV of the argument range, using additional numbers specified in the arguments. The standard formula for net present value is

$$\sum_{t=1}^{N} \frac{F_t}{(1+k)^t} - I$$

where F(1), F(2), through F(n) are cash returns for years 1 through n, k is the interest rate, and I is the initial cost. The format for the NPV function is

NPV([bnd-1],[bnd-2],k,I)

where k and I correspond to the same variables in the formula. The value for n is computed as the COUNT of the locations in the range bounded by [bnd-1] and [bnd-2].

Section 5

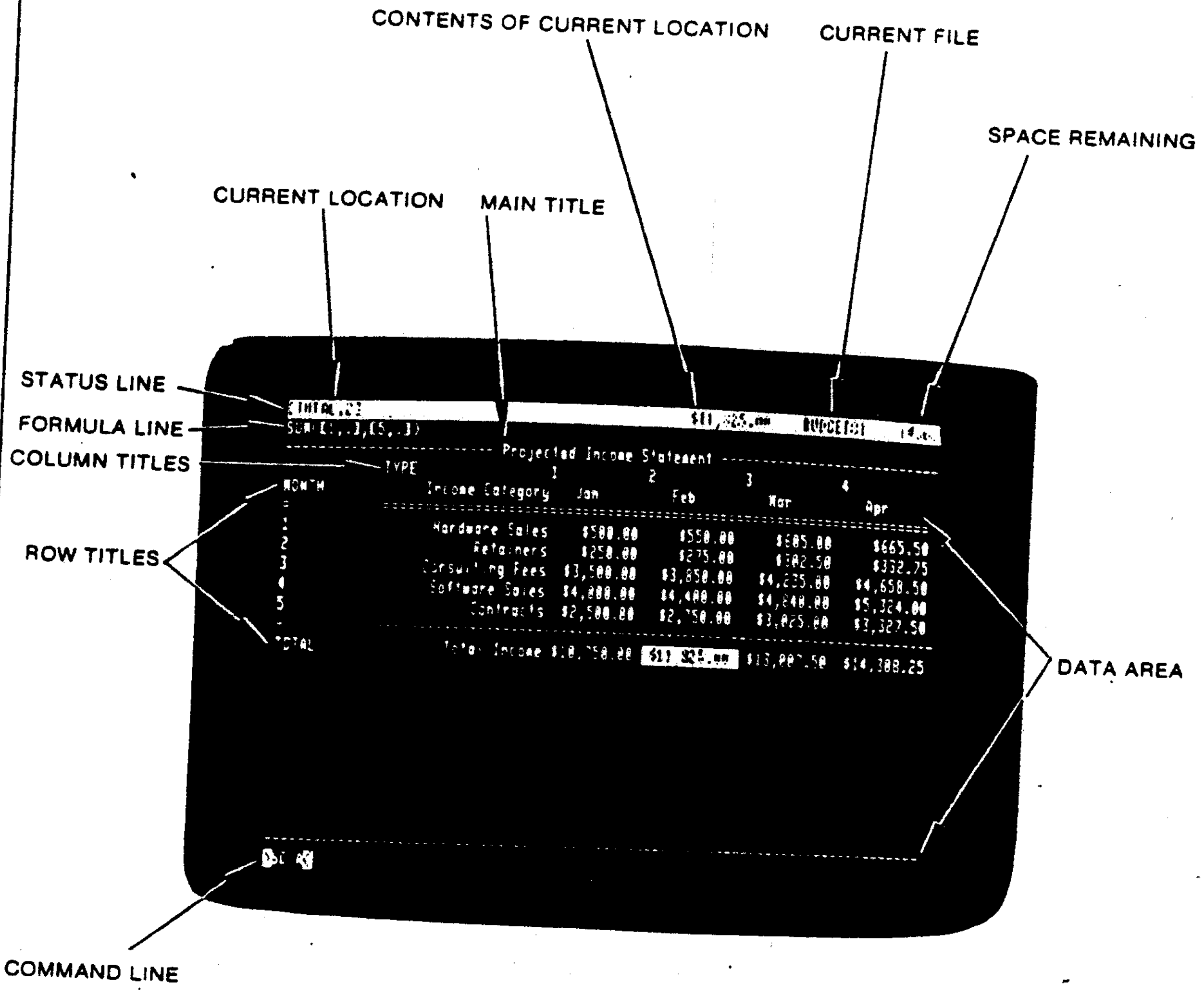
Miscellaneous

A miscellaneous collection of information that may help the user from time to time.

Appendix

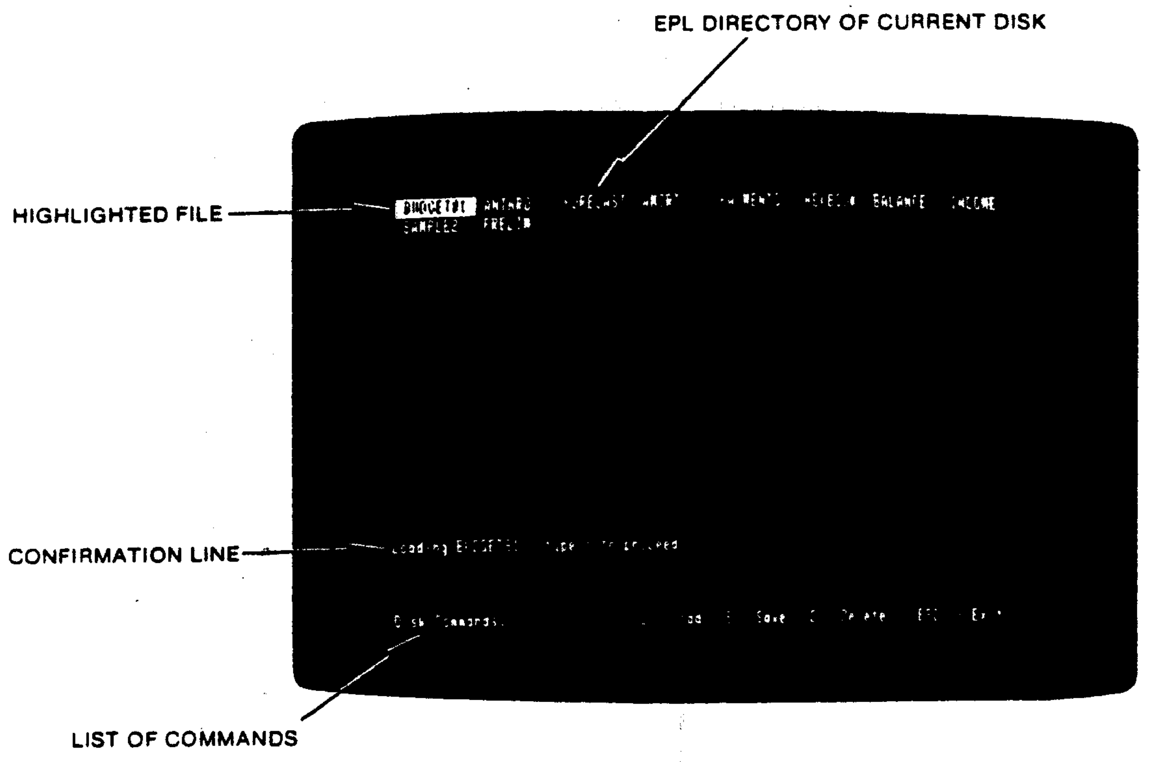
Sample Screens

Some sample screens to assist in understanding how various information is displayed and where it is displayed at.



```
[TOTAL,2]                                        $11,325.00   BUDGET01    [illegible]
SUM([1,.],[5,.])
-------------------------- Projected Income Statement --------------------------------
          TYPE
MONTH                              1            2            3            4
=             Income Category    Jan          Feb          Mar          Apr
              ====================================================================
1              Hardware Sales    $500.00      $550.00      $605.00      $665.50
2                  Retainers     $250.00      $275.00      $302.50      $332.75
3            Consulting Fees   $3,500.00    $3,850.00    $4,235.00    $4,658.50
4             Software Sales   $4,000.00    $4,400.00    $4,840.00    $5,324.00
5                  Contracts   $2,500.00    $2,750.00    $3,025.00    $3,327.50
-             --------------------------------------------------------------------
TOTAL           Total Income  $10,750.00   $11,325.00   $13,007.50   $14,308.25
```

Figure 1: Sample Display

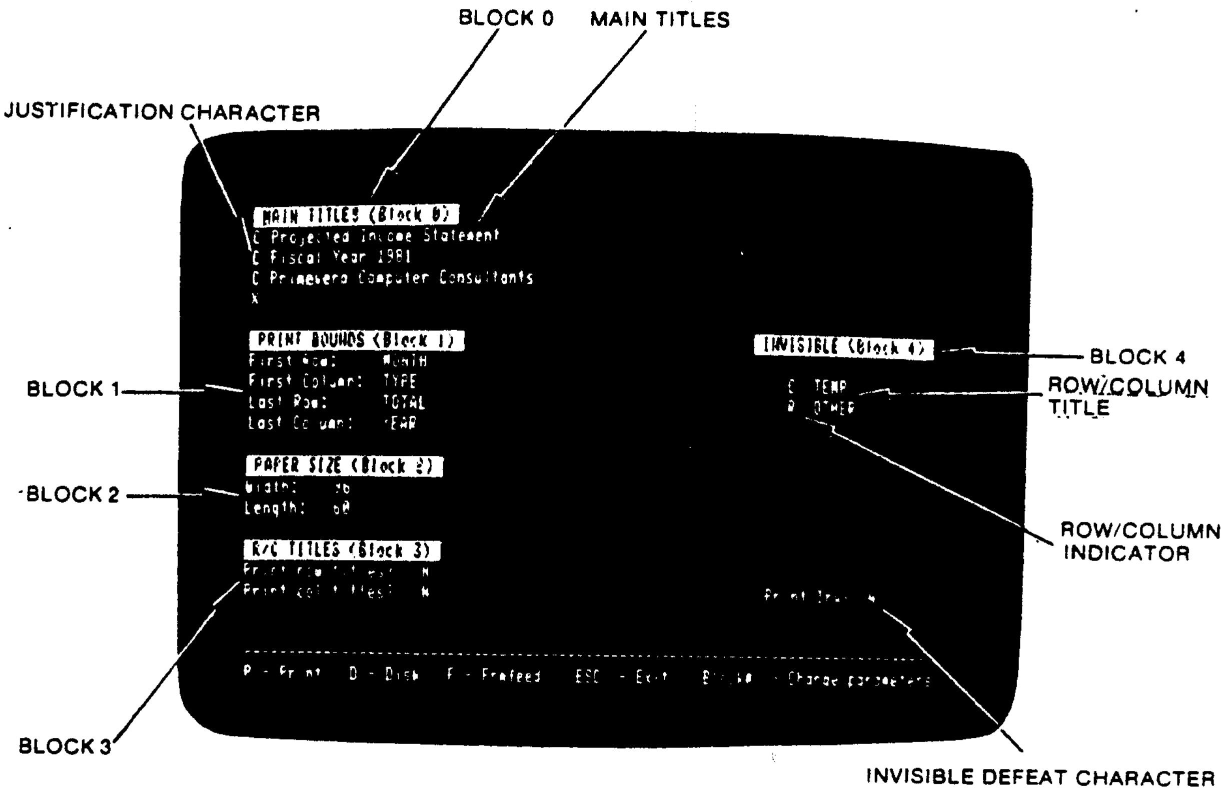


```
BUDGET01  ANTHRO  [illegible]  [illegible]  [illegible]  [illegible]  BALANCE  INCOME
SAMPLE2   [illegible]

Loading [illegible]  [illegible] to proceed

Disk Commands:            [illegible]  Save  [illegible]  Delete  ESC  Exit
```

Figure 2: Sample Disk Screen



```
MAIN TITLES (Block 0)
C Projected Income Statement
C Fiscal Year 1981
C Primevera Computer Consultants
X

PRINT BOUNDS (Block 1)                              INVISIBLE (Block 4)
First Row:      MONTH
First Column:   TYPE                                  C  TEMP
Last Row:       TOTAL                                 R  OTHER
Last Column:    YEAR

PAPER SIZE (Block 2)
Width:    96
Length:   60

R/C TITLES (Block 3)
Print row titles?   N
Print col titles?   N                              Print Inv?  N

---------------------------------------------------------------------------
P - Print   D - Disk   F - Formfeed   ESC - Exit   Block# - Change parameters
```

Figure 3: Sample Print Screen

INDEX

Memory Utilization

This information is provided so that the user may have some idea of how memory is used under ExecuPlan. This information is somewhat advanced, and if you don't understand it, don't worry, it doesn't matter.

