VisiCalc®

Programmer's Reference Manual

For Use With the HP-75

May 1983

00075-90223

Printed in U.S.A.

Introduction

HP-75 VisiCalc has capabilities that make it unlike any other spreadsheet program. Some of these features, already discussed in the *VisiCalc Owner's Manual*, are:

- Multiple worksheets can exist in user memory simultaneously, and worksheets can reference each other.
- Formulas in a worksheet can contain calls to specially designed BASIC programs, called extension functions.
- Messages displayed by VisiCalc can be changed by placing substitute messages in a text file called `VISIMSGS`.

The power of HP-75 VisiCalc doesn't stop here, however. There are two additional features designed for more advanced users:

- The many BASIC language keywords provided by the VisiCalc module can be used to write BASIC programs that create, examine, and modify worksheets—as programs independent of VisiCalc, as extension functions called from a cell, or as programs that extend the command set. These language extensions constitute a spreadsheet language—a language designed for and oriented toward spreadsheet operations.
- The set of commands that VisiCalc can execute can be extended. Delete, Insert, Format, Global, etc. are no longer the only commands available to modify worksheets. Commands can be created to do whatever you want, by altering certain messages in `VISIMSGS`, adding new ones, and creating specialized BASIC programs.

The *VisiCalc Programmer's Reference Manual* describes these two new capabilities and some of their uses. However, the manual is written primarily as a reference for programmers and contains very few examples. To take full advantage of these capabilities, you should thoroughly understand HP-75 VisiCalc, and be well-versed in HP-75 BASIC language programming and the operation of the HP-75.

We expect that it will take some time to become proficient with the concepts and keywords, through a combination of reading the manual and experimenting with the many features. Once you gain this expertise, you will be able to extend HP-75 VisiCalc to uses never before envisioned for a spreadsheet program.

Contents

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Section 1

Getting Started

Up to now, you have seen just one side of HP-75 VisiCalc—what you have learned from the *VisiCalc Owner's Manual.* VisiCalc is actually a product with a dual identity:

- **An interactive tool for performing calculations on tabular data.** This part of VisiCalc provides the user interface that allows you to examine, format, and modify cells, display and print worksheets, etc.—in short, everything described in the *VisiCalc Owner's Manual.* When you run VisiCalc, you actually run this BASIC program.
- **A spreadsheet language for direct access of worksheets from BASIC.** This is the set of BASIC language keywords, provided by the VisiCalc module, that extends HP-75 BASIC. This set of keywords allows direct worksheet access. When the VisiCalc interactive program accesses or modifies worksheets, it uses these keywords.

The *VisiCalc Owner's Manual* illustrated the interactive nature of VisiCalc. In the *VisiCalc Programmer's Reference Manual,* the spreadsheet language will be described. When using this library of keywords, you will interact with worksheets in a different way. For example, suppose you want to examine cell `Mar^TotExp` in worksheet `BUDGET`. There are two ways to do this—by running VisiCalc, or by using the spreadsheet keywords. You already know how to do the former—by using the arrow keys and the Go To command, pressing [APPT] to see the formula, etc. Now let's see how to examine `Mar^TotExp` with the keywords.

Load the `BUDGET` worksheet into memory, and then execute the `WORKSHEET` statement to specify `BUDGET` as the worksheet to be examined:*

```
WORKSHEET 'BUDGET'
```

* To execute a statement from the keyboard of the HP-75, you must press [RTN] after typing the statement. We will not show this implied keystroke in the examples in this manual.

VisiCalc coordinates like A1 or Jan^Income are not used by the keywords. Instead, column and row coordinates are represented as numbers from 0-255. Below is a worksheet showing the numeric coordinates of different cells, including the column and row headers:

		A	B	C	D
	0,0	1,0	2,0	3,0	4,0
1	0,1	1,1	2,1	3,1	4,1
2	0,2	1,2	2,2	3,2	4,2
3	0,3	1,3	2,3	3,3	4,3
4	0,4	1,4	2,4	3,4	4,4

The column coordinate is first in the coordinate pair, just as the column coordinate is first in coordinate A1 or Jan^Income. The column headers are in row 0, and the row headers are in column 0. The upper-left corner, coordinate 0,0, is referred to as *cell zero*; it has a special function and is discussed later in this manual. Three functions, COL, ROW, and COORD$, are available to perform transformations between numeric coordinates and VisiCalc coordinates.

Mar^TotExp corresponds to column 3, row 9. