platinum software

WORKSAVER PLUS

user's manual

TABLE OF CONTENTS

WORKSAVER PLUS WHAT???

When the WORKSAVER was first introduced, we recieved many compliments for the convenience it added to program entry and debugging. As users of the WORKSAVER, we also agree that it certainly makes programming easier, but there are still certain tasks which the computer should do for us to make programming even easier and more enjoyable. One command that is used in an almost tedious manner is the LIST command, so we have added two features to eliminate its tedious use. The first is LIST ON ERROR which will list a program line automatically when BASIC encounters an error. The second is word processor-like scroll control of BASIC program listings. Another useful feature for programmers is global search and replace that allows variable names to be searched and/or changed automatically throughout a program. Since many programmers find it difficult to avoid rearranging BASIC programs, we decided to make a line renumbering routine that will renumber all line references throughout a program whenever a block of lines is moved. To top off the PLUS, we added two more minor features. One allows you to program the screen to pause after a predefined number of lines have been written on the screen, and thereby keep information from scrolling off the screen before you can read it. The other feature allows you to dump information from the screen to the printer or use your computer like an intelligent typewriter.

TECHNICAL HELP

We hope you find the explanations in the following sections to be easy to follow and the features easy to execute. Should you have any difficulties we are most willing to assist you with your problem. Our technical person is Bob McLennon, and he can be reached by phone at 1-802-878-2436. Please call evenings 8:30 to 10:00 Eastern Time.

The PLUS is an advanced programming aid, packed with powerfull features. We believe the program is quite simple to use, but we realize getting aquainted with all the features will require spending a few hours covering the material in the WORKSAVER and PLUS MANUALS. You won't have to do that right away however, the next couple of pages will get you started and using the program right away. You can leave the details until later when the newness has worn off, and you feel comfortable tackling the rest. Eventually, the quick reference card will be all you'll probably ever need.

LOADING THE PLUS

CASSETTE VERSION

The PLUS cassette contains the same initialization routines as the WORKSAVER. It therefore, loads the same as the WORKSAVER, and transfers to disk the same. These instructions are covered in the WORKSAVER manual page 1 'GETTING STARTED'.

DISK VERSION

The PLUS disk has three versions of the program, and each one is geared for a different computer memory size.

16K DISK COMPUTER

The 16K owner can run the PLUS from disk by entering;

```
LOADM"WPLUS-A1"
```

NOTE: The A1 is the version number on the disk label, if the version is A2 or something else, then enter that version number instead of A1.

After the program is loaded, type EXEC. This will display the title screen, and prompt you for a RELOCATION ADDRESS. Just press the (ENTER) key, and you will be greeted with the copyright notice and WORKSAVER PLUS prompt, >>, which replaces the familiar 'OK' promt of the Color Computer.

32K DISK COMPUTER

The 32K owner can run the PLUS from disk by entering;

```
RUN"P32
```

P32 is a BASIC Progam that performs three functions;

1. Loads a non-relocatable version of the PLUS at the uppermost location in the lower 32K of RAM.

2. Executes a PCLEAR0 to open up space for a non graphics program. (If you need graphic pages see IMPORTANT NOTE BELOW)

3. Loads in the BASIC PROGRAM "TABLE", which does not contain any BASIC Program Lines, but it does contain the Single Key ENTRY words given on the reference card.

64K DISK COMPUTER

The 64K owner can run the PLUS from disk by entering;

```
RUN"P64
```

P64 is a BASIC PROGRAM that performs the same three functions as P32, plus it boots up the 64K computer by transferring ROM to RAM.

IMPORTANT NOTE FOR *** ALL *** USERS

All versions of the WORKSAVER PLUS execute a PCLEAR0 when they are loaded. If you run a program that requires graphic screens, you will need to enter a PCLEARn before running the program. Where 'n' is the number of pages required for your program. Forgetting to do this will result in an

```
?FC ERROR
```

in whatever line contains the PMODE command. At which time you should enter the correct PCLEARn command and start the program over.

EXPLORING THE WORKSAVER PLUS

For those of you who want to get going right away, the full screen editor, single key entry, and program listing control can be covered quite rapidly, and you can start using the program without learning about the rest of the features. To introduce you to the basics of these features we have presented an overview below with a description of what to read in the manual to get started.

BEFORE YOU READ ABOUT THE EDITOR, YOU SHOULD READ THE 'USING THIS MANUAL' PARAGRAPH ON PAGE 2 OF THE WORKSAVER MANUAL.

THE EDITOR

This is a very powerfull, yet quite simple editor to use. The first thing you'll notice is that the (LEFT ARROW) key does not erase characters as you backup over them. This makes it easy to change a character in the middle of the line you are typing without having to retype everything to the right of that character. Furthermore, when you enter a line, the cursor can be anywhere on the line when you press the (ENTER) key and the whole line will be entered. Once a line is entered, you can use the (UP ARROW) key to move back up to that line, make a few changes to it, and quickly reenter it again. For more information on special cursor controls read the SCREEN CONTROL, CURSOR CONTROL, and EDIT CONTROL paragraphs on page 3 of the WORKSAVER MANUAL.

SINGLE KEY ENTRY

This is undoubtably the easiest feature to learn. First of all, the control keys are the (CLEAR) and the (BREAK) keys. The (CLEAR) key generates the words listed across the top of the keys, and the (BREAK) key generates the words listed down the left side of the keys. To print any of the words on the reference card, you need only press the appropriate control key (CLEAR) or (BREAK), release that key, and then press the appropriate keyboard key. For example;

(CLEAR) , (L) will print the word LIST on the screen.

The keys are completely redefinable, but learning how can wait until later.

IMPORTANT NOTE: Before you use the WORKSAVER PLUS to edit a program that was lying around when you recieved the WORKSAVER PLUS, you will need to read the paragraph,
'LOADING A PROGRAM THAT DOES NOT HAVE A KEY DEFINE TABLE',
on page 7 of the WORKSAVER MANUAL. Programs entered without the WORKSAVER or WORKSAVER PLUS will not have the key define table, and that paragraph explains how to load these programs.

PROGRAM LIST CONTROL

As you enter more and more lines of a program, you will eventually have more lines than you can display on the screen. To view lines that are off the top of the screen, press

(SHIFT) + (DOWN ARROW),

and the next lowest line number in your program will be listed at the top of the screen. This may cause the screen to scroll down. To list lines which are off the bottom of the screen, press

(SHIFT) + (UP ARROW),

and the next highest line in your program will be listed at the bottom of the screen. This may cause the screen to scroll up. There is a more complete discription of the List Control on pages 6 and 7 of this manual, but this is enough to use them effectively.

? xx ERROR ... what the ???

LIST ON ERROR

Before you get surprised and wonder what's happenning, there is one feature of the WORKSAVER PLUS that you may wish to at least be aware of, LIST ON ERROR. This is a very usefull feature that will automatically relist BASIC lines after BASIC has encountered an error when running your program. Furthermore, the cursor will be placed at or near the error in your line. The full screen editor will then make it easy to make a quick correction at that point.

To try this out, enter the following program line and run it.

```
    10 A=(12+(34-2)
```

When the line is run the screen will be as shown below;

```
|- - - - - - - - ---- - - - - - - - -|
|10 A=(12+(34+2)                     |
|RUN                                 |
|? SN ERROR IN 10                    |
|>>                                  |
|10 A=(12+(34+2)                     |
|                                    |
|- - - - - - - - ---- - - - - - - - -|
```

The Syntax Error is because of unmatched parenthesis in line 0. As you can see, the PLUS listed line 10 automatically after the error was encountered. The cursor will be flashing immediately at the end of the line where another ')' is needed. Now all you have do is type a ')' and press the enter key to make the correction.

If you don't want to edit the line that is listed, you can just press the Enter Key and the cursor will be moved down to the next line.

Another aspect of List on Error deals with errors encountered from commands typed from the keyboard, such as CLOAF"AEDIT. After an error like this, the PLUS will allow BASIC to print its normal error message, but then the PLUS will automatically move the cursor back up the screen and place it over the F in CLOAF. The F is the point where BASIC first noticed the problem and stopped trying to interpret the rest of the line.