In order, ExecuPlan keeps the following tables: Column widths, Row titles, Column titles, Primary addresses, Numbers, Formulas, and Strings. All of the tables start from the end of ExecuPlan and build up, except for the strings, which start at the end of memory (actually the base of the BDOS in CP/M) and build down.

The column width table takes one byte per column, that byte being the width of the column.

The row and column title tables take eight bytes per title.

The primary address table, which is used as a giant reference table for the array, takes three bytes per location on the array. The first two bytes are a relative pointer to the actual data in memory, the last byte is the format byte for that location.

The numeric table holds all of the numbers. This table is dynamic, that is, only as many numbers as are actually in the system are kept. Numbers take eight bytes each, and are stored in Microsoft double precision floating point format, which yields 16 digits of precision.

The formula table holds all of the formulas. Unlike the numeric table which is pointed to, the formula table is fully independent. Only the beginning is pointed to. Each formula has a length byte, a destination row and column (taking two bytes), the text of the formula, and then a termination byte. Therefore, formulas take up the number of bytes in the text, plus four.

Text, known to a computer as strings, is stored in the string table. The string table is simply sequentially allocated, down from the top of memory. The strings are stored in reverse order since the table builds down. There are no overhead bytes with strings (the end is indicated by bit 7 on, and the beginning is pointed to from the primary address table), strings take only as many bytes as the string is long.

System Recovery

From time to time, some type of error might occur that will result in the user being dropped out of ExecuPlan into CP/M or the Monitor.

The most common possibility is that you might accidentally try to access a disk drive that does not exist. CP/M will respond with a message like

```
BDOS Error on D: Bad Sector
```

or something like that. This type of error is called "fatal", because there is no direct way to recover from it. Another possibility is that you might accidentally hit the reset button.

In any case, the probability is good that there was something you were working on that you don't want to lose. Therefore, it is nice to be able to recover from these conditions.

Monitor

From the Monitor, type "G 0100" and see what happens. Chances are, you should be right back in ExecuPlan. You might have to type "JB" to clean up the screen.

It is suggested that you dismount the disk just in case. There could be a possibility that the memory image is goofed up, and that might cause crazy things to happen. Better safe than sorry.

CP/M

More common is the case where you get dropped back into CP/M because of a disk error or read-only error. When you get the message mentioned above, or one like it, hit <Return>. You will then be back in CP/M. Now, type the following command:

```
A>SAVE 0 HOPE.COM
```

What this does is to create an empty file on the disk, without disturbing the memory image. If you already have a file called HOPE.COM on the disk, use another name. Now type "HOPE". With any luck, you will be back in ExecuPlan and can continue. If this does not work, reset the computer and proceed as explained in the above section. If that doesn't work, there is probably no recovery possible.

Foreword

Welcome to the world of ExecuPlan!

This reference guide is intended to provide all of the reference information needed to help you to use ExecuPlan to its fullest capability. Sections are provided covering how to bring up the system, read the screen, type and edit commands, enter data, and so forth. Complete explanations of all commands are provided. A section covers all of the errors that might occur and what they mean. A complete section is devoted to the math capabilities, entering formulas, how formulas are evaluated, functions, and so forth. Finally, some helpful information is provided for special circumstances, and the appendix has some annotated sample screens.

This guide is actually not intended for the first-time user. The accompanying guide, ExecuPlan Primer, is the most basic manual. It covers beginning concepts and moves step-by-step to more complex models.

It is recommended that first the primer be read and understood, then this reference guide read and understood. Once the concepts are understood, the reference guide will then be useful for looking up information as needed. This guide does cover most topics in more detail than the primer, but they are explained in a more direct manner, assuming that the reader has some familiarity with the system.

It cannot be stressed enough that you should READ this guide. There will be countless occasions where you aren't sure about something, or don't understand why what happens happens, or something else along that line. If this is the case, get out this reference guide, or the primer if necessary, and look it up. Almost always the information that you need will be found in moments.

To briefly preface the sections of this guide: Section 1 contains general information, such as bringing the system up, understanding the array, reading the screen, understanding titles and references, and typing and editing commands. Section 2 gives detailed explanations of all of the commands. Section 3 covers all errors. Section 4 explains the math package, how formulas are evaluated, how operators and functions work, and other math-related information. Section 5 just has some miscellaneous information.

Again, congratulations on your acquisition of ExecuPlan, and we hope that it will be beneficial to you in whatever applications you have in mind.

Acknowledgment

The ExecuPlan program and this reference guide were completely designed and written by Neale E. Brassell exclusively for Vector Graphic Inc.

The program was developed on a Vector Graphic VIP computer system with two drives, a Sprint 3 printer, and a Teletype model 40 line printer, utilizing the SCOPE editor, ZSM assembler, and RAID debugging system. The documentation was written with the MEMORITE III word processing system

The source program consists of nearly a quarter million bytes of source code, contained in fifteen program modules totaling about 170 pages. After assembly, the object program is about 20K long. Additionally, the help screens occupy another 54K data file.

The floating point math routines and the single-argument functions are derived from Microsoft BASIC version 5.1 which is licensed from Microsoft Inc. The balance of the math routines (parser, expression evaluator, and multi-argument functions) and the balance of the program are all original code.

The author would like to thank the software staff at Vector Graphic for their support and suggestions on the development of ExecuPlan. I am particularly grateful to Chris Cory, who contributed many significant ideas, was instrumental in the debugging of the program, and who wrote the primer and developed all of the sample applications that come with ExecuPlan.

STARTING UP

ExecuPlan is a CP/M-based program, supplied on a CP/M diskette. To execute the program, insert the diskette into the drive and type

```
A>EPL            (You type what's underlined)
```

After a few seconds the screen will display a large banner identifying ExecuPlan, and giving the revision and copyright notice. Type any character on the keyboard. The banner will be erased and an initialized array will be shown on the screen. You're up and running!

At the time EPL is loaded, you may also tell the system to automatically load or create a file on the disk. For example, to run ExecuPlan and load the file BUDGET81, you would type

```
A>EPL BUDGET81
```

After the banner is erased, the system would display the message

```
Loading BUDGET81
```

and proceed to load the program. After the program is loaded, the array containing the data will be displayed.

One other capability when EPL is executed is the ability to create a new file. The command is typed the same as to load a file. If the file does not exist, then the system will display the message

```
Creating BUDGET81
```

and proceed to create the file. The array displayed will be blank, but the file specified will be created, and when the array is updated it will be written into that file.

THE ARRAY

Concept

ExecuPlan is built around the concept of an electronic array. This array contains a number of rows and columns. Information may be put into each intersection of a row and a column. Then, information put into the array can be used to compute other information, which is also put into the array. Finally, the array may be printed in a report fashion.

Each intersection of a row and column is called a "coordinate" or "location". If, for example, the array contains 20 rows and 20 columns, then there would be 400 locations in the array.

References

Given all of these locations, a method must be used to refer to each of them. This is called a "reference", and has the following format:

[Row,Column]

This is the scheme used throughout ExecuPlan to reference locations. For example, a reference to row 10 and column 8 would be

[10,8]

Each row and column has a "title", which is used in a reference. In the preceding example, "10" is the row title, and "8" is the column title. Titles do not have to be numbers, however. A reference might be

[INCOME,JANUARY]

which would refer to row INCOME and column JANUARY. A command is provided for changing the titles of rows and columns.

In the next section, which talks about the screen, the idea of a "current" location will be brought up. For the purpose of references, suffice it for the moment to say that there is a location in the array that is considered to be the current one. From this comes the idea of "relative references". A relative reference is one where rather than specifying the title of a row or column, the distance away from the current location is specified. The character period (.) is used to mean the current location. For example, the reference

[.,.]

means "the current row and the current column", that is, the current location. The reference

    [.,.-1]

means "the current row, and the column that is 1 to the left of the current". If the current location is [4,4], then the aforementioned reference would point to [4,3].

Following are some examples of relative references.

[.,.-5]	Current row, current column - 5
[7,.]	Row "7", current column
[.-1,JAN]	Current row - 1, column JAN

THE SCREEN

This section explains how the screen looks and what all of the things on the screen mean. It is suggested that you look at the picture of a sample screen in the appendix while reading this section.

Concept

Conceptually, the screen is a movable "window" over the array. When the system is initialized, that window is over the top-left corner of the array. Using the arrow keys, the window can be moved over any section of the array. The portion of the array that appears under the window is always 18 rows (unless there are less than that many) by however many columns will fit. Initially, columns have a width of 16 characters, so 4 will fit on the screen. Columns may have almost any width, however, so the number of columns displayed will vary from 1 to 34. If a column's full width will not fit on the screen, it will not be displayed.

Standard Information

At this point, let's take a look at the screen layout.

FIRST LINE: The first line (which is highlighted) is the "status" line, because it tells you the status of the system. Four things are displayed on the status line. 1: The current location. This is the location that the cursor is currently on (see below for additional explanation of the cursor). 2: Contents of the current location. This is simply whatever is in the current location, displayed according to the format of that location. 3: The current file, if any. This area might be blank. If it is not, then it contains the name of the current file, which is used if the file is updated. 4: The amount of space still available in memory. Initially, this is about 30,000 characters.

SECOND LINE: The second line is the "formula" line. If the current location contains a formula, that formula will be displayed on this line. If there is no formula associated with the current location, this line will be blank.

THIRD LINE: This is the "main title" line. Initially, this line contains a line of dashes. When main title #1 is set (see T command), then that title is centered in the field of dashes and displayed on this line.

BOTTOM LINE: This is the "command" line, where commands are typed. This will be explained in detail in the next section.

Data Area

The area between the main title line and the row of dashes above the command line is the data area of the screen. It, in turn, actually has two sections, the titles and the data.

The titles are displayed across the top of the data area and down the left side. The COLUMN titles are across the top, and the ROW titles down the left side.

The data itself fills the remainder of the area. If there is no data in the system, this area will be blank.

It is somewhat difficult to explain, but very easy to understand, the layout of the data in the data area. Essentially, each intersection of a row and a column (each LOCATION) can contain an item of data. If you visually trace a horizontal line across from the row title, and visually trace a vertical line down from the column title, then where those lines intersect will be where the data contained in that location will be displayed. For an easy example, the data contained in [1,1] will be the top-left corner; data contained in [18,4] (on an initialized system) will be in the bottom-right corner.

The Cursor

The "cursor" is the name given to the large white rectangle that is somewhere in the data area. The position of that cursor, taken as a reference, is called the "current location", a term that has been and will be used OFTEN. If the cursor in in the top-left corner (again, of an initialized system), then it is in location [1,1], and therefore [1,1] is said to be the current location. When the arrow keys are used (or a couple of other keys, all explained later), the cursor moves. If it is in the top-left corner, then pressing the <down arrow> key will move the cursor to the second row. Pressing it again will move to the third row, and so forth.

Moving the Cursor

As hinted at above, the cursor is moved around on the screen with the <arrow> keys. Here, we'll go into more detail on this.

The arrow keys move the cursor in the appropriate direction. When the cursor reaches the edge of the screen, then instead of moving the cursor, the whole screen is moved. More specifically, the window that the screen represents moves with respect to the entire array. If column 4 is the rightmost column on the screen and the <right arrow> key is pressed, the first column on the screen will be shifted off and the next column to the right will be shifted on. Then, if the <left arrow> key is pressed, only the cursor will move. Repeated pressings will move the cursor to the left until it

is on column 2, which at this point would be the leftmost. The next pressing would shift the screen left and move column 1 onto the screen.

When the cursor is at the extreme point in any direction and that arrow key is pressed, the system will simply ignore it.

In addition to the regular arrow keys, the <up arrow> and <down arrow> keys may be shifted to move the screen 18 rows (a screenful) at a time. For example, if the cursor is on row 1 and <shift down arrow> is pressed, the cursor will be on row 18, which would be the beginning of the second screenful of rows. The exception to this is if that action would result in less than 18 rows being displayed. In that case, the screen will be shifted as far as possible, but not so far as to result in less than 18 rows being displayed.

Finally, there are three other cursor-movement keys. Pressing <Lf> will move the cursor to the leftmost column of the current row, as if the <left arrow> key were pressed repeatedly. Typing <Control-T> will move the cursor to the top of the array and put it at the top line of the data display also. <Control-E> will move the cursor to the end of the array, and put the cursor on the bottom line also.

TYPING COMMANDS

In order to really DO anything with ExecuPlan, you must give it "commands", which are instructions telling it what to do.

There are two directions taken when it comes to giving a computer commands. One is the "language" method. In this method, you create a list of instructions, feed it to the computer, and get back the results. This method is interesting, but it really just amounts to simplified computer programming. This is not an acceptable method of getting results to someone who is not a programmer.

The other method is known as "interaction". In this method, you give the computer AN INSTRUCTION. The computer follows that instruction and displays the result, if any. You then give the computer another instruction, and it carrys that out. In other words, the computer interacts with the user. It is this method that is used by ExecuPlan.

Recall from the last section that the bottom line is the "command" line. It is on this line that commands are typed.

In the very bottom left corner of the screen are two things. These are the "command prompt" and the "command cursor". Before you begin to type a command, these look like ><, only highlighted. The left character, >, is the prompt. The right character, <, is the command cursor, NOT TO BE CONFUSED WITH THE CURSOR IN THE ARRAY. The command cursor tells you where on the line you are at. When you type a character, the character appears on the screen where the command cursor WAS, then the cursor reappears one character to the right.

Notice in the last sentence of that paragraph that the command cursor was refered to simply as the cursor. This might seem confusing, since the white rectangle on the screen in the current location is called the cursor. Well, they are SO conceptually different that you will have no difficulty ascertaining which one is meant when you see the word "cursor".

Characteristics of Command Mode

Yes, when the system is sitting waiting for you to type a command, that is called "command mode". When the system is in command mode, you may type commands (seems reasonable, doesn't it?).

As mentioned above, when a character is typed, it appears on the screen where the cursor was, then the cursor reappears one character to the right. This should not be new, since virtually all software operates in this manner.

At any time during the typing of a command, all of the capabilities of moving the screen (arrow keys, etc.) are available.

When a command has been typed satisfactorily and you wish the computer to carry out that command, the <Return> key must be pressed. As soon as that key is pressed, the system STOPS waiting for you to type, and begins to execute the command.

In addition to normal letters and numbers used when typing commands, certain other "special" characters can be used, which will be explained in the following paragraphs.

Finally, and for lack of a better place to explain it, it should be noted that upper case and lower case letters are fully interchangeable. That is, typing "EVC" is the same as "evc" or "Evc" or "evC" or whatever. Actually, this is true not just in command mode, but everywhere in the system. EXCEPT: Titles do not allow interchanging of cases. That is, "Jan" and "JAN" are different titles.

Special Characters

While typing commands, there are several special characters available. They are of two types. The first is editing, which means that they allow you to "edit", or change, what you've typed. The second is "inserting", meaning that you can "insert" certain things from the system without having to type them.

Each special character will be explained here.

<Bs> - Typing the <Bs> key (Backspace) will cause the cursor to "back up", effectively erasing the last character that was typed. For example, if you typed "ET HELLI" then realized that you meant to type "O", not "I", then you could hit the <Bs> key which would back up and erase the "I". Then, you'd type "O" and continue.

<Del> - Typing <Del> simply erases the entire command that you've typed. It is equivalent to hitting <Bs> repeatedly until every character was erased.

<Control-K> - For users with extensive Memorite experience, <Control-K> is the same as <Del>. (Memorite uses <Control-K> to erase the current line.)

<Esc> - Hitting the <Esc> key enters "command edit mode". In this mode, more extensive editing of the line is possible. Command edit mode will be explained in following paragraphs.

All of the above characters are of the editing type. The remainder are the inserting type.

<Control-A> - This will take the current location, make it a reference, and insert it into the command line. If the cursor is at [1,1], for example, then typing <control-A> would insert the characters "[1,1]" into the command line, as if you had typed it. This is useful for grabbing locations to be used in a formula. The rationale behind the character <Control-A> is that it inserts the location that the cursor is AT.

<Control-F> - This will insert the current FORMULA into the command line. Look at the second line on the screen, the formula line. Whatever is there will be inserted into the command if <control-F> is typed. If there is no formula associated with the current location, then nothing will be inserted.

<Control-C> - This will insert the CONTENTS of the current location into the command line. The data inserted will be exactly as it is displayed.

<Control-L> - This will simply insert the entire LAST command typed into the command line. This can be useful for repeating an operation, such as entering data, for several different locations. It also is useful, particularly, when a lengthy command is typed which contains an error. Rather than retyping the entire line, you can simply type <control-L> then use the editing commands, below, on the line.

Keep in mind that the above characters can be combined with the screen movement characters. For example, say you'd like to insert a formula into the command line. Not the current formula, but the formula that is associated with a particular location. You can simply move the cursor to that location, type <control-F>, then move the cursor back to where you want it!

Command Edit Mode

As mentioned above, typing <Esc> enters command edit mode. When in command edit mode, none of the above-mentioned characters work, and none of the screen-movement characters work. This mode is used when you've typed a long command and need to change it, but just don't want to retype the whole thing.

Command edit mode, or simply edit mode, is indicated by having a character of the command highlighted. At first, this will be the rightmost character of the command. The highlighted spot is itself like a cursor, insofar as that is the spot where whatever is done will take place.

While in edit mode, typing characters is just like normal command mode, except that whatever character is typed will replace the character under it.

Certain special characters exist in edit mode, and they are as follows.

<Left-arrow> *- this will move the highlight one character to the left.*

<Right-arrow> *- this will move the highlight one character to the right. It will not move any further than the cursor.*

<Lf> *- this will move the highlight to the first character of the line.*

<Control-V> *- this will enter "insert mode". At this point, when normal characters are typed, they will not replace the character under the highlight. Instead, the remainder of the line will be shifted to the right and the character typed will be inserted. Typing <control-V> a second time will leave insert mode.*

<Control-D> *- this will delete the character under the highlight. The remainder of the line will be shifted to the left.*

<Esc> *- this will leave edit mode and return to the normal command mode.*

<Return> *- this is like typing <Esc> then typing <Return>. That is, it will leave edit mode, but then proceed as if you had typed <Return> in command mode and execute the command.*

Forced References

Throughout this guide, the expression "current location" will be used. As previously explained, this means the location where the cursor is at.

There is a way to make the current location somewhere else, if desired. For example, the C command clears the current location. You might wish to clear location [20,5], but the current location is [1,1]. You could just move the cursor to the desired location and then clear it, but another method is provided.

This method is called "forcing" the current location. That is, you can make the system think that somewhere else is the current location. This is done by simply typing the reference before the command. For example, you could type

```
[5,20] C
```

Which would force [5,20] as the current location, then execute the C command which clears the current location, which would be [5,20]. Another example:

```
[INCOME,JAN] EVC 1000 1500 2000
```

This would force location [INCOME,JAN] and then execute the EVC command, *which will put the values into successive locations, starting at the current location. Of course, the current location would be what it was forced to be.*

It should be noted, though, that the forced reference is only applicable to the command with which it is typed. After execution of the command is complete, the spot where the cursor is at will again be the current location.

In general, anywhere in this guide where any of the expressions "current location", "current row", or "current column" are used, they refer to the current as defined, unless a forced reference is used, in which case they refer to the forced current location.

Manual Recalculation

The system has two modes for calculations, automatic and manual, which are selected with the SC command (which is explained in another section). When in automatic mode, entering any formula or value will cause the system to reevaluate the array. In manual mode, this will not happen. To cause a recalculation when in manual mode, press the <Tab> key. You must do this as the first character on the command line - if you are in the middle of a command, the system will ignore you. The message -Recalculating- will appear in the bottom right corner of the screen while the system is doing the calculations. When it is done, you will be returned to command mode.

Note that it is possible to have circular references such that it takes two or more recalculations before the array is fully evaluated. You can simply keep pressing the <Tab> key as often as needed.

Section 2

Commands

Complete detailed descriptions of all commands and how they are used.

C - the CLEAR command

The clear command is used to clear certain locations in the array of the data and/or formulas associated with them. The various forms are as follows.

C - Clear current location. The data and formula will be cleared from the current location.

CF - Clear current formula. If there is a formula associated with the current location, it will be erased. The data in the current location will be left unchanged.

CR - Clear row. The data and formulas for all locations in the current row will be erased. Alternate form:

CR row

In this case, (row) specified will be cleared.

CC - Clear column. The data and formulas for all locations in the current column will be erased. Alternate form:

CC column

In this case, (column) specified will be cleared.

CA - Clear all. All data and formulas in the array will be erased.

D - the DISK command

The disk command is used to access disk directories to load, save, or erase files. It is also used to select the current disk to be used for subsequent disk operations.

D - Disk commands. Disk command mode will be entered, using the currently-selected disk.

D x - Disk select/commands. Disk x will be selected, then disk command mode will be entered. The value x must generally be in the range A to P, but specifically must be a valid drive on the system being used at the time.

Disk Command Mode

When disk command mode is entered, the directory of EPL files on the selected disk will be displayed at the top of the screen; the current disk and a list of available commands will be displayed at the bottom. Furthermore, the first file name will be highlighted.

At this point, typing any of the four arrow keys will move the highlight in the appropriate direction. Once the highlight is moved to the desired file, the disk commands below can be used. All disk commands reference the currently-highlighted file.

Disk Commands

L - Load a file. The highlighted file will be loaded into the array. Note that this erases the current array; therefore, if it is desired to save the current array, it should be updated or saved before executing the load command.

S - Save a file. The array will be saved to the highlighted file. The former contents of the file are lost. This command should be seldom used, and is included only for symmetry. Normally, the update command is used to save the array.

D - Delete a file. The highlighted file will be erased from the disk. This is equivalent to using the ERA command under CP/M.

<Esc> - Exit disk command mode. When <Esc> is typed, the user will be returned to the main system.

Each of the disk commands (except <Esc>) requires confirmation. When either L, S, or D is typed, one of the messages

```
Loading <file> - type Y to proceed -
Saving <file> - type Y to proceed -
Deleting <file> - type Y to proceed -
```

will be displayed. At this point, typing Y will cause the selected action to be carried out. Any other character, including <Return>, will cancel the action.

If when disk command mode is entered, there are no EPL files on the disk, the message

```
No files...
```

will be displayed. Obviously, there are no files to load or delete; therefore the only possible action is to save the array, which in this case MUST be done with the update command. Typing any character will return to the main system.

E - the ENTER command

The enter command is used to enter data or formulas into the array. It has four forms, for entering text, lines, values, or formulas. Since the four are drastically different, they will be explained separately.