In the two examples given the cursor was placed at the exact location where the error could be corrected. This will not always happen, and in fact in the first example, line 10 would be more correct if the first '(' was deleted instead of a second ')' being added to the end. The PLUS always places the cursor at the point where BASIC becomes confused when interpretting a line, and you will have to decide if the error is at the cursor or just nearby somewhere. At any rate, the location of the cursor will generally be a sufficient clue for finding the real problem.

PROGRAM LISTING CONTROL INTRODUCTION

Scrolling a program listing up or down the screen is a convenience that has until now been reserved to word processor text. With this feature, the limitation of the 16 lines of screen text virtually dissappears. Controlling the listing is done by simply using the (SHIFT)+(UP ARROW) and (SHIFT)+(DOWN ARROW). In this section we will present the basics for using the scroll control. In addition to the basics, there are two additional features of the scroll control and these are presented after the basics.

The first step is to have a program loaded, and for the sake of consistency, we will use the AEDIT program in all of our examples. For cassette users this program is located at the beginning of the flip side of the tape, and it is loaded by the CLOAD command. Disk users will load it using 'LOAD "AEDIT".

After the program is loaded, clear the screen [(SHIFT)+(CLEAR)], and follow the steps below.

1> With the cursor at the upper left-hand corner of the screen, list line 10 the conventional way using LIST10 [(CLEAR),(L),(10)].

2> Press (SHIFT)+(UP ARROW) once, and try not to hold the keys down at this point or they will repeat. The screen should appear as:

```
|-----------TOP BORDER-----------|
|LIST10                          |
|10 CLS:DIM LT$(30):MX=30        |
|12 DATA EGGS,BACON,BREAD,STEAK,S|
|PAGETTI,CHICKEN,MILK,ORANGE JUIC|
|E,LETTUCE,PAPER TOWLS,LAUNDRY DE|
|TERGENT,APPLES,POTATOES,SOUP,BRO|
|CCOLI,ICE CREAM,BUTTER,FISH     |
|                                |
- - - - - - - - - - - - - - - - -
```

Thus (SHIFT)+(UP ARROW) means list the next highest line number.

3> Press (SHIFT)+(DOWN ARROW) once, and try not to hold the keys down at this point or they will repeat. The screen should appear as shown below.

```
|-----------TOP BORDER-----------|
|8 X=(PEEK(&H74)*256+PEEK(&H75))-|
|12:RETURN                       |
|10 CLS:DIM LT$(30):MX=30        |
|12 DATA EGGS,BACON,BREAD,STEAK,S|
|PAGETTI,CHICKEN,MILK,ORANGE JUIC|
|E,LETTUCE,PAPER TOWLS,LAUNDRY DE|
|TERGENT,APPLES,POTATOES,SOUP,BRO|
|CCOLI,ICE CREAM,BUTTER,FISH     |
|                                |
- - - - - - - - - - - - - - - - -
```

Thus (SHIFT)+(DOWN ARROW) means list the next lowest line number, and in this case you should have noticed that lines 10 and 12 shift down the screen in the process. Furthermore, the 'LIST 10' was written over by the listing of line 8.

4) Now you are ready to scroll the listing up and down the screen at will. To do this simply press and hold down the (SHIFT)+(UP ARROW) until line 12 has been scrolled off the top of the screen, and then try holding down (SHIFT)+(DOWN ARROW) until you reach the first line of the program and the listing stops. Then switch back and forth.

We hope you found that to be about as simple as moving the cursor about using the arrow keys. The rest of this section will now discuss the subtle ways that the scrolling operates.

ADDITIONAL FEATURES

Before explaining the two additional features we thought you may be interested in the rules that govern the scroll control operation. Despite the appearance, the PLUS controls the scrolling ONLY FROM THE INFORMATION it can read off the screen. That is to say, 'IT LOOKS AT THE SCREEN AND PRINTS THE NEXT OR PREVIOUS LINE ACCORDING TO WHAT IT SEES*.' It 'knows' when lines are only partially listed at the top or bottom of the screen. It can usually tell the difference between data on the screen and line numbers. It assumes that BASIC line numbers will always be located at the beginning of a logical screen line (see glossary). Furthermore, a BASIC line number will always be separated from the code for that line by just one space. That space must exist for the PLUS to interpret the number on the screen as a BASIC line number.

When (SHIFT)+(UP ARROW) is pressed to list the next line, the PLUS starts looking for a line number at the bottom left-hand corner of the screen. It then moves up the screen checking the beginning of each logical screen line for a BASIC LINE NUMBER. When it finds a line number on the screen (say line 50), it then finds the next line number in your program (say line 55), and prints that line below line 50 on the screen. As you have seen, this sometimes results in the screen being scrolled up.

THE LINE CLOSEST TO THE BOTTOM OF THE SCREEN DETERMINES THE NEXT LINE TO BE LISTED USING (SHIFT)+(UP ARROW).

OVERRIDING THE CLOSESTS TO THE BOTTOM SEARCH

When editing and debugging programs, there may be times when the line you are editing in the middle of the screen has a higher line number (see line 100 on next page) than the one at the bottom of the screen (line 60 on next page). In this case you will not be able to list line 101 using (SHIFT)+(UP ARROW) without first using the feature described next.

NOTE: *For the technically minded: The WORKSAVER PLUS uses memory &H3FF to &H5FF for the screen display, although only &H400 through &H5FF gets displayed.

```
- - - - - - - - - -- - - - - - - - -
|100 IF NX=MX AND X>MX THEN PRINT|
|@448,"*** array is full":GOTO54 |
|>>                              |
|                                |
|54 PRINT@((A-1)-(A=0)),"":LINEIN|
|PUT"";A$                        |
|60 X=VAL(LEFT$(A$,5))           |
---------BOTTOM BOARDER-----------
```

SELECTIVELY SCROLL LINES FROM BOTTOM

For this reason, we have included a feature to selectively delete the space after each line number from the bottom of the screen. As explained above, the PLUS will not read a line number unless there is a space after the number. Thus lines can be hidden from the 'eyes' of the PLUS by deleting the spaces after each line number below line 100. This could be done manually, but then it would be easier to just use the LIST command to list line 101. To make deleting spaces easy we added a function to selectively delete spaces after lines using (CLEAR),(SHIFT)+(UP ARROW). When a (CLEAR),(SHIFT)+(UP ARROW) is executed, the PLUS will look for the first BASIC line it can find by starting at the bottom left-hand corner and working its way up the screen. If the line it finds happens to be line 60 as shown above, then it will delete the space after the 60 as shown here;

```
- - - - - - - - - -- - - - - - - - -
|100 IF NX=MX AND X>MX THEN PRINT|
|@448,"*** array is full":GOTO54 |
|>>                              |
|                                |
|54 PRINT@((A-1)-(A=0)),"":LINEIN|
|PUT"";A$                        |
|60X=VAL(LEFT$(A$,5))            |
---------BOTTOM BOARDER-----------
```

After that space has been deleted, you can delete the space in the next line up (line 54) by just pressing the (UP ARROW) by itself, and each successive (UP ARROW) will then delete the space after the next line number up the screen. Pressing any other key including (SHIFT)+(UP ARROW) will end this feature. Following the example we started with, line 101 can be listed after the space has been deleted in lines 60 and 54. In fact a (SHIFT)+(UP) can be used to both end the space delete function as well as list line 101 below line 100.

```
- - - - - - - - - -- - - - - - - - -
|100 IF NX=MX AND X>MX THEN PRINT|
|@448,"*** array is full":GOTO54 |
|101 IFX>=NX THEN N=NX+1:NX=N:LT$|
|(N-1)=MID$(A$,5):D=0:GOTO113    |
|54PRINT@((A-1)-(A=0)),"":LINEIN |
|PUT"";A$                        |
|60X=VAL(LEFT$(A$,5))            |
---------BOTTOM BOARDER-----------
```

OVERRIDING THE CLOSESTS TO THE TOP SEARCH

When (SHIFT)+(DOWN ARROW) is pressed to list the previous line, the PLUS starts looking for a line number at the top left-hand corner of the screen. It then checks down the left-hand side of the screen for a line number. When it finds a line number, it then finds the previous line in your program and list it directly above the line it found. As you have seen, this sometimes results in the screen being scrolled down.

THE LINE CLOSEST TO THE TOP OF THE SCREEN DETERMINES THE LINE TO BE LISTED USING (SHIFT)+(DOWN ARROW)

As with (SHIFT)+(UP ARROW), this can create a limitation if line numbers toward the top of screen are greater than the line you are editing in the middle of the screen.

SELECTIVELY SCROLL LINES FROM THE TOP

For this reason we have included a feature to delete the space after line numbers starting at the top of the screen. This is executed by a (CLEAR),(SHIFT)+(DOWN ARROW). After pressing this once, the space after the line number closest to the top of the screen will be deleted. Pressing just the (DOWN ARROW) again will delete the space in the next line number down from the top, and each successive (DOWN ARROW) will work on the next line down from the top. Pressing any other key including (SHIFT)+(DOWN ARROW) will end this function.

RELIST LAST LINE ENTERED

There will be times when there are no lines on the screen. (SHIFT)+(UP ARROW) and (SHIFT)+(DOWN ARROW) will both do the same thing in this case. They will both relist the last line entered from the keyboard. For example assume you are debugging a program, and that you just made a change to line 20 and entered it. Now you decide to try out the change by running the program, but the first thing the program does is clear the screen and print a menu. If you suddenly realize that you should have also made a correction to line 21, then it is a simple 2 step proceedure to get line 21 listed:

1. BREAK the program, and
2. Press (SHIFT)+(UP ARROW) twice.

Lines 20 and 21 will then be listed and ready to edit.

GLOBAL SEARCH AND REPLACE INTRODUCTION

There are times when programs need major surgery, requiring repetitive deletes and changes. Such changes may be as simple as renaming all AN$ variables to all XN$ variables, or they may be as complex as adjusting all POKE addresses in a program to avoid conflict with a new utility you may have just bought (hint hint). Other uses include: converting cassette programs with INPUT#-1 commands to INPUT#1 commands for disk; deleting all spaces and remarks in a program to speed execution and conserve memory; search for a remark that locates a subroutine; or perform any repetetive search and/or replace you may need. The PLUS can handle all of these, and in this section we will present the various features of the SEARCH and REPLACE FUNCTION.

The first step is to have a program loaded, and for the sake of consistency, we will use the AEDIT program in all of our examples. For cassette users this program is located at the beginning of the flip side of the tape, and it is loaded by the CLOAD command. Disk users will load it using 'LOAD "AEDIT".

SEARCH/REPLACE BASIC COMMANDS, DATA, VARIABLES, & ARRAYS

We will start with an example to discuss the pertinent details of search and replace, and then followe it with a more complete discussion later. In our example we will rename the variable NX to NEXTX throughout the AEDIT program.

Please follow the steps given below.

1> Clear the screen (SHIFT)+(CLEAR)

2> Type (SHIFT)+(@): An inverted backslash (\) will be printed. This must be at the beginning of a logical screen line (see glossary), and it is always used to initiate the GLOBAL SEARCH FUNCTION.

3> Type .NX.NEXTX. immediately after the backslash (do not leave any spaces). The first (.) tells the PLUS to define the Search Item to be everything that is typed until it comes to the next (.). In this case NX is the Search Item. Then it checks for a third (.), and if one is found, it will define the Replace Item as everything that is typed between the second and third (.). In this case NEXTX is the Replace Item. As in this example, the lengths of the Search and Replace items need not be the same, and quite often they won't be.

The first character after the backslash (the '.' in our example) defines the separator character for the search and replace function. It could be any printable character except ('), (-), or (") which are special characters to be discussed later. Thus we could have defined the Search and Replace Items by entering /NX/NEXTX/ or :NX:NEXTX: or ;NX;NEXTX; or even ANXANEXTXA.

4> Press the enter key. Your screen will now appear as

```
|----------TOP BORDER------------|
| All This Skip @-delay BRK-quit |
|14 FOR N=1 TO 15: READ LT$(N):A=|
|N*32-32:GOSUB90:NEXT:NX=N:SX=N:N|
|=14:GOSUB 134:A=32:T=1:N=2:GOTO5|
|4                               |
|                                |
 - - - - - - - - - - - - - - - -
```

The cursor will be fixed over the (N) in (NX) and the PLUS will be waiting for your next choice from the menu at the top of the screen. These choices do the following;

(A) Pressing the (A) key will cause all NX's to be changed to NEXTX's through out the program. As the changes are being made, each line will be displayed below the menu. You can stop the PLUS at any point by pressing the BREAK key (HOLD IT DOWN UNTIL THE INTERRUPT OCCURS) when it is executing the ALL choice.

(T) Pressing the (T) key will change this occurrence of NX to NEXTX, and then the PLUS will search for the next NX in the program. When it finds one it will list the line and stop to wait for your choice.

(S) Pressing the (S) key will leave this occurrence of NX unchanged and then the PLUS will search for the next NX in the program. Again when it finds one, it will list the line and stop to wait for your choice.

(@) Pressing the (@) key will temporarily abort the search and replace function, and put you in the edit mode. You will then be able to make any corrections you wish to the current line. After you press the (ENTER) key and enter the line, the search and replace function will automatically resume looking for the next occurrence of NX in your program.

(BREAK) Pressing the (BREAK) key will end the search and replace function. When it is ended, you will be returned to the edit mode and the cursor will be flashing over the (N) in (NX).

5> Try out each of the 5 choices using different search and replace strings. When you are comfortable with how the search and replace choices work, then you will be ready to go onto the next phase.

SEARCHING A RANGE OF LINES

The Search and Replace function can be used to search only through a specified range of lines.

To start a search and replace, such as /NX/NEXTX/, at a given line, say 100, and continue to the end of a program, you would enter;

```
  \/NX/NEXT/100-
```

Thus the line number information follows the third separator character. The syntax for other line range selections is as follows;

\/NX/NEXTX/-100 do search and replace from beginning of program up to and including line 100.

\/NX/NEXTX/100-200 do search and replace from and including line 100 up to and including line 200.

\/NX/NEXTX/100- do search and replace from and including line 100 up to the end of the program.

SEARCHING WITHOUT A REPLACE

The search and replace function can also be used to just search for an item. The syntax for search only is;

[(SHIFT)+(@)] [SEPARATOR CHARACTER] [SEARCH CHARACTERS] [SEPARATOR CHARACTER] [LINE RANGE if any]

For example, the following could be used to search for NX

```
  \/NX/
  \/NX/-100
  \/NX/100-200
  \/NX/100-
```

When using search only, the 'All' and 'This' choices at the top of the screen have no meaning.

SEARCH INSIDE PRINT STRINGS

Up to now all searches have excluded looking inside quote marks or remarks. For example if there were a line such as

```
  30 ?"NX =";NX
```

and we had executed the \/NX/NEXTX/, and then selected All, line 30 would have been changed to

```
  30 ?"NX =";NEXTX.
```

The (NX) inside the quotes would have been skipped. Search and Replace can be made to look only inside quotes and nowhere else using the following syntax;

[(SHIFT)+(@)] ["] [SEPARATOR CHARACTER] [SEARCH CHARACTERS] [SEPARATOR CHARACTER] [REPLACE CHARACTERS and SEPARATOR CHARACTER if required] [LINE RANGE if any]

For example, the (NX) in line 30 PRINT "NX =";NEXTX could be changed to (NEXTX) by

```
   \"/NX/NEXTX/
```

NOTE: This can be used to search and replace inside any pair of quotes such as

```
   A$="INSIDE STRING ASSIGNMENTS"
   INPUT"INPUT PROMPTS";A$
```

SEARCH INSIDE REMARKS

Using a similar syntax to search inside quotes, you can also have the PLUS perform a search and replace on remarks only. This is most useful for finding a subroutine that has a title inside a remark. The syntax for search and replace inside remarks is

[(SHIFT)+(@)] [' or REM] [SEPARATOR CHARACTER] [SEARCH CHARACTERS] [SEPARATOR CHARACTER] [REPLACE CHARACTERS and SEPARATOR CHARACTER if required] [LINE RANGE if any]

For example,

```
   \'.SORT STRING.
```

or

```
   \REM.SORT STRING.
```

would search your program for a (SORT STRING) remark.