ET - Enter Text

The enter text command will accept a single argument and write it into the current location. The general format is

```
ET text
```

The format for the ET command is somewhat precise. There must be one character after the ET, then whatever is after that is considered the text. For example, in the command

```
ET Hello There<                 (the < indicates the cursor)
```

will enter the text "Hello There". On the other hand, the command

```
ET  Income Type <
```

will enter the text " Income Type ". In other words, exactly what you type is what you get. Since whatever the user types is taken as the text, no multiple form of the ET command is possible (unlike the other enter commands).

EL - Enter Line

The enter line command is used to enter a line of data where the data is simply one character repeated. This is typically used for a dividing line of some sort, or perhaps the line after a column of numbers above the total, or something similar. There are two formats of the enter line command:

```
EL character
EL character ncolumns
```

The first type will create a line consisting of the (character) all the way across the array. The second will create a line only extending across a certain number of columns, specified by (ncolumns). In either case, the current location will be used as the first (leftmost) column; that is, the column in which the line begins.

The EL command operates much like the ET command. The data in the line is

simply treated by the system like any other text. The EL command automatically creates the sequence of characters to be the exact width of each column as it is entered. This can be used to create a "broken" line, also. For example, if the widths of all columns are set to 10, then the EL command is used, then the widths of all columns are set to 12, there will be a 2-character break in the line between each of the columns.

EV - Enter Value

The enter value command and its variations are used to enter numeric values into the current location and possibly adjacent locations in the array. The simplest form is

```
EV number
```

which will write the value (number) into the current location. Additionally, this form of the command can be "implied" by simply typing

```
number
```

when in command mode. That is, typing "123" and "EV 123" are the same.

The second form of the EV command is used to enter values into adjacent locations, either across a row or down a column. The forms are

```
EVR number number number number etc.
EVC number number number number etc.
```

The first will enter as many successive numbers as are typed into the current location and successive columns on the same row; the second will do the same, only the numbers will be entered into successive rows on the same column. For example, if the current location is [1,1], the command

```
EVC 1 2 3 4 5
```

will enter 1 into [1,1], 2 into [2,1], 3 into [3,1], etc.

The final form is the "repeat" form. Is is used to enter the SAME number into successive locations. The forms are

```
EVRR number count
EVCR number count
```

The first will enter the number (number) into successive columns on the same row for (count) columns; the second will do the same down a column. For example, if the current location is [1,1], the command

```
EVCR 100 5
```

will enter the value 100 into locations [1,1] through [5,1].

EF - Enter Formula

The enter formula command is used to enter formulas into the array. It has two forms, single and multiple. The single form is

```
EF formula
```

which will enter the (formula) into the formula table. It will set the value in the current location to 0, unless a "SC A" has been done (see the set command), in which case the formula will be evaluated and the result of the evaluation put into the current location.

The multiple form of the EF command is

```
EFM formula nrows ncolumns
```

where (nrows) and (ncolumns) are the number of successive rows and columns, respectively, to write the formula into. The array section which will get the formula may be thought of as a rectangle consisting of (nrows) rows and (ncolumns) columns, with the current location as the upper-left corner. For example, if the current location is [1,2], then the command

```
EFM 1.1*[.,.-1] 8 11
```

will write the formula "1.1*[.,.-1]" to 8 rows and 11 columns, or specifically, into locations [1,2] through [1,12], [2,2] through [2,12], and so on through [8,2] through [8,12].

F - the FORMAT command

The format command is used to choose the manner in which the data in the array will be displayed and printed. There are four forms of the command, but they differ only in the portion of the array that they affect. They are

F format
FR format
FC format
FA format

F means to format only the current location. FR means format the current row, FC means the current column. FA means format the entire array.

The (format) consists of zero or more individual format characters. These characters can be listed in any order, in a generally "free" fashion. Following are the various format characters.

$ - Dollar Sign. If $ is included, numbers will be printed with a leading dollar sign. For example, 123 will be displayed as $123.

, - Comma. If a comma is included, numbers will be printed with commas inserted every three digits to the left of the decimal point. That is, the value 123 would be unchanged, but the value 1234567 would be 1,234,567.

0-15 - Digits. Including a number in the range 0 through 15 will set the number of digits printed to the RIGHT of the decimal point. For example, the value 123 with a format of 4 would be printed as 123.0000.

% - Percent. The percent character indicates that the value is to be considered a percentage, and it will be printed with a percent sign following it. In addition, the number will be multiplied by 100 before being displayed. For example, the value .13 will be displayed as 13%.

R - Right Justify. For text, the R character causes right justification.

Each location in the array is either "formatted" or "unformatted". Typing a format command without any arguments (for example, "FC ") sets a location or locations to "unformatted". In this case, numbers and text will both be displayed left justified. Numbers will be displayed in a "general" format, meaning however necessary to express the value (For example, 100 will be 100, 3.14159 will be 3.14159).

The only format option for text is R, right justify. Its absence indicates left

justification.

For numbers, formatting is a little more complicated. ANY numeric format sets the location to "formatted". When a location is formatted, numbers are RIGHT justified. This right justification should not be confused with the R character for text.

Each time a F command is used, it overrides any previous F command. For example, if a "F 2" command is issued, the current location will be set to 2 places to the right of the decimal point, typically used for "dollars-and-cents" notation. If it is then desired to add the dollar sign, the command "F $" will NOT function as expected, since it cancels the effect of the "F 2" command. The proper command would be "F 2 $" since this combines the two commands. This brings up an important point: Leaving out the number-of-digits character in a format command is the same as using 0; that is, "F $" is identical to "F $ 0".

As was indicated above, the format characters are entered in a "free" fashion. Their order is unimportant. For example, the commands

F 2$,   F $,2   F ,$2   F $, 2   and F 2 , $

are all identical. Format characters may be used in whatever combination desired, except that the combination "$%" will produce a meaningless figure; for example, 123.45 will be displayed as $12345%.

Since there is limited space available for format characters, the R and , characters are actually the same thing. That is, using a comma on text will right justify it, and using R on a number will insert commas. Since a location cannot contain both text and a number, this should not cause any problem.

H - the HELP command

The help command is used to access a screenful of assistance (commonly called a "help screen") for a particular command. There are actually three different forms of the help command.

```
H
H letter
?
```

Just typing H gives a help screen on typing commands, editing, and moving the cursor. Typing H followed by a letter gives a help screen on the command beginning with that letter. For example, HF gives help on the format command.

The help command reads the help screen from the file EPL.SYS on the currently-logged-in disk. That disk should not be confused with the currently-selected disk used for ExecuPlan! The logged-in disk is the disk that CP/M thinks is the current one. To be more specific, the disk that was in the prompt before ExecuPlan was executed. If it was A>, then the logged-in disk is drive A; if it was B>, then it was drive B, etc. If the file is not present on the disk, then the system will say "Help Unavailable".

The ? command is used to get a QUICK help screen. It tells how to get more help (via the H command) and gives a list of the command letters and their meanings. This screen is part of the program, not read in from the disk. Therefore, it is always available.

I - the INITIALIZE command

The initialize command is used to set everything back to the standard. Specifically, the command

Resets the array size to 20 x 20,
Resets the row and column titles back to 1 2 3 4 5 etc.,
Clears all main titles,
Clears the entire array,
Sets the current file to none,
and Resets the print blocks back to standard.

The format of the command is simply

```
I
```

It will display a warning message,

```
** Warning: Initialization erases ALL data - type Y to proceed -
```

at which point you may type Y to proceed with the initialization. Typing any other character (including <Return>) will cancel the command.

J - the JUMP command

The jump command is used as a quick way to move the screen around on the array, faster than using the arrow keys. There are three jump commands:

```
JB
JR row
JC column
```

The first, JB, simply jumps to the top-left corner of the array, which would be, [1,1] on an initialized array.

JC jumps to the specified column. That column will be the leftmost on the screen after execution.

JR jumps to the specified row. The row will be the top on the screen after the jump, unless the size of the array makes this impossible. If the jumped-to row is within 18 rows of the end, it will be somewhere in the middle of the screen.

K - the KILL command

The kill command is used to "kill", or remove, a row or column from the array. It does not, however, change the size of the array. Therefore, when it kills a specified row or column, it creates a new one at the end in order to keep the array the same size.

The format of the kill command is

```
KR row new-row
KC column new-column
```

where (row) or (column) is the row or column to kill, and (new-row) or (new-column) is the title to be assigned to the row or column created at the end of the array.

Example: If your array currently has 12 rows, numbered 1 through 12, and you execute the command "KR 6 13" then your resulting rows will be 1,2,3,4,5,7,8,9,10,11,12,13.

There is one VERY IMPORTANT thing to note about the kill command! Relative references in formulas which refer to or over the killed row or column will NOT be changed. In other words, they will be INCORRECT after the command is executed. In the example above, if you had a reference in row 7 which contained something like .-2, before you executed the kill command, that would have pointed to row 5; after the command, it will point to row 4.

L - the LIST command

The list command is used to produce a list on the printer of all of the formulas associated with the array. The list is printed in the order in which the formulas will be evaluated; that is, either in row-major or column-major order, depending on the current "SC R/C" setting. The format of the command is simply

L

The listing will have the heading at the top of each page

```
FILENAME        Formula list by row
                            (or column, if appropriate)
```

where FILENAME is the name of the file if one is assigned. The format of the listing is

```
[destination]       = formula
```

where [destination] is the destination of the formula, and (formula) is the text of the formula.

If during the listing it is desired to stop, typing <Esc> will cancel the command.

M - the MOVE command

The move command is used to move a row or column from one place in the array to another. The command format is

```
MR row dest-row
MC column dest-column
```

where (row) or (column) is the row or column to move, and (dest-row) or (dest-column) is the row or column to move it adjacent to.

Depending on which direction the row or column is moved, it will either be placed above/to the left of the destination, or below/to the right. Specifically: for a COLUMN, if it is being moved to the left, is will be placed to the left of the destination; if it is being moved to the right, it will be placed to the right of the destination. For a ROW, if it is being moved up, it will be placed above the destination; if it is being moved down, it will be placed below the destination.

For example, say you have the rows 1,2,3,4,5,6. If you execute the command "MR 5 3", the resulting sequence will be 1,2,5,3,4,6. If you had executed the command "MR 2 6", the resulting sequence would have been 1,3,4,5,6,2. This example is equally applicable to columns.

NOTE: Like the K command, the M command does not change relative references in formulas.

O - the OPEN command

The open command is used to open up a new row or column in the array. The command does NOT change the size of the array, therefore when a new row or column is created, the last row or column of the array is removed, and its contents lost.

The format of the open command is

OR ref-row row
OC ref-column column

where (row) or (column) is the title of the new row or column to create, and (ref-row) or (ref-column) is where to put it. If COLUMN, the new column will be to the LEFT of the ref-column; if ROW, the new row will be ABOVE the ref-row.

Example: If there are currently 10 rows, numbered 1 through 10, then executing the command "OR 7 NEW" will result in rows 1,2,3,4,5,6,NEW,7,8,9 with row 10 being lost.

NOTE: Like the K and M commands, the O command does not change relative references within formulas.

P - the PRINT command

The print command is used to print the array, or to cause what would be printed to be written into a disk file for editing with Scope or Memorite. Like the disk command, the print command is actually an entire command mode. It is invoked simply with

```
P
```

Print Command Mode

When print mode is entered, the screen will be erased and replaced with what is called the "print screen". The screen is divided into 5 "blocks", each one controlling certain aspects of what will be printed.

Block 0 - Main titles. This block is used to select how the main titles will be printed. For each of the four main titles, which will be displayed, the choice may be made whether to R - right justify, L - left justify, C - center, or X - not to print at all.

Block 1 - Print bounds. This is used to select the portion of the array to be printed. Specifically, the starting row, starting column, ending row, and ending column are specified. By proper manipulating of these bounds, an array much larger than a piece of paper can be printed on several sheets, then the sheets rearranged to form a large sheet.

Block 2 - Paper size. This informs the system of the size of paper being used, in terms of number of characters per line and number of lines per page. The width is used only for centering the titles, but the length tells the system the maximum number of lines to print on one page before skipping to the next page.