SEARCH AND DELETE

The search and replace can be used to search for something and replace it with nothing. The syntax for search and delete is;
[(SHIFT)+(@)] [SEPARATOR CHARACTER] [SEARCH CHARACTERS] [SEPARATOR CHARACTER] [SEPARATOR CHARACTER] [LINE RANGE if any]
For example, the following could be used to search and delete spaces in a program;

```
   \. ..
```

WARNING: Before you use this to remove unneccessary spaces in a program you should be aware of a small pitfall. BASIC does not require any spaces inside a program to run it. Unfortunately, as you probably know, you must include certain spaces when entering a program or you will get a syntax error when the line is run. This happens because BASIC tokenizes lines before it stores them in memory. For example, BASIC does not store a command such as LINEINPUT using 9 characters. Instead it reduces it to a shorthand version of only 2 characters. Consequently BASIC must read and rewrite each BASIC LINE you enter before it stores it in memory. For example, BASIC will read the following two versions of the same line differently because of the lack of a space in the second version

```
   10 IFA=B GOTO20      entering this line is OK
   10 IFA=BGOTO20       entering this line won't work
```

After the first version has been correctly entered, executing \. .. will remove the space, and the line will still work properly. If you now list line 10, however, it will look identical to the second version above. To edit this line now, you will have to put back the space before entering it. You can then use \. .. to remove the space again after the line has been entered.

It is a good idea to keep a copy of the original program before you delete spaces out of it and save it. Then if you need to make some major changes, you can use the original and avoid having to keep putting the necessary spaces back into each line you edit.

SEARCH AND DELETE REMARKS

Deleting remarks is not automatic, but it is quite easy as we will demonstrate. The syntax for deleting remarks is

```
   \.'.        NOTE: USE .'. SYNTAX ONLY. /'/ WILL NOT WORK!!
```

and

```
   \.REM.      NOTE: USE .REM. SYNTAX ONLY. /REM/ WILL NOT WORK
```

As you can see, you delete the (') type remarks and then go back and delete the (REM) type remarks.
When either is executed, the PLUS will search and find the remark symbol itself. To delete the remark you will have to

1> Choose the Delay option by pressing the (@) key option
2> Execute an EEOL (by (CLEAR),(RIGHT ARROW)) to erase the remark
3> Then finally press (ENTER) to restart the search for the next remark.
4> Repeat steps 1 thru 4 until all remarks have been found and deleted.

THINGS THAT DON'T WORK

You can not search for a combination of commands, quotes, and/or remarks. For example assume you have the following line in a program.

```
    10 PRINT #-2,"1)·NAME IS";N
```

If you defined a search and replace such as

```
    \PRINT #-2,"1\PRINT"1\
```

it will not work because the quote in the search string comes after a tokenized statement namely PRINT #-2, in this case. The result of trying this will be that the PLUS will not find a match in the program. The above search could be accomplished by

```
    \#-2,\\
```

which is easier anyway.

LINE RENUMBERING INTRODUCTION

The Full Screen Editor will allow you to quickly and easily change line numbers, but it does not update the GOTO, GOSUB, THEN, and ELSE line references throughout your program. With the PLUS, you can move a line or a group of lines from one location to another in your program, and all line references are automatically updated. As with most WORKSAVER PLUS features, there are a few variations to line renumbering, and each one is described in detail.

The first step is to have a program loaded, and for the sake of consistency, we will use the AEDIT program in all of our examples. For cassette users, this program is located at the beginning of the flip side of the tape, and it is loaded by the CLOAD command. Disk users will load it using 'LOAD "AEDIT".

RELOCATING A SINGLE LINE

When AEDIT was first written, there was a line 50 and a line 52, but they were deleted. Line 54 remains and we had always wanted it to be renumbered to line 50 so in this example of moving a single line, we will renumber line 54 to be line 50. The syntax for moving a single line is quite simple;

[OLD NUMBER] [;] [NEW NUMBER].

The only other requirement is that it be placed at the beginning of a logical screen line (see glossary). Thus we renumber line 54 to be line 50 by;

54;50 and then press the (ENTER) key.

The renumbering process takes time so don't be alarmed by the pause. Also the PLUS uses part of Extended BASIC'S Renumbering program to update the line references throughout the AEDIT program. Consequently, renumbering may produce a warning message of the type:

```
UL xxxx IN yyyy
```

which is read as Undefined Line xxxx in line yyyy.

RELOCATING A GROUP OF LINES

A group of lines can be renumbered by either of two methods. First of all, the lines can all be moved by the same fixed amount. That is to say, if a group of lines are numbered 100, 101, 102, 103, 110, 111, and 112, they can be renumbered to be lines 400, 401, 402, 403, 410, 411, and 412 by adding 300 to each line number. The other method for moving a group of lines will move them and resequence them at the same time. Using the same group of lines as before, they can be renumbered to be lines 400, 405, 410, 415, 420, 425, 430, and 435 respectively.

The syntax for the first method is:

[FIRST LINE NUMBER in the group] [-] [LAST LINE NUMBER in the group] [;] [NEW NUMBER for the first line number].

For example:

```
   100-112;400
```

would renumber lines 100, 101, 102, 103, 104, 110, 111, 112 to be 400, 401, 402, 403, 404, 410, 411, 412 respectively.

The syntax for the second method is:

[FIRST LINE NUMBER in the group] [,] [LAST LINE NUMBER in the group] [;] [NEW NUMBER for the first line number] [,] [INCREMENT to be used between lines]

For example:

```
   100-112;400,5
```

would renumber lines 100, 101, 102, 103, 104, 110, 111, 112 to be 400, 405, 410, 415, 420, 425, 430, 435 respectively.

ERROR MESSAGES

The renumbering process may report any of three types of errors.

1. ?UL ERROR:
   The target lines (i.e., lines to be moved) must exist, if they do not exist the error reported will be an Undefined Line Error of the form ?UL ERROR. In the examples above, if either line 100 or line 112 was not a line in the BASIC program, then an Undefined Line Error would have been reported.

2. ?FC ERROR:
   A second possible error is an Function Call Error of the form ?FC ERROR. This will occur whenever the destination of the single or block of lines would overwrite or surround existing lines. For example in the relocation of lines 100 through 112 given above, if there had been a line 422, the first method described above would work OK, but the second method would generate an FC error. Line 422 would be surrounded by lines 420 and 425, and this is not allowed.

   NOTE: This rule does not apply to the line numbers involved in the move. For example lines 130,131,132,133,134 of the AEDIT program can be renumbered to be 130,132,134,136,138 by
   ```
       130-134;130,2
   ```
   without generating an FC Error.

3. ?UL xxxx IN yyyy:
   As mentioned earlier, the PLUS uses part of Extended BASIC'S Renumbering routine to update the line references through out the AEDIT program. Consequently, renumbering may produce a warning message of the type:
   ```
       UL xxxx IN yyyy
   ```
   which is read as Undefined Line xxxx in line yyyy.

SCREEN SCROLL CONTROL INTRODUCTION

Everyone who has ever listed a program, or a long disk directory or a long listing of data, has experienced the frustration of not being able to get the screen to pause. The (SHIFT)+(@) is very difficult and inconvenient to use. With the PLUS however, you can have the screen pause for each page (16 lines) or each line or any number of lines you wish.

To gain control of the scroll you:

1> Press (BREAK) , (UP ARROW) and the (#) character will be printed on the screen.
2> Next enter the number of lines you wish printed to the screen before a pause is to take place.
3> Press the (ENTER) key.

After the scroll control has been set, it can be changed by repeating the above proceedure and entering a new number. If you wish to turn off the scroll control, you need only press (BREAK) , (DOWN ARROW).

Although this feature is called scroll control, the PLUS actually counts the number of screen lines (see glosary) printed. For example, if you have a clear screen, and you entered 8 for step 2 above, then after entering the command LIST at the top of the screen, the PLUS would pause the listing after eight screen lines were written on.

When the screen pauses, an inverted S will be displayed at the lower right corner of the screen. You can then press any key except (BREAK), (CLEAR), or (DOWN ARROW) to continue the listing until the next pause.

Pressing the (BREAK) key will terminate the listing, and put you back into the normal keyboard edit mode. If the listing happens to be data printed while running a BASIC program, then pressing the (BREAK) key will terminate the listing and execute a normal interrupt of the BASIC program.