Block 3 - Row/column titles. This block allows the user to choose whether or not to have the system print the row and column titles on the report.

Block 4 - Invisible. This allows the user to set rows and columns to "invisible", meaning that they will NOT be printed, even if they are within the print bounds selected by block 1. This is most often used to prevent the printing of some type of intermediate result column. There is also a provision for overriding the invisible function, that is, to go ahead and print the invisible rows and columns.

Print Commands

In addition to the blocks, the bottom of the screen will list the available commands. Following are the commands, and how to use the blocks.

P - Print the array. Typing P will cause the array to be printed, using the settings of the blocks to define the bounds, titles, etc. If during printing you wish to stop, type <Esc>.

D - Disk. Typing D is just like P, only instead of printing the array, the data will be written to a disk file. The format of the data will, however, be identical to when it is printed.

When D is typed, the message

    Please enter file name:

will be displayed. Type the name of the file that you wish to write the data to, followed by <Return>. The file will be assumed to have .MEM as the extension, and must not already exist. If it does, an error will be displayed and the command cancelled. The current disk will be used for the file; to use a different disk, first select it with the disk command from the main system.

If when you are prompted for the file, you decide not to execute the command, simply type <Return> without typing the file name, and the command will be cancelled.

F - Formfeed printer. Typing F will simply cause a formfeed character to be sent to the printer. This normally has the effect of rolling the paper up to the top of the next page.

0 - Edit block 0. Typing 0 will allow you to change the information in block 0. Note that only the justification character can be changed at this time. To change the text of the title, the TMx command must be used when under the main system.

When 0 is typed, the justification character of the first main title will be highlighted. At this point, you have several options:

> Type <down arrow> - this will move the highlight down to the next title. If you are already at the bottom (fourth) title, the highlight will be moved back to the first one. The justification character for the current title will not be changed.
>
> Type <up arrow> - this is the opposite of <down arrow>. The highlight will be moved up to the previous title. If you are already at the top, it will be moved down the the bottom one. The justification

character will not be changed.

Type <Esc> - this will leave the current character unchanged, and return to print command mode.

Type <Return> - this is identical to <down arrow>, except that if you are at the bottom title, it will stop editing block 0 and return to print command mode, similarly to <Esc> above.

Type a justification character - typing either L, R, C, or X will set the justification character for the current title to whatever is typed. The highlight will not be moved, so if the wrong thing is typed, you may simply retype the correct character.

Anything other than the above characters will simply be ignored.

1 - Edit block 1. *Typing 1 will allow you to edit the information contained in block 1. When 1 is typed, a cursor will appear a little to the right of the first line in block 1. At this point, you are in a mode similar to block 0, but a little different. Essentially, while block 0 is an "instant" block, meaning that when you type a character, it immediately replaces the previous character, block 1 is an "updated" block, meaning that the new information appears to the right of the old information and is edited by itself, and only replaces the old information when you leave block 1 and return to the print command mode.*

Of course, the data you type is not a justification character. Instead, the appropriate title is typed. The first time print command mode is entered, the bounds are set to the size of the entire array.

When typing the title, up to eight characters may be typed, terminated by either <Return> or <Esc> (the difference is explained below). In the process of typing the title, <Bs> may be typed to back up one character.

Additionally, there are two special characters allowed, if typed as the first character. Typing <Control-F> will display the "first" row or column of the array; typing <Control-L> will display the "last" row or column. This could be useful if you wish to, say, start the printing at the first row, but you're not sure what the title of the first row is.

The same editing characters are available for block 1 as for block 0 (<down arrow>, <up arrow>, <Esc>, <Return>), but the way they work is slightly different. When by some method the cursor comes to be on a line, any previously-typed data on that line is erased. To change a single item without having the cursor move down the next line (and consequently erase something that you might have typed there), <Esc> may be typed instead of <Return>, which will immediately return to print command mode.

When print command mode is reentered, the information in block 1 will be updated based on the new information typed. If a title typed does not exist, it will be highlighted and an error displayed. Any line which contains an error will not be updated.

Although as explained this probably seems very complicated, it is virtually self-explanatory when actually done.

2 - Edit block 2. Typing 2 allows the user to modify the information under block 2, the paper size block. All aspects of block 2 are identical to block 1, except that instead of typing row or column titles, you type decimal numbers. For page width, you should type the number of characters per line. This is only used for centering the main titles. For page length, you should type the maximum number of lines you wish printed on a page. On a standard 11" page, a length of 56 lines allows reasonable margins on the top and bottom. For both length and width, the system will accept values in the range of 40 through 255.

3 - Edit block 3. Typing 3 allows the user to select whether row or column titles are to be printed as part of the report. If the appropriate line is Y, the titles will be printed; if N, they won't. Block 3 is similar to block 0, except that the items are only updated when print command mode is returned to.

4 - Edit block 4. Block 4 is used to determine which rows or columns, if any, are to be considered "invisible", meaning that they will not be printed.

Block 4 is perhaps the most confusing block, because it includes two individual fields for each line. The first is the row/column indicator, one of the characters R or C. The second is the title of the row or column.

When block 4 in entered, a cursor will appear on the top line of block 4. At this point you have several options.

The <up arrow>, <down arrow>, and <Esc> keys function the same as block 0. The exception is that when you are on the bottom (tenth) line, the down arrow moves to the "Print Inv? " field. From there the <down arrow> will move back to the first line. The <up arrow> moves in the same manner, only up instead of down.

Typing R or C will set the row/column indicator for the current line to whatever is typed.

Typing <Space> will turn OFF the line; that is, when <Return> is typed after the <Space>, it will also remove the title on the current line.

<Return> is typed to allow movement to the title field. For example, to

set a line to Row 1, you would type "R" <Return> followed by "1" as explained below. If the row/column indicator already contained an R, you could simply type the <Return> and then proceed with the "1".

If <Return> was typed, the cursor will jump three spaces to the right. At this point the system is awaiting a title. Type the row or column title that you wish to set to invisible. If the title is terminated with <Return>, the cursor will proceed down to the next line. If it is terminated with <Esc>, block 4 will be updated and the print command mode will be returned to.

When the cursor is moved to the "Print Inv? " field, you may type Y or N, in the same manner as block 3. If N is typed, the invisible function will work, that is, the rows and columns indicated will not be printed. If Y is typed, the function will be effectively overridden, that is, the rows and columns listed will be printed anyway.

When print command mode is returned to, the rows and columns in block 4 are looked up. If any of them are not found, they will be highlighted and an error displayed. However, the information will be left in the block, and when the array is printed, the invalid entries will be ignored.

Characteristics of print command mode.

The information in the print blocks remains as set until changed. In addition, all of the information is saved with the file, so when a file is loaded, all of the information in the print blocks will be the same as when the file was saved.

Q - the QUIT command

The quit command is used to return to CP/M. There are three forms of the quit command:

```
Q
QY
QN
```

If QY is typed, the array will automatically be updated; if no current file exists, you will be asked for one. If QN is typed, the array will not be updated.

If just Q is typed, you will be prompted with

```
Exiting - Type Y to update -
```

If at this point you type Y, the file will be updated and the program exited. If you type N, the file will not be updated, and the program exited. If you type <Esc>, the system will cancel the command altogether. Anything else will be ignored.

R - the ROUND command

The round command is used to change the precision of numbers in the array. Specifically, all numbers in the array that are NOT calculated as the result of a formula will be changed to match their representation on the screen. For example, if the value 1.469 is in a certain location, but the format is 2, then the number is being displayed as 1.47. Therefore, when the round command is executed, the number will actually be changed to 1.47.

The round command is executed simply by typing

```
R
```

at which time the system will warn

```
** Caution: Data will be changed - type Y to proceed -
```

If you type Y, the action will be carried out. Typing anything else will cancel the command without having any data changed.

S - the SET command

The set command is used to set or change certain aspects of the system. There are four set commands; set size, calculation, printer, and disk. These are explained separately.

The Set Size Command

The set size command is used to change the size of the array. The form of the command is

```
SS nrows ncolumns
```

where (nrows) is the number of rows to make the array, and (ncolumns) is the number of columns to make the array. When the system is initialized, the size is 20 x 20. To change to a 9 x 13 array, for example, the command would be

```
SS 9 13
```

If the number of rows or columns is being decreased, the message

```
** WARNING - Some rows containing data may be lost - type Y to
proceed -
```

will appear. If the number of columns is being decreased, the message will indicate columns instead of rows, of course. Typing Y will cause the action to be carried out; anything else will cancel the action. If you are decreasing both rows and columns, you will get both messages, one at a time. Responding Y to rows but something else to columns will result is the number of rows decreasing, but the number of columns remaining the same.

If is is desired to only change one dimension, the value 0 may be put in the other. For example, to change the array from 20 rows to 30, but not affect the number of columns, the command would be

```
SS 30 0
```

NOTE: There is a substantial amount of logic involved when shrinking the array, particularly when columns are involved. If an array with many rows has the number of columns lessened, several moments (read a minute or so) could elapse before the system completes the command. Therefore, do not fear system failure. SUGGESTION: When changing the size of the array, the rows are changed first. Therefore, say you're changing the 20 x 20 array to 10 x 100. The system will first create 100 rows, then shrink to 10 columns.

This will take quite a while, since the system is removing 10 columns of 100 rows each! To make the operation faster, FIRST change the number of columns (SS 0 10) THEN change the number of rows (SS 100 0). Using this method, only about 3 seconds will be used, a substantial increase in speed. If you are decreasing the number of rows and increasing the number of columns, FIRST decrease the rows, THEN increase the columns. The object is to make the number of rows as small as possible when creating or removing columns.

The Set Calculation Command

The set calc command is used to change the order and frequency of when the formulas are evaluated.

Formulas may be evaluated in either "row-major" or "column-major" order. If they are evaluated in row-major order, that means that they will be evaluated across each row, then down to the next row and across it, etc. In other words, the same way you read. In column-major order, they are evaluated down each column, then moved over to the next column and down it, etc.

The frequency may be set to either "manual" or "automatic". In manual mode, the formulas are only evaluated when the <Tab> key is pressed. In automatic mode, they are evaluated whenever a value or formula is entered.

The commands accepts the letters R, C, A, and M to represent each of the above options. Obviously, only two may be entered at once, but they can be entered in any order. For example, to set the calc order to column-major, and the frequency to automatic, the command would be

```
      SC C A          or              SC A C
```

To change only, say, the calculation order, this time to manual, the command would be

```
      SC M
```

While it will not produce an error to enter both R and C or both A and M at the same time, only the last one on the line will be used.

The Set Printer Command

The set printer command is used to send special characters to the printer. This should seldom be used, but is provided for special cases. The most common use would be so set a printer for some special mode.

The format of the command is

```
SP byte byte byte etc.
```

where (byte) is a decimal number. As many or few bytes may be sent as needed. For the exact bytes required for particular printers, you will have to refer to the appropriate printer's manual.

For an example on what this might be used for, consider the Vector Matrix printer. If you typed the command

```
SP 27 80
```

which corresponds to "ESC P" (the printer manual explains this), the printer would switch between 132 and 80 characters per line.

The Set Disk Command

The set disk command is used to reset the disk system in case you wish to change floppies. CP/M will normally not allow you to do this; it you do, you will get a BDOS read-only error, which can't be recovered from. Therefore, if you wish to change a floppy disk, put the new diskette into the drive and type

```
SD
```

which will tell CP/M about the new disk and allow you to access it.

T - the TITLE command

The title command is used to set and delete the main titles for the array, and to change the row and column titles. The commands for the row and column titles are substantially different from the commands for the main titles, therefore they will be explained separately.

Main Titles

The first two forms of the title command are for handling the main titles. They are as follows.

```
TMx text
TDx
```

where (x) denotes the number of the referenced main title. Since there are up to four main titles, (x) must be in the range of 1 to 4. If you are referencing the first main title, the 1 may be skipped. That is,

TM text	is the same as TM1, and
TD	is the same as TD1.