To terminate a listing and not generate a normal (BREAK) interrupt to a BASIC program, you can press the (CLEAR) key. After pressing the (CLEAR) key, the PLUS will freeze the screen, and invisibly print the rest of your listing. In other words, the PLUS allows BASIC to continue sending characters to be printed on the screen, but the PLUS will intercept the characters and keep them from being printed to the screen. Consequently, it may take some time before BASIC completes its invisible listing, but in the case of running a BASIC program, at least the program execution will not be disrupted.

Pressing the (DOWN ARROW) while the screen is stopped with an inverted S in the lower right corner will cancel the scroll control. The listing will then continue uninterrupted, and scroll control will be turned off until you execute a (BREAK),(UP ARROW) as described above.

One final note, the scroll control only works on continuous printing to the screen. That is, whenever the cursor is flashing, the PLUS resets its count of lines-since-last-pause to zero. Thus the screen will never pause when you are just keying in program lines or data.

PRINTER ECHO INTRODUCTION

Many computers and operating systems provide the capability to send characters to both the screen and the printer. Now the PLUS makes this possible two different ways. First of all, you can echo to the printer every line entered from the keyboard or printed to the screen. This is handy for quick disk directory listings or dual screen/printer BASIC program listings. A second option allows you to move the cursor to any logical line (see glossary) and dump it to the printer. This can even be done from the input mode while running a program.

SCREEN PRINT ECHO

To have everything printed on the screen echoed to the printer, you need only turn on the printer echo by,

(CLEAR) , (SHIFT)+(=).

Once the printer echo is on, everything entered from the keyboard or listed to the screen will also be sent to the printer. If you are entering program lines or data from the keyboard, the printer will only print the line that exist when the (ENTER) key is pressed. Therefore, if you are editing a line, and you make multiple corrections to it, the printer will only list the line that exist when you finish editing and press the (ENTER) key. You can make a before and after copy of the line by listing the line [which will automatically send a copy to the printer], and then editing and entering the line.

LOGICAL SCREEN LINE PRINTER DUMP

There are times, when you may want a line listed to the printer without having to (ENTER) it. This is quite common when running a BASIC program where every press of the (ENTER) key will be in response to an INPUT Statement, and placing the cursor at any old place on the screen and pressing (ENTER) would not be a valid response. To selectively print lines of the screen to the printer without disturbing the execution of a BASIC program, you need only,

1. Position the cursor anywhere on the logical line of interest

2. Press (CLEAR) , (SHIFT)+(RIGHT ARROW) and the line will be sent to the printer.

EDIT MARKER

The Edit Marker is graphics character 140 which is solid black on the lower half and green on the upper half. This block is used to mark BASIC LINES on the screen whenever a change is made to the line. The block is placed in the space after the BASIC LINE NUMBER. The WORKSAVER always interprets these markers as spaces.

END OF LINE MARKER

The end of line marker is graphic character 143 which is a solid green block. Under typical conditions these blocks will exist only on the rightside of the screen, and they define the end of each logical (screen) line.

LOGICAL (SCREEN) LINE

The logical (screen) line is the information on the screen which will be sent to BASIC when the enter key is pressed. It is defined by END OF LINE MARKERS, and its length can actually take up to eight screen lines. For example the BASIC Line

```
116 X=A:N=N-1:FOR A=N TO 416STEP32:N=N+1:IF N=NX+1 THEN N=1:SX=NX
```

would be one logical line, and it would require 3 screen lines to list.

```
- - - - - - - - -- - - - - - - - - -
|116 X=A:N=N-1:FOR A=N TO 416STEP|
|32:N=N+1:IF N=NX+1 THEN N=1:SX=N|
|X                               |
|                                |
- - - - - - - - -- - - - - - - - - -
```

SCREEN LINE

A screen line is a physical line of the video display. There are 16 screen lines on the color computer and each is 32 characters wide.

APPENDIX: DISK LOADERS FOR WORKSAVER PLUS

The disk version and the cassette version of the WORKSAVER PLUS are the same program with the same features. The disk however comes with two special loader programs that make it more convenient to use the WORKSAVER with a disk system. If you have a 64K computer then you can get the WORKSAVER PLUS up and running by simply entering 'RUN P64'. Likewise for a 32K computer, 'RUN P32' will load and execute the WORKSAVER PLUS.

Both the 32K and 64K versions are not relocatable, but there is a copy of the WORKSAVER PLUS on the disk which is. This program is named WPLUS-A1/BIN, and it is loaded by 'LOADM WRKSV-A1'(do not use an offset). After entering 'EXEC' you will be prompted for a relocation address. This address is the address where the program will start. Once the program is relocated by WPLUS-A1, WPLUS-A1 will make the relocated version non relocatable (WPLUS-A1 must be used to relocate the WORKSAVER PLUS). Furtermore, if you use WPLUS-A1 to relocate the WORKSAVER, we leave it up to you to reserve the memory locations (that you locate the it at) SO THAT NO BYTES OF THE WORKSAVER ARE WRITTEN OVER BY ANYTHING ELSE YOU DO WHEN USING THE PROGRAM.

Once you have relocated the WORKSAVER PLUS you can make a loader program like P64 or P32 by making the appropriate changes to those programs. The copy of P32 points out the neccessary changes to make your own loader.

A note on the AEDIT program will not run as is with the W64 version of the WORKSAVER. To use AEDIT with that version replace the computed value X+13 in line 6 (only in line 6) with &HE001 such that line 6 reads;

```
6 A$=RIGHT$("0000"+HEX$(&HE001),4) or
6 A$="&HE001"
```

One final note the BASIC loaders can be used to perform other system setups such as printer baud rates, default drives, disk stepping rates, etc. We included comments in the loader programs showing some of the potentials.

WPLUS-64 LOADER FOR 64K DISK COLOR COMPUTER

```
1 ' P64
      COPYRIGHT 1983
       BY PLATINUM SOFTWARE INC.

2 '     << WARNING >>
     do not attempt to attach
     worksaver key define table
     to this program.  Doing so
     will destroy the boot pro-
     gram.

3 'THIS IS A WORKSAVER LOADER
     FOR A COPY OF THE WORKSAVER
     THAT WAS LOCATED AT &HE000
     AND THEN TURNED OFF & SAVED.

4 'THIS PROGRAM CAN BE RUN FROM
     DISK BY 'RUN"P64"'.  WHEN IT
     IS RUN, IT WILL BOOT UP 64K,
     LOADM THE WORKSAVER, TURN IT
     ON, AND ALLOW OTHER PROGRAMS
     TO BE LOADED AS WELL.

5 PCLEAR1:
  CLEAR 500,&H3000

10 A=PEEK(&H1B)*256+PEEK(&H1C)

12 DEFUSR0=A-5:
   A=USR0(0)

15 CLEAR 1000,&H7FFF

20 'ENABLE USE OF RESET BUTTON
        WITH 64K

21 POKE &HA055,&H0B:
   POKE &H72,&H03:
   POKE&H73,&HF8:
   DATA 12,B7,FF,DF,7E,C0:
   FOR I=0 TO 5:
   READ A$:
   POKE &H3F8+I,VAL("&H"+A$):
   NEXT:
   A=&HD4-19*(PEEK(&HC153)=&H31)
     :
   POKE &H3F8+I,A

25 'SET TRACK TO TRACK SPEED OF
        DISK TO 6 MSEC. DELETE LINE
        26 IF YOUR DISK CAN NOT RUN
        AT THIS SPEED.
```

```
26 POKE&HD723,&H1C:
   POKE&HD6CD,0

30 PRINT"BASIC IS NOW IN RAM":
    ' TO USE BOOT PROGRAM WITHOUT
        WORKSAVER, DELETE THE FOLLOWIN
      G LINES, i.e.,DEL 100-

100 CLS:
    PRINT":
    < W O R K S A V E R ++ P L U S >
                VERSION A-1
            COPYRIGHT (C) 1983 BY
            PLATINUM SOFTWARE INC.
                 P.O. BOX 833
          PLATTSBURGH, N.Y.   12901
                518 643 2650"

120 POKE&HE00,0:
    POKE&HE01,0:
    POKE25,14:
    POKE26,2:
     '  PCLEAR 0 FOR DISK

150 LOADM"WPLUS-64":
    DEFUSR0=57348:
    A=USR0(A)
       :
     '  USR0 WILL NOW TURN WORKSAVER
          ON AND OFF

160 'INSERT OTHER LOADM'S HERE.
      TURN WORKSAVER OFF BY
       'A=USR0(0)',
      EXEC NEWLY LOADED PROGRAM, AND
      THEN TURN WORKSAVER BACK ON BY
      'A=USR0(0)', E.G:165 A=USR0(0):
       LOADM"SPOOLER":   EXEC:   A=US
      R0(0)

170 LOAD"TABLE"                 :
     ' LOADS KEY DEFINE TABLE.  THIS
      CAN BE CHANGED TO BE ANY BASIC
      PROGRAM.

171 '<<NOTE>>
      LOADING THE KEY DEFINE TABLE AS
      SHOWN ABOVE IS  NOT THE SAME AS
      'ATTACHING' THE KEY DEFINE TABLE
      TO THIS PROGRAM, AND THEREFORE
      EXECUTING LINE 170 WILL NOT
      DESTROY THE BOOT PROGRAM.
```