The (text) specified in the TM command refers to the text of the title, that is, what you wish the title to be. For example, the command

```
TM Budget Forecast
```

will set the first main title to "Budget Forecast".

The TD command is used to delete a main title, which is to say, make it blank.

The first main title is displayed on the screen, centered in a field of dashes. The remainder of the main titles can be viewed by using the P command, where they will be displayed in block 0 (see the P command). Note that when a main title is entered (or changed), the justification character is set to L. When a title is deleted, the justification character will be changed to X. It is possible to make a "comment" by typing a main title, then changing its justification character to X, thus keeping it from being printed.

Row and Column Titles

The second pair of title commands are used to change the row and column titles. The commands are

```
TR old-title new-title
TC old-title new-title
```

The TR is used to title a row, and TC is used to title a column. The (old-title) denotes the current title of the row or column you wish to change, and (new-title) denotes the new title you wish to assign to the row or column. For example, to change the title of row 1 to INCOME, the command would be

```
TR 1 INCOME
```

If you then changed you mind, and wanted to make the title SALES, the command would be

```
TR INCOME SALES
```

Note that the second time, the previously-assigned new title, INCOME, was used to refer to the row.

It is acceptable to use the relative method of referencing when specifying the old title. For example, to change the title of the current column to "JAN", you could use the command

```
TC . JAN
```

Titles for rows and columns may contain any characters, but must be only one word with a maximum of 8 characters, and must not start with a period (.). This is to prevent confusion with relative references.

U - the UPDATE Command

The update command is used to update the disk with the current contents of the array. Assuming there is a "current" file, all that need be typed to update the disk is

```
U
```

and the update will occur. If there is no current file, the system will say

```
No "current" file to update; Please type NEW filename:
```

and await your response. Type the name of a new file and <Return>. The system will create the file and save the array into it. In addition, it will be made the current file. If the file already exists, an error will occur. When asked for the file name, just typing <Return> will cancel the command.

Under certain circumstances, it may be desirable to save the array to disk, but NOT under the current file. This can be done by typing

```
UN
```

which stands for "Update New". The system will prompt with

```
Please type NEW filename:
```

and the proceed as above for a new file.

V - the VERIFY command

The verify command is used to verify the size of the array, or determine what the current actual cursor position is.

TAKE NOTE: Say for example that you have rows titled

HDGS, MONTHS, -, 1, 2, 3, 4, and 5.

Now you wish to remove ONE row. Your inclination will be to type the command "SS 4 0". This is not correct, however! Although the last row's title is 5, it is not the fifth row! If you typed that command, you would lose rows 2, 3, 4, and 5, which was certainly not your intention!

To eliminate the problem, two commands are provided. They are

```
VS
VC
```

The first, VS, tells the size of the array. The second, VC, tells the current cursor position on the array, in terms of absolute position. The VS command causes the message

```
Current size is x rows by y columns.
```

to be printed, where (x) is the number of rows and (y) is the number of columns. The VC command causes the message

```
Cursor is at row x and column y.
```

to be printed, where (x) and (y) are the row and column, respectively, where the cursor is at. In the example above, if the cursor was in the first column and on the row titled "5", then the VC command would result in the message

```
Cursor is at row 8 and column 1.
```

Typically, you might move the cursor to the first row (or column) that you wish to remove, then use the VC command, which will tell you the actual number of that row or column. If, on the other hand, you know that you wish to decrease the size of the array by a certain number of rows or columns, the VS command would be most useful.

By utilization of these commands, the possibility of accidentally destroying data with the SS command should be reduced.

*W - the **WIDTH** command*

The width command is used to set the widths of columns in the array. There are two forms provided.

```
W column width width width etc.
WA width
```

The first form allows you to specify (column), which is the column to start with, and a many (width)s as desired. Each (width) typed will be assigned to the following column. For example, assume that columns are titled JAN, FEB, MAR, and so forth. The command

```
W APR 20
```

will set the width of column APR to 20 characters. The command

```
W JAN 15 15 25 25 7
```

will set the width of columns JAN and FEB to 15, columns MAR and APR to 25, and column MAY to 7.

The second form will assign all columns the (width) specified.

Columns widths may be in the range of 2 to 64 characters, although columns narrower than about 6 characters begin to get useless.

X - the EXCHANGE command

The exchange command is used to exchange, or swap, two rows or columns. The commands have the form

```
XR row-1 row-2
XC column-1 column-2
```

where (row-1) and (row-2) or (column-1) and (column-2) are the rows or columns to be exchanged. All aspects of the rows or columns are exchanged - the data, the formulas, the titles, and if columns, the widths.

NOTE: As with the K, M, and O commands, relative references in formulas affected by the execution of the command will not be changed. References over an exchanged row or column will not be bothered. References TO the row or column will simply get the new data instead of the old. However, references CONTAINED in the row or column will now be evaluated relative to the new position, whereas they were entered relative to the old position. This could potentially result in incorrect calcalations, so beware.

Section 3

Errors

All errors, what they mean, what causes them, and how they can be avoided.

ERRORS

From time to time, something will be done wrong. ExecuPlan has a vast number of error messages to help you figure out what was haywire.

When an error occurs, the error message is displayed in the far bottom-right corner of the screen. Normally, a character or word somewhere on the screen is also highlighted. That character or word is the source of the error. Not all errors, however, have this feature.

After the error is displayed, the system just stops and waits for the user to type something. As soon as a character is typed, the system proceeds. Most errors result in the system returning to command mode and awaiting another command. Some, however, have other results. Certain math errors, for example, simply warn you; when you type a character, processing continues.

If an error occurs while formulas are being evaluated, then an additional message is displayed on the command line telling you where the formula is that caused the problem. Also, the formula itself will be displayed on the formula line of the display.

Following are explanations of all of the errors and their causes.

- COMMAND ERROR -

The command error indicates that the command typed is invalid. If the first character is highlighted, then that is the invalid command. If the second character is highlighted, then the first character is valid, but the second one is not.

- SYNTAX ERROR -

Syntax error indicates one of several things. If the first character following the command is not a space, then a syntax error will result. If an invalid character is encountered while a decimal number is being read, that will also cause a syntax error. There are also a couple of other obscure conditions that will cause a syntax error. The character highlighted will normally be the character that was undigestable.

- MISSING ARGUMENT -

This indicates that something was expected, but nothing was found. For example, typing a title command but leaving out one of the titles will cause a missing argument error. A character is not always highlighted, but if one is, it is at that point that another argument was expected.

- TITLE TOO LONG -

This indicates that a title was being read, but more than eight characters were found in the title. The ninth character will be highlighted.

- TITLE NOT FOUND -

This should be pretty obvious. A title was read, but there is no row or column with that title. The entire title that was not found will be highlighted. Also, typing a relative reference that refers to someplace off the array will cause this error.

- INVALID RELATIVE -

When a title is being read, if the first character is a period (.), then it assumes that a relative reference is in the works. If the character(s) following the period do not make sense, then this error will result.

- DUPLICATE TITLE -

Duplicate title indicates that a title was entered that should not already exist, but it does. An example would be the second argument in a TR command. This error is also used when a disk file name is typed for a new file, and the file already exists. The entire title (or file name) will be highlighted.

- OUT OF RANGE -

Certain commands expect a number within a certain acceptable range. If the number typed in not within that range, the out of range error will be the result. For example, typing a column width less than 2 or greater than 64 would cause this error. The number will be highlighted.

- BAD FORMAT CHAR -

A character is encountered in a F command that is not acceptable. The character that was unacceptable will be highlighted.

- DECIMALS > 15 -

This is similar to the above in that it means that something is wrong in a F command. This error, however, indicates that a number for the decimal count is too large. Fifteen is the maximum number of decimal places that may be specified. The offending number will be highlighted.

- BAD CALC ORDER -

During the reading of a SC command, a character other than R, C, M, or A was reached. The character is highlighted.

- INPUT ERROR -

An input error occurs when an EV command is being executed and something wrong is reached. Normally, this is a decimal number containing some garbage characters. The number being read is highlighted.

- EXCESS INPUT -

This is reached during one of the multiple forms of the EV or EF commands. For example, if the command "EVCR 45 100" is given, but there are only 50 rows, an excess input error will be generated. Whatever portion of the command that caused too much input to occur will be highlighted. Under the EV command, the extra will be ignored. If the command is EFM, the whole command will be cancelled.

- INVALID DRIVE -

When a D command is executed and a drive is specified, that drive must be in the range A to P. Anything outside of that range will cause an invalid drive error. The offending character will be highlighted. Note that even something within the range A to P may be invalid, since few systems have 16 disks! However, the system reqlly doesn't know that, hence the extended range. If you type a drive within the range that doesn't exist, then CP/M will get into the act and give a BDOS error. These are unrecoverable!

- WRITE PROTECTED -

Under CP/M version 2, files may be set to "read-only" status. Trying to write to or erase such a file will result in this error. Nothing is highlighted.

- DISK I/O ERROR -

This error means one of several things. One possibility is that there is a physical error on the disk. Another is that an attempt is being made to write to the disk, but there is no more room. Finally, an attempt may be being made to read a file which is goofed up somehow. Nothing is highlighted by this error.

- FORMULA ERROR -

This is actually a rather general error. What it means is that there is something wrong in a formula, such as a non-existant function, an improperly-typed number, an invalid operator, or something else along that line. The system will try to highlight the character that caused the problem, but depending on the cause, that character might not actually be the source of the error.

- BAD RANGE BOUNDS -

When evaluating a multi-argument function, the arguments were invalid. There are actually two separate things that could be wrong. First, the arguments aren't references at all; second, they could be references, but define an invalid range for the function. The arguments for such a function must be, respectively, the top-left and bottom-right corners of a rectangle. The rectangle may in fact be a line, or even a point. However, the second reference can't have a row or column that is less than the row or column in the first reference. Nothing will be highlighted.

- MATH IMPOSSIBLE -

Certain things just can't be done with real numbers, and things like logarithms or square roots of negative numbers are such things. Nothing will be highlighted.

- DIVISION BY ZERO -

The cause of this is quite apparent. The particular thing about this error is that the character typed to recover from the error condition determines what will be used as the result of the operation (that caused the error). If the character "0" is typed, then zero will be used as the result. If the character <Esc> is pressed, then the system will abort the operation and return to command mode. Any other character will cause the value 9.99999999999999 times 10 to the 35th power to be used as the result of the division.

- OVERFLOW -

Some math operation resulted in a number that is just too big. The largest possible number or something near it will be used instead. Nothing will be highlighted.

- OUT OF MEMORY -

This indicates that there is too little memory available to carry out the operation requested. Normally, there is around 30K of free space to start with. With gobs of text and formulas in memory, this can disappear quickly. When an operation would result with less than about 100 bytes (characters) of free space, this error is caused. The padding is allowed because certain operations use some memory during their execution. Note that if this error is given during an EFM command with large arguments, there might really be enough room. The system allows for maximum tolerences when calculating the space available. If you think there is enough space, try reentering the formula, but in smaller multiples. Nothing is highlighted by this error.

- HELP UNAVAILABLE -

This indicates that the help command was used, but the help file was not found. The help file is called EPL.SYS, and must reside on the logged-in disk under CP/M. Either the disk containing EPL.SYS was not in the drive, or the user is assuming the wrong drive is the logged-in one. Nothing is highlighted.

Section 4

Math Capabilities

How the math package is used, how formulas are formed, and explanations of all operators, functions, and special capabilities.

MATH CAPABILITIES

ExecuPlan has a very powerful math package incorporated into it. Virtually all operators and functions necessary for any type of calculations are provided. Furthermore, their usage is in a simple, algebraic format.

The EF command is used to enter formulas into the system. For example, the command

```
EF AVG([1,1],[5,1])
```

would enter a formula which would compute the average of locations [1,1] through [5,1].