WPLUS-32 LOADER FOR 32K COLOR COMPUTER

```
 1 'P32
     COPYRIGHT 1983
     PLATINUM SOFTWARE INC.

 2 'THIS IS A WORKSAVER LOADER
    FOR A COPY OF THE WORKSAVER THAT
    WAS LOCATED AT THE TOP OF 32K
    AND THEN TURNED OFF BY
      EXEC 28880
    AND SAVED BY
      SAVEM"WPLUS-32",28876,32767,0

 3 'ENTER - RUN"P32 - AND THE
       WORKSAVER WILL BE LOADED
       AND EXECUTED

 100 CLS:
     PRINT":
     < W O R K S A V E R ++ P L U S >
                  VERSION A-1
            COPYRIGHT (C) 1983 BY
            PLATINUM SOFTWARE INC.
                  P.O. BOX 833
          PLATTSBURGH, N.Y.   12901
                  518 643 2650"

 110 CLEAR 1000,28877

 120 POKE&HE00,0:
     POKE&HE01,0:
     POKE25,14:
     POKE26,2:
     '  PCLEAR 0 FOR DISK

 140 POKE &H74,&H70:
     POKE &H75,&HCC
                         :
     'SET TOP OF MEMORY TO BE START
       OF WORKSAVER, E.G. &H70CC=28876
        (THIS CAN BE LEFT OUT IF YOU
          ARE MAKING YOUR OWN LOADER)

 150 LOADM"WPLUS-32":
     DEFUSR0=28880:
     A=USR0(A)

 160 LOAD"TABLE"
```

Addresses which are underlined in the above listing are subject to change without notice. Check the P32 program on your disk for correct addresses.

```
WORKSAVER PLUS DISK -- VERSION A1 (C)1983

NAME    EXT : <GRANS USED> : <SECTORS IN LST GRAN> : <BYTES IN LST SECTOR>
------------------------------------------------------------------------------
WPLUS-A1BIN    :  20  21  22:  5:  7
P64     BAS    :  23:  9: 5E
WPLUS-64BIN    :  1E  1F:  7: 9E
P32     BAS    :  24:  6: F8
WPLUS-32BIN    :  25  26:  7: 9E
TABLE   BAS    :  1C:  2: 2A

BINARY FILES:STRT END  EXEC
----------------------------
WPLUS-A1BIN  0E00 23FC 1100
WPLUS-64BIN  E000 EF93 E004
WPLUS-32BIN  706C 7FFF 7070

FREE GRANULES= 58


   FILE ALLOCATION TABLE ** TRACK 17 ** SECTOR 2 ** BYTES &H00 THRU &H43

        00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
   --------------------------------------------------------------------------
   00:  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF
   10:  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  C2  FF  1F  C7
   20:  21  22  C5  C9  C6  26  C7  FF  FF  FF  FF  FF  FF  FF  FF  FF
   30:  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF  FF
   40:  FF  FF  FF  FF
```

Addresses which are given in the BINARY FILE LISTING are subject to change without notice. Use DISK WORKSAVER PACKAGE or equivalent to verify these addresses.

PLATINUM SOFTWARE WORKSAVER
ADDENDUM VERSION A4

```
<<<<<  BEFORE LOADING A PROGRAM SAVED BEFORE YOU RECIEVED >>>>>
<<<<<  THE WORKSAVER, WE SUGGEST THAT YOU READ THE MANUAL  >>>>>
<<<<<  PAGES 6 & 7 AND THE KEY TABLE INFORMATION GIVEN IN  >>>>>
<<<<<  THIS ADDENDUM                                       >>>>>
```

PROGRAM MEMORY REQUIREMENT:

Numerous program improvements and additions have increased the memory requirements from less than 2K to 2137 bytes. Except for adding just 4 bytes to fix a bug in the use of AUTORUN keys with DYNAMIC INPUT, these changes have resulted from customer request for improvements. We hope that you feel they are worth your memory.

MANUAL CORRECTIONS:

PAGE 9: Change the ',' in
```
10 INPUT "DYNAMIC INPUT",A$
```
to a ';'.

PAGE 12: Under AUTOMATIC LINE NUMBERING the directions state; To invoke the auto line numbering mode, press
(CLEAR) , (0)
but in the manual the (CLEAR) , (0) is incorrectly spread out across the page.

PAGE 14: Version A4 deletes the EOLC slightly differently than the discussion at the top of page 14. When the EOLC is deleted, the 1 will not be pulled around to the cursor as shown on page 14. With version A4 (SHIFT)+(LEFT ARROW) deletes only the EOLC and does not effect any of the characters to the right of the EOLC. You can observe the deletion of the EOLC by the fact that the cursor switches from blinking white to blinking black.

PAGE 15: The third sentence at the top of the page should read as follows: To do this type (CLEAR) , (UP ARROW) and all the end of line positions currently defined are represented by A SOLID BLACK GRAPHIC CHARACTER (HEX 80).

PAGE 18: The last paragraph should read: We will now demonstrate how the WORKSAVER can be used to increase string space to 300 bytes without loosing any data. As discussed in the C.MEM SECTION (pg 8) string space can only be expanded 100 bytes at a time. Currently there are 100 bytes reserved, so to expand it to 300 bytes, we must first increase the string space to 200 bytes by (CLEAR) , (B) , (200). Now string space can be expanded to 300 bytes by (CLEAR) , (B) , (300), and the A$ array will not be lost. You can demonstrate this by ?A$.

PAGE 19: The program on page 19 is also used to demonstrate program chaining as discussed on page 20. Program chaining uses the DYNAMIC EDITING feature of the WORKSAVER. To conform with the requirements of DYNAMIC EDITING (see pages 15-17), string data which is to be passed on from one program to the next CAN NOT have strings which are located inside BASIC data statements.

The program on page 19, therefore, needs line 15 changed to the following:

```
15 READ A$(N):A$(N)=A$(N)+""
```

This will move the DATA STRINGS to the CLEAR STRING SPACE.

RELOCATING THE WORKSAVER

The WORKSAVER allows the user to locate it virtually anywhere in memory that the user would like it to be. It has a default location as described in the manual. When the default is used, the WORKSAVER automatically executes a CLEAR 100,n' where the 'n' signifies the location of the bottom of the WORKSAVER. If you do not use the default option, you need to execute a CLEAR command before you load and execute the WORKSAVER.

For example, assume you have a 32K computer and you want to locate the WORKSAVER at decimal address 28000. Then you should

1) Executed a CLEAR 100,28000
2) Load and execute the WORKSAVER
3) Enter 28000 into the relocation address.

IMPORTANT INFORMATION ABOUT THE KEY TABLE:

The A4 version provides a new feature for loading the KEY DEFINE TABLE. Before we present that feature, we would like to explain a little more about what the table is and where it is.

The table is the tokenized (one or two byte 'words' that represent a BASIC command) table that the WORKSAVER 'looks up' to determine which command or group of commands are to be printed for a given (CONTROL KEY) (KEY) combination. For example, pressing (CLEAR) , (L) will print the command 'LIST'. To print LIST the WORKSAVER had to go find the location in the table for (CLEAR) , (L), and when it found that location, it found the byte &H94 which BASIC translates into 'LIST'.

This is an easy consept for most of our users, however, we have caused some confusion by attaching the table to BASIC PROGRAMS. For the most part, this confusion has come about for two reason. First of all, there is the normal curiosity of how it is done, and secondly, there is the real problem of loading a BASIC PROGRAM that does not have a key define table. The manual explains how to protect the resident table before a 'non-table' BASIC PROGRAM is loaded (see page 7), but if you forgot to use this proceedure with earlier versions of the WORKSAVER, you had to either do without the table or start over. Both of these alternatives are user hostile, which has promted us to release version A4.

For the technically curious we will explain how the table is attached to a BASIC PROGRAM. The illustration below will explain most of how it is done.

```
---------------------------------------------------------------
- START OF BASIC PROGRAM          SAME FOR WORKSAVER          -
-   POINTED AT BY THE TWO       -------------------------------
-   BYTES AT &H19,&H1A        <  THE KEY DEFINE TABLE        >
------------------------------<  RESIDES BETWEEN THE END     >
- END OF BASIC PROGRAM        <  OF BASIC AND BEGINNING      >
-   NOT POINTED TO BY ANY     <  OF SIMPLE VARIABLES         >
-   BYTES. THE END IS MARKED  <    ------------------        >
-   BY TWO ZERO BYTES         <  IT IS ISOLATED FROM         >
------------------------------<  BASIC BY TWO ZERO BYTES     >
- START OF SIMPLE VARIABLES   <  WHICH DEFINE THE END OF     >
-   POINTED AT BY THE TWO     <  ANY BASIC PROGRAM           >
-   BYTES AT &H1B,&H1C        -------------------------------
-                                 SAME FOR WORKSAVER          -
---------------------------------------------------------------
```

When a BASIC PROGRAM is saved in tokenized form, the CSAVE or SAVE command assumes that the BASIC PROGRAM ends at the start of simple variables. Thus both the program and the key define table get saved when a CSAVE or SAVE is done. Furthermore, the CLOAD and LOAD commands set the byte at &H1B according to the length of the 'BASIC PROGRAM' that is loaded. Thus paying no attention to the real end of the BASIC PROGRAM.

When a BASIC PROGRAM is saved in the ASCII format, the CSAVE and SAVE commands determine the end of the progam by looking for the two zero bytes which really do define the end of the program. Thus the key define table is not saved when a program is saved in ASCII format.

LOADING A KEY DEFINE TABLE AFTER A 'NON-TABLE PROGRAM' HAS BEEN LOADED:

When the WORKSAVER is loaded and executed, it clears out any existing BASIC PROGRAM and loacates the key define table between the 'NULL BASIC PROGRAM' and the start of simple variables as discussed above. Since there are no real lines of any BASIC PROGRAM, the key define table is the only non zero bytes between the start of BASIC (&H19) and the start of simple variables(&H1B).

The table can be partially or totally redefined at this point. In this way you can customize it to any form you like, and then it can be saved using a CSAVE or SAVE command such as CSAVE "TABLE". YOU DO NOT HAVE TO HAVE A BASIC PROGRAM PRESENT TO SAVE THE SOLITARY KEY DEFINE TABLE.

Now we will assume you have saved the solitary key define table and named the file 'TABLE'. If you

- Load a BASIC PROGRAM that does not have a key define table (any program entered and saved without using the WORKSAVER will not have a key define table.),
- and you forgot to protect the table by (CLEAR) , (SHIFT)+(>) (see manual page 7),
- then you can load your 'TABLE' program by
  1) (CLEAR) , (SHIFT)+(<)
  2) Entering CLOAD "TABLE (or LOAD"TABLE for disk)
  3) Pressing the enter key.

This sequence performs the well known 'magazine merge proceedure'. We call it that because there have been numerous articles on how to merge programs by manipulating the pointers at &H19 and &H1B. A combination of (CLEAR),(SHIFT)+(>) and (CLEAR),(SHIFT)+(<) will perform the 'magazine merge' of a non-ASCII format BASIC PROGRAM file, and this may be usefull to somebody even though we have a built in CASSETTE MERGE of ASCII format files. DO NOT USE THE MAGAZINE MERGE TO MERGE TWO PROGRAMS THAT BOTH HAVE A WORKSAVER KEY DEFINE TABLE. YOU WILL END UP WITH TWO KEY DEFINE TABLES WHICH WILL RESULT IN NEEDLESSLY WASTING MEMORY.

WORKSAVER ON/OFF SWITCH:

Although the WORKSAVER was written to be totally transparent to the user, there are circumstances that may require the user to temporarilly turn off the WORKSAVER. These circumstances are most frequently encountered when another machine language program interferes with the WORKSAVER. Such interference can be very disturbing if it results in locking up the computer.
CAUTION: AFTER ENTERING A PROGRAM THAT HAS A MACHINE LANGUAGE PROGRAM IN IT, YOU SHOULD SAVE IT BEFORE TRYING TO RUN IT. IF YOU FIND A PROBLEM WITH THE PROGRAM RUNNING WITH THE WORKSAVER, USE THE ON/OFF SWITCH AS DESCRIBED BELOW.
To turn off the WORKSAVER follow these steps.

1. Check the location of the WORKSAVER by (BREAK),(B). The forth number is the location of the WORKSAVER.
2. Add 4 to the WORKSAVER location found in step 1
3. Execute this address by: EXEC n, and the WORKSAVER will turn off.
4. To turn it back on execute that same address.

You can also define a USRFNC to be at this location and then turn the WORKSAVER on and off while runnimg a program.

NOTE: If you have a machine language program that hangs up the computer when it is run with the WORKSAVER on, try turning off the WORKSAVER. Then after activating the machine language program, try turning on the WORKSAVER. The WORKSAVER will then attempt to tie itself to the OTHER program, and they may work together.

PLATINUM SOFTWARE INC. WORKSAVER
ADDENDUM VERSION A5

<<<< THE INFORMATION PRESENTED IN ADDENDUM A4 APPLIES TO >>>>
<<<< THE A5 VERSION ALSO >>>>

We have revised the WORKSAVER once again in our effort to refine its function to better suit you, the user. Besides the minor improvements that have been made, the A5 is 100% compatible with the new WORKSAVER PLUS (see enclosed info). That is, the PLUS can be purchased later and attached to this version of the WORKSAVER.

MANUAL CORRECTIONS:

PAGE 1: The line above 'INITIALIZATION EXAMPLE:' reads 'After you press the 'ENTER' key, you will be greeted with the familiar 'OK' statement'. The new A5 version replaces the 'OK' prompt with '>>'.

PAGE 2: The CLOADM"",&HA00 has caused some confusion because the "" has been incorrectly interpretted to be '''' instead of the correct "".

PAGE 4: @ KEY FUNCTIONS HAVE CHANGED AS FOLOWS:
1) When the (@) key is pressed the cursor will stop flashing.
2) The (@) key must be pressed twice to print the @ character.
3) (@),(Any screen printable key) searches for the first occurance of that key on the screen to the right of the cursor. The search continues to the lower right-hand side of the screen. If no match is found, the cursor will not move, but it will start flashing again.
4) If (@) is pressed accidently, press the (ENTER) key to abort on screen search, and return cursor to flashing.
5) (@),(LEFT ARROW) moves cursor up one screen line
(@),(RIGHT ARROW) moves cursor down one screen line
NOTE: You can remember which key moves up or down by looking at the arrow keys on the keyboard. The left arrow key points at the up arrow key and functions like the up arrow key. Also the up arrow key is slightly left of the right arrow key thus the left arrow key functions like the up arrow key and the right arrow key functions like the down arrow key.
6) (@),(BREAK) moves cursor to upper left-hand corner of the screen.
7) (@),(CLEAR) moves cursor to lower left-hand corner of the screen.

PAGE 8: C.MEM now prints five numbers. The first four are as descibed in the manual, and the fifth is the location of the top of the WORKSAVER. Subtracting the forth number from the fifth will give you the length of the WORKSAVER in bytes.

ANY PAGE Throughout the manual ignore any reference to using (@),(LEFT ARROW) to do a backward search. The feature has been dropped due to lack of use.

ANY PAGE The control cursors have been changed from solid blocks to blocks with one or more corners filled in with black. This will help users with black and white TV's distinguish the different modes.