Characteristics of Formulas

This section will be devoted to explaining exactly how formulas are formed. If you, the reader, are familiar with the programming language BASIC, then suffice it to say that ExecuPlan handles formulas the same way. Assuming that you're not, then read on.

Formulas are essentially a list of items, where each item is either data of some type, or an operator. Things like numbers or functions are data; plus, minus, and so forth are operators.

Data and operators are simply strung together to form an algebraic expression. For example,

```
3+4+9
```

is a valid expression, containing numeric data and the operator "+". Under ExecuPlan, most formulas will "reference" locations in the array. A sample formula with a reference would be

```
5*[1,1]
```

which means "take the number 5 and multiply it by the contents of location [1,1]".

One of the more powerful features is that of functions. These can be confusing, because while they perform an operation like an operator, they are treated as data, because when evaluated, a function is a value. A simple function might be

```
SQR(15)
```

which means "take the square root of 15". However, when contained in an expression, such as

```
[3,5]-SQR(15)
```

note that it is treated like data. Note also the syntax of a function: the function itself, followed by a left parenthesis, then the data the function is to be performed upon (the argument), then a right parenthesis. Within the parentheses can be another expression, such as

```
SQR(45-[3,5]*12)
```

which will take the square root of the result of the expression which is its argument.

Parentheses may also be used as part of a formula, besides being used to enclose the argument for a function. They are used just as in algebra, to represent a partial result. For example

```
[1,5]/(3-4*[2,5])
```

which means to take the contents of [1,5] and divide it by the result of the parenthesized expression. Unlike most systems, ExecuPlan will not complain if there are not a matching number of left and right parentheses. Instead, it will just ignore the extras. Also, parentheses may be nested to any level, that is, you may have as many as you need to properly represent your expression.

There are really only two rules with regard to formulas. First, they CANNOT contain ANY SPACES (blank characters). The first blank encountered is considered the end of the formula, and the extra past it will either be ignored or cause an error. Second, a formula can only be as long as you can type, which limits it to about 74 characters.

Precedence of Operators

Given an expression, the question arises as to in what order to evaluate the operators and functions. There are two normal ways to do this. One is called "left-to-right", and means that the operators are evaluated in the order they are encountered. The order is called "precedence", which means that they are evaluated in a specific order with certain operators first, regardless of the order they're in.

It has been said that business people use the l-to-r method, and scientific people use the precedence method, and that since computer programmers are scientific types, that's why computers always use the precedence method. Well, that may be true, but the programmer of this system is just as much a business type as scientific type. The reason that precedence was used is simply that it is more powerful; that is, certain operations cannot be done as easily with the l-to-r method. Besides, by now most business types are so used to precedence that it would cause even more confusion to have a program use l-to-r!

Before getting to precedence, though, it might be a good idea to mention what the operators are! There are five of them:

+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation (that is, raise to power. 2^3 is 8.)

So much for the operators. Now, basically their precedence is as follows:

1. parenthesized expressions
2. functions (remember that functions are evaluated, then treated as a value from that point on)
3. exponentiation
4. multiplication and division
5. addition and subtraction

When operators of equal precedence are met, then those operators are evaluated left-to-right.

Examples:

3+4*5	4 * 5 first, then add 3
3*(3+5)	3 + 5 first, because it's parenthesized
12/int(3*11)	3 * 11 first, then function "int", then division.
4+4-2*7	take 4, add 4, then subtract the product 2*7 (hence the result is -6)

NOTE: Often it will be desired to use a negative number, for example -3, in an expression. Therefore, it should be explained how it will be handled by the formula evaluator.

Essentially, whenever two consecutive operators are encountered, the program inserts a 0 between them. Thus, the sequence 2++4 would result in 2+0+4, which would give the presumed correct answer. In some cases, however, an incorrect answer might be arrived at. For example, the sequence 4-3, which should evaluate to -12, will evaluate to 4*0-3, which is -3.*

Normally, two consecutive operators should never be used. The example above, however, is a valid possibility. It is an example of the unary "negative" operation, which is the only usual possibility.

To eliminate the problem, use parentheses around such an operation when it in used in an expression. For example, -SQR(2) would not need parentheses, but 4-3 would, so you'd enter 4*(-3).*

Defined Constants

There is one other nice little feature of the program, defined constants. These are simply a couple of numbers that may or may not be used very much, but will save some typing when they are. The defined constants are

#PI	The value PI, 3.1415926 etc.
#E	The value e, 2.718281 etc.
#RND	A random number in the range 1 to 2.

Defined constants can be used wherever a number would normally be used.

Functions

At this point, we'll take a look at the functions provided in ExecuPlan. There are basically two types of functions, single-argument and multi-argument. A single-arg function is something like square root; a multi-arg function would be something like standard deviation.

These two types of functions are slightly different, beyond the obvious fact that they take a different number of arguments. A single-arg function can take anything as an argument - a number, defined constant, another function, a reference, even a whole expression.

A multi-arg function requires that the arguments be references. Specifically, these references represent the top-left and bottom-right corners of a

rectangular portion of the array. The rectangle may actually be a line, or even a single location, but in a manner of speaking, these are still rectangles.

For example, the arguments ([1,1],[5,5]) define a 5 x 5 rectangle. The arguments ([1,1],[5,1]) define a vertical line; the arguments ([1,1],[1,5]) define a horizontal line. The arguments ([1,1],[1,1]) define a point. Nevertheless, they would all be acceptable. The arguments ([3,3],[4,2]) would, however, not be allowed, since the second reference is to the LEFT of the first. It would also not be allowed if it were ABOVE the first.

The reason for this restriction should be apparent. The functions which take multiple arguments operate on a range, that is, a group of values. Only by specifying the bounds of the range, as references, can the function possibly know what numbers to use.

Naturally, there is always an odd case. Here, it is the net present value function, which requires both a range and numeric arguments. The exact format of this function will be explained when the function is explained.

Following are explanations of all of the functions, how they're used, what they do, and which type they are.

ABS **Absolute Value**

Type: single-arg

The absolute value function returns the absolute value if its argument. In other words, if the argument is positive, it is returned unchanged. If it is negative, it will be made positive. Example:

ABS(3) = 3 ABS(-4) = 4.

INT **Integer**

Type: single-arg

The integer function returns the greatest integer less than or equal to the argument. Example:

INT(3) = 3 INT(#PI) = 3 INT(5.9) = 5 INT(-1.1) = -2.

SIN	***Sine***
COS	***Cosine***
TAN	***Tangent***
ATN	***Arctangent***

Type: single-arg

These functions return the result of the appropriate trigonometric function. The argument is expected to be in radians (with the exception of ATN, which returns its result in radians).

LN	***Natural Logarithm***
LOG	***Decimal Logarithm***

Type: single-arg

These functions return the appropriate log of the argument. LN is the natural, or naperian, log (base e), while LOG is the decimal (base 10) log. Example:

LN(#E) = 1 LOG(100) = 2.

EXP ***Exponent***

Type: single-arg

This function also commonly called antilog. It returns the natural (base e) antilogarithm of the argument. Example:

EXP(1) = e

SQR ***Square Root***

Type: single-arg

The square root function returns.... the square root of its argument. Bet you would have never guessed.

SUM	***Summation***

Type: multi-arg

The sum function returns the total of all of the numbers in the range specified. For example, if [1,1] through [5,1] contain the values 1, 2, 3, 4, and 5, then

SUM([1,1],[5,1])	*returns the value 15.*

MIN	***Minimum***
MAX	***Maximum***

Type: multi-arg

The MIN and MAX functions return the smallest or largest, respectively, number is the range. Assuming the conditions above (in the SUM explanation), MIN would return 1 and MAX would return 5.

AVG	***Average***
MEAN	***Mean***

Type: multi-arg

The AVG and MEAN functions are the same thing - the average of the numbers in the range specified. Both are provided so that whichever term is preferred by the user may be used.

VAR	***Variance***
SD	***Standard Deviation***

Type: multi-arg

The VAR and SD functions compute the variance and standard deviation, respectively, of the argument range.

COUNT Counter

Type: multi-arg

The COUNT function will simply return the number of items in the argument range. This function is used by the average, variance, standard deviation, and net present value functions. IMPORTANT NOTE: This function, and therefore all of the functions that use it, react in a certain way to invalid contents of locations in the array. That is, when a certain location within the argument range does not contain a number (instead, it contains nothing or text), the value zero will be used instead. The location will still be counted! Therefore, any of the above-mentioned functions could return an invalid result if any locations in the argument range are invalid.

NPV Net Present Value

Type: special multi-arg

This function returns the computed NPV of the argument range, using additional numbers specified in the arguments. The standard formula for net present value is

$$\sum_{t=1}^{N} \frac{F_t}{(1+k)^t} - I$$

where F(1), F(2), through F(n) are cash returns for years 1 through n, k is the interest rate, and I is the initial cost. The format for the NPV function is

NPV([bnd-1],[bnd-2],k,I)

where k and I correspond to the same variables in the formula. The value for n is computed as the COUNT of the locations in the range bounded by [bnd-1] and [bnd-2].

Section 5

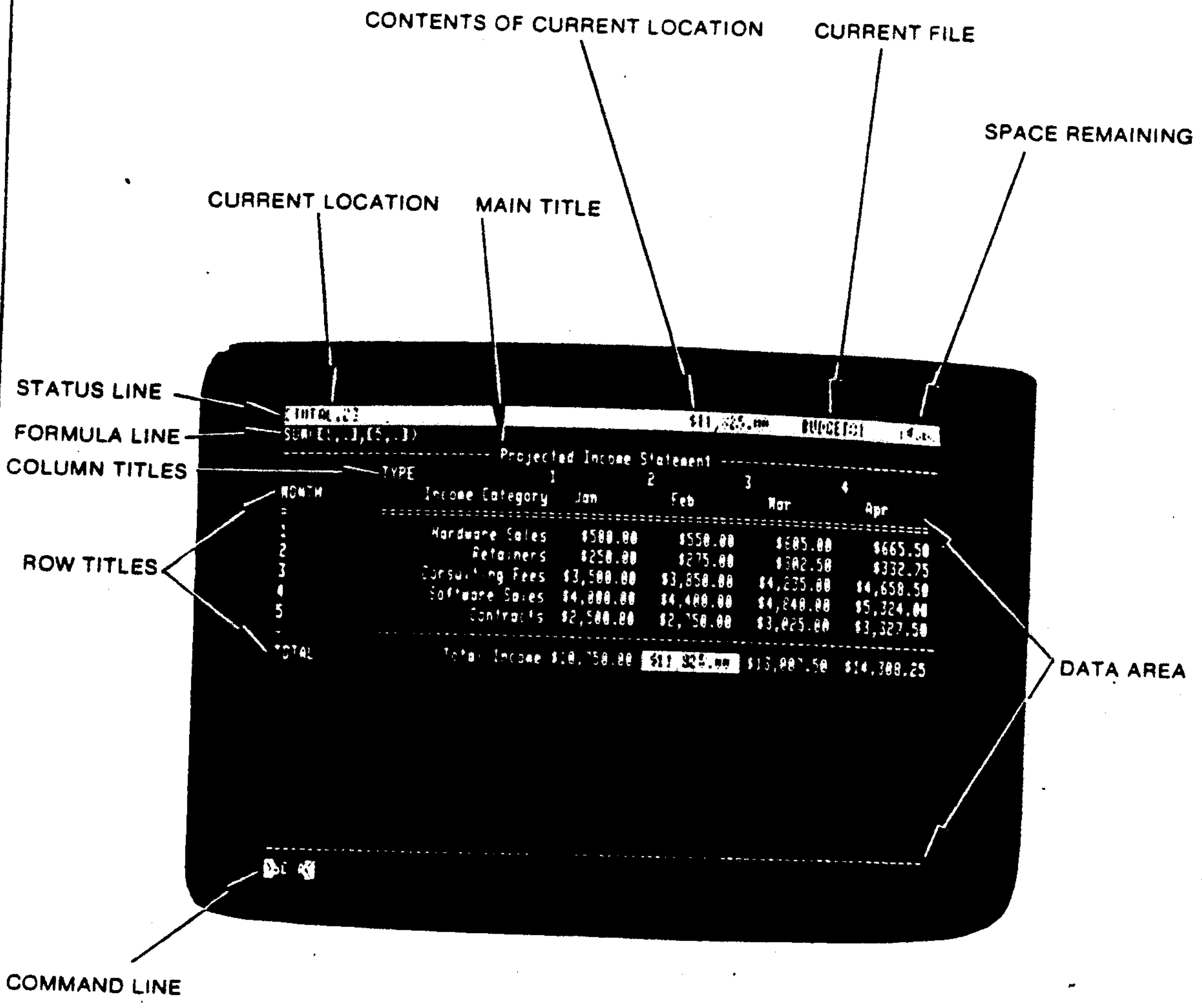
Miscellaneous

A miscellaneous collection of information that may help the user from time to time.