WORKSAVER FEATURES

EDIT CONTROLS

Delete characters	(SHIFT) + (LEFT ARROW)
Erase to end of line	(CLEAR), (RIGHT ARROW)
Erase line	(BREAK), (0)
Insert spaces	(SHIFT) + (RIGHT ARROW)
Move cursor to left edge of screen	(CLEAR), (LEFT ARROW)
Split lines	(CLEAR), (DOWN ARROW)

SYSTEM CONTROLS

(CLEAR), (SHIFT) + (<) load table and append to existing BASIC program
(CLEAR), (SHIFT) + (>) load BASIC program and attach existing table to it.
(CLEAR), (SHIFT) + (9) cassette merge
(CLEAR), (SHIFT) + (*) dynamic edit: auto dynamic edit commands NEW, LOAD, CLOAD, CLEAR, PCLEAR, DEL, RENUMBER, MERGE

PROGRAM CONTROL FROM INPUT/LINEINPUT COMMANDS

DYNAMIC INPUT (SHIFT) + (ENTER); use '?' to compute total
PROGRAM BREAK (SHIFT) + (BREAK)

REDEFINE KEY PROCEDURES

((CLEAR) or (BREAK)), (SHIFT) + (@) initiate redefine key
1. enter new key definitions
2a. (SHIFT) + (@) defines print only type
2b. (ENTER) auto execute type
2c. (SHIFT) + (ENTER) transparent type
3. PRESS KEY to store new definition

@ KEY

(@), (ANY CHARACTER) search for character to right of key
(@), (RIGHT ARROW) move cursor up one line
(@), (LEFT ARROW) move cursor down one line
(@), (BREAK) move cursor to top left corner
(@), (CLEAR) move cursor to bottom left corner

PLUS FEATURES

PROGRAM LISTING CONTROL

*(SHIFT) + (UP ARROW) scroll program listing up the screen
*(CLEAR), (SHIFT) + (UP ARROW) delete spaces after line numbers from bottom of screen. (UP ARROW) deletes successive lines up the screen. Any key ends this function.
*(SHIFT) + (DOWN ARROW) scroll program listing down screen
*(CLEAR), (SHIFT) + (DOWN ARROW) delete spaces after line numbers from top of screen. (DOWN ARROW) deletes successive lines down the screen. Any key ends this function.

GLOBAL SEARCH AND REPLACE (SHIFT) + (@)

/.NX.	Simple search for NX throughout a program
/.NX.NEXTX.	Simple search for NX and replace with NEXTX throughout a program
/.NX.NEXTX.-100	/.NX.NEXTX.100-200 /.NX.NEXTX.-200 search and replace syntax for line ranges.
/".MAY. JUNE.	Search and Replace inside strings.
/'.SORT DATE.	Search remarks for SORT DATE
/. ..	Delete spaces in a program
/.'.	Search remarks in a program. USE .'. SYNTAX ONLY. Delete remarks by using @-delay and (CLEAR), (RIGHT ARROW).

LINE RENUMBERING

54;50	Renumber line 54 to be line 50
100-112;400	Renumber lines 100 through and including line 112 to be lines 400 through 412.
100-112;400;5	Renumber lines with an increment of 5

LINE RENUMBERING ERROR MESSAGES

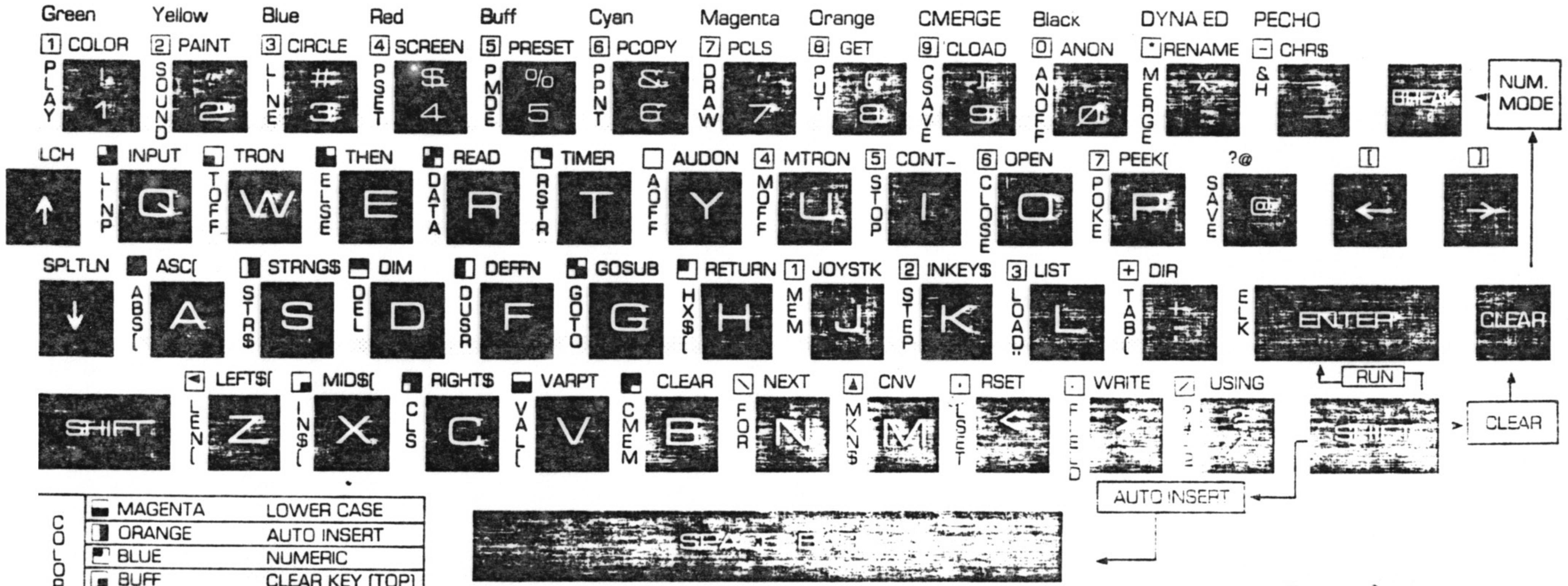
?UL ERROR generated when the lines specified to be moved do not exist.
?FC ERROR generated when lines to be moved would overwrite or surround existing lines.
UL xxxx IN yyyy generated during renumbering process to indicate lines which are referenced but do not exist.

SCREEN SCROLL CONTROL

(BREAK), (UP ARROW) prints the # promt for number of lines to list between screen pauses.
(BREAK), (DOWN ARROW) cancels scroll control
(ANY KEY) continues listing or use (DOWN ARROW) to cancel control

PRINTER ECHO

(CLEAR), (SHIFT) + (=) echoes screen print to the printer. To turn off execute another (CLEAR), (SHIFT) + (=).
(CLEAR), (SHIFT) + (RIGHT ARROW) sends a logical screen line to printer. Subsequent (RIGHT ARROW)'s list next line to printer. Any other key ends this feature.


Green Yellow Blue Red Buff Cyan Magenta Orange CMERGE Black DYNA ED PECHO
1 COLOR 2 PAINT 3 CIRCLE 4 SCREEN 5 PRESET 6 PCOPY 7 PCLS 8 GET 9 CLOAD 0 ANON • RENAME – CHR$
PLAY SOUND LINE PSET PMDE PPNT DRAW PUT CSAVE ANOFF MERGE &H
BREAK
NUM. MODE
LCH INPUT TRON THEN READ TIMER AUDON 4 MTRON 5 CONT- 6 OPEN 7 PEEK(?@ [ ]
LINP TOFF ELSE DATA RSTR AOFF MOFF STOP CLOSE POKE SAVE
SPLTLN ASC(STRNG$ DIM DEFFN GOSUB RETURN 1 JOYSTK 2 INKEY$ 3 LIST + DIR
ABS(STR$ DEL DUSR GOTO HX$(MEM STEP LOAD: TAB(ELK
ENTER
CLEAR
RUN
LEFT$(MID$(RIGHT$ VARPT CLEAR NEXT CNV RSET WRITE USING
LEN(INS(CLS VAL(CMEM FOR MKN$ LSET FIELD
SHIFT
CLEAR
AUTO INSERT
SPACE

COLOR		
	▬ MAGENTA	LOWER CASE
	◧ ORANGE	AUTO INSERT
	◩ BLUE	NUMERIC
	◪ BUFF	CLEAR KEY (TOP)
	◰ RED	BREAK KEY (LFT)

platinum software