Appendix

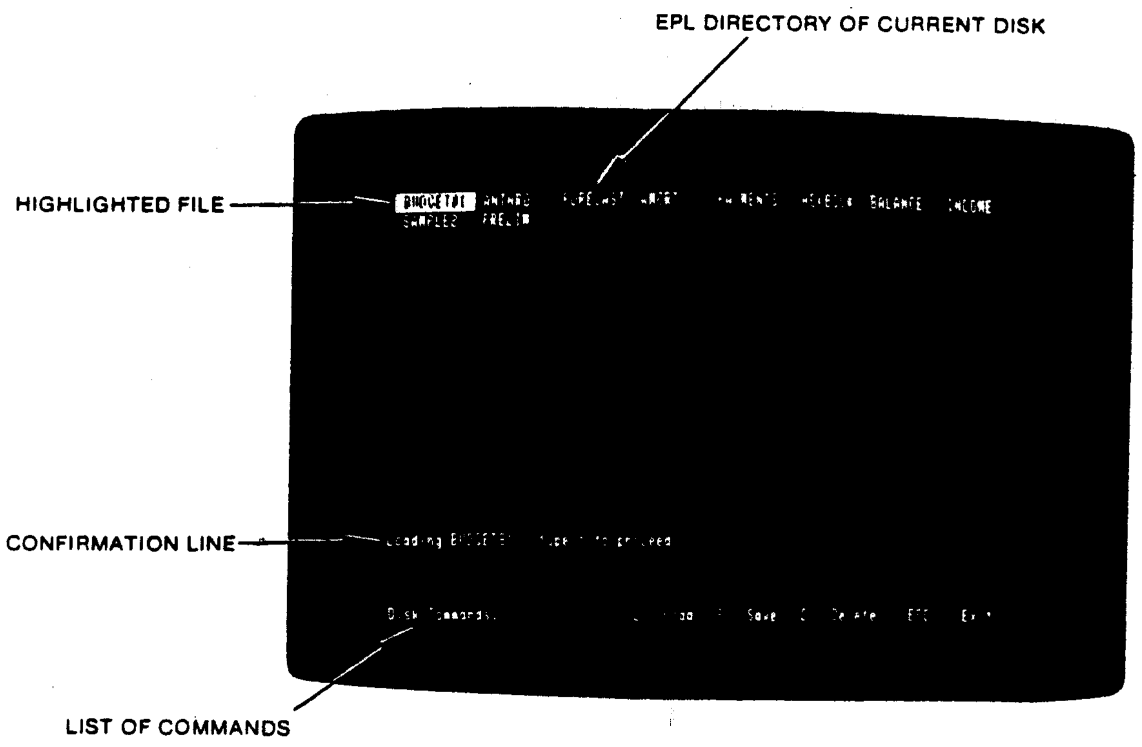
Sample Screens

Some sample screens to assist in understanding how various information is displayed and where it is displayed at.



```
[TOTAL,2]                                  $11,825.00   BUDGET:1     [illegible]
SUM([1,.],[5,.])
------------------------- Projected Income Statement --------------------------
TYPE                      1            2            3            4
MONTH     Income Category  Jan          Feb          Mar          Apr
=         ==========================================================
1          Hardware Sales    $500.00      $550.00      $605.00      $665.50
2              Retainers     $250.00      $275.00      $302.50      $332.75
3        Consulting Fees   $3,500.00    $3,850.00    $4,235.00    $4,658.50
4         Software Sales   $4,000.00    $4,400.00    $4,840.00    $5,324.00
5              Contracts   $2,500.00    $2,750.00    $3,025.00    $3,327.50
-         ----------------------------------------------------------
TOTAL       Total Income  $10,750.00   $11,825.00   $13,007.50   $14,308.25
```

Figure 1: Sample Display

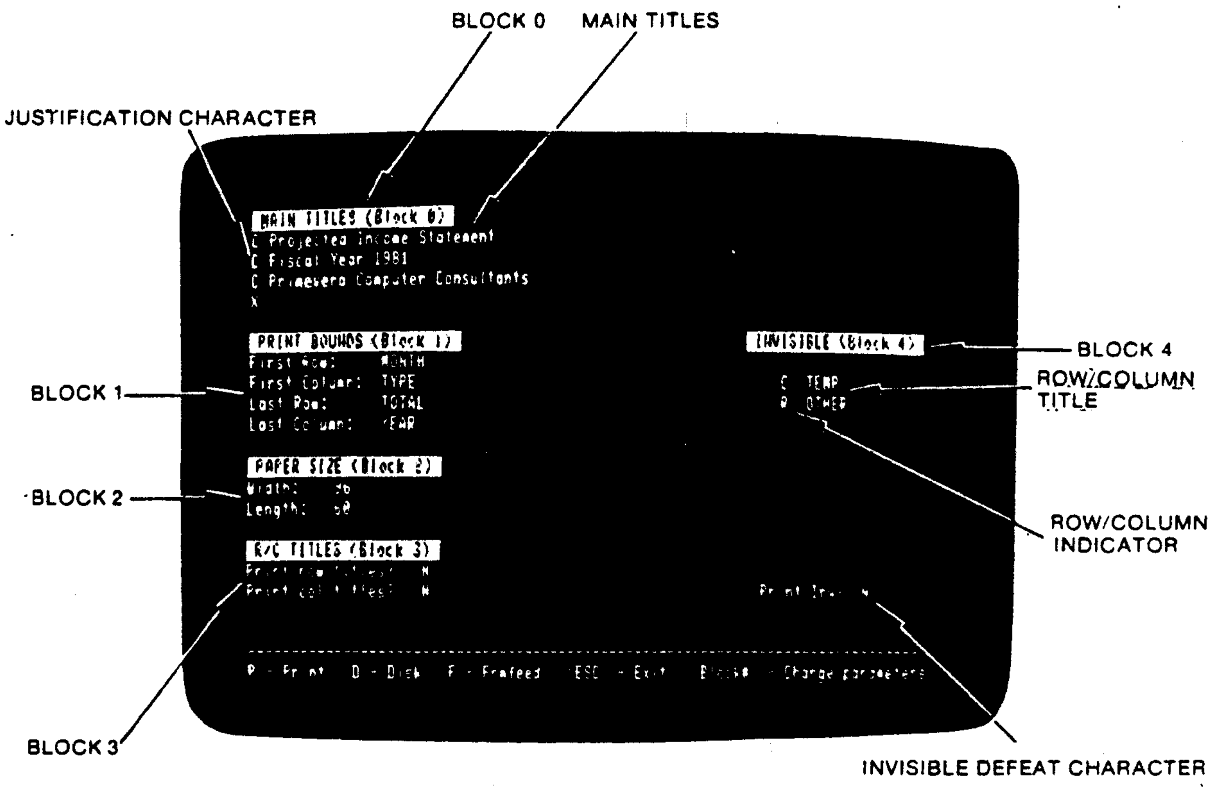


```
BUDGETOT ANTHRO   FORECAST WMART   PAYMENTS CHECKBK BALANCE  INCOME
SAMPLE2  PRELIM

Loading BUDGETOT  Type [illegible] to proceed

Disk Commands:              L Load  S Save  D Delete  ESC  Exit
```

Figure 2: Sample Disk Screen



```
MAIN TITLES (Block 0)
C Projected Income Statement
C Fiscal Year 1981
C Primevera Computer Consultants
X

PRINT BOUNDS (Block 1)                                INVISIBLE (Block 4)
First Row:      MONTH
First Column:   TYPE                                      C  TEMP
Last Row:       TOTAL                                     R  OTHER
Last Column:    YEAR

PAPER SIZE (Block 2)
Width:     36
Length:    60

R/C TITLES (Block 3)
Print row titles?   N
Print col titles?   N                                 Print Inv:  N

------------------------------------------------------------------------
P - Print   D - Disk   F - Frmfeed   ESC - Exit   Block# - Change parameters
```

Figure 3: Sample Print Screen

INDEX

Memory Utilization

This information is provided so that the user may have some idea of how memory is used under ExecuPlan. This information is somewhat advanced, and if you don't understand it, don't worry, it doesn't matter.

In order, ExecuPlan keeps the following tables: Column widths, Row titles, Column titles, Primary addresses, Numbers, Formulas, and Strings. All of the tables start from the end of ExecuPlan and build up, except for the strings, which start at the end of memory (actually the base of the BDOS in CP/M) and build down.

The column width table takes one byte per column, that byte being the width of the column.

The row and column title tables take eight bytes per title.

The primary address table, which is used as a giant reference table for the array, takes three bytes per location on the array. The first two bytes are a relative pointer to the actual data in memory, the last byte is the format byte for that location.

The numeric table holds all of the numbers. This table is dynamic, that is, only as many numbers as are actually in the system are kept. Numbers take eight bytes each, and are stored in Microsoft double precision floating point format, which yields 16 digits of precision.

The formula table holds all of the formulas. Unlike the numeric table which is pointed to, the formula table is fully independent. Only the beginning is pointed to. Each formula has a length byte, a destination row and column (taking two bytes), the text of the formula, and then a termination byte. Therefore, formulas take up the number of bytes in the text, plus four.

Text, known to a computer as strings, is stored in the string table. The string table is simply sequentially allocated, down from the top of memory. The strings are stored in reverse order since the table builds down. There are no overhead bytes with strings (the end is indicated by bit 7 on, and the beginning is pointed to from the primary address table), strings take only as many bytes as the string is long.

System Recovery

From time to time, some type of error might occur that will result in the user being dropped out of ExecuPlan into CP/M or the Monitor.

The most common possibility is that you might accidentally try to access a disk drive that does not exist. CP/M will respond with a message like

```
BDOS Error on D: Bad Sector
```

or something like that. This type of error is called "fatal", because there is no direct way to recover from it. Another possibility is that you might accidentally hit the reset button.

In any case, the probability is good that there was something you were working on that you don't want to lose. Therefore, it is nice to be able to recover from these conditions.

Monitor

From the Monitor, type "G 0100" and see what happens. Chances are, you should be right back in ExecuPlan. You might have to type "JB" to clean up the screen.

It is suggested that you dismount the disk just in case. There could be a possibility that the memory image is goofed up, and that might cause crazy things to happen. Better safe than sorry.

CP/M

More common is the case where you get dropped back into CP/M because of a disk error or read-only error. When you get the message mentioned above, or one like it, hit <Return>. You will then be back in CP/M. Now, type the following command:

```
A>SAVE 0 HOPE.COM
```

What this does is to create an empty file on the disk, without disturbing the memory image. If you already have a file called HOPE.COM on the disk, use another name. Now type "HOPE". With any luck, you will be back in ExecuPlan and can continue. If this does not work, reset the computer and proceed as explained in the above section. If that doesn't work, there is probably no recovery possible.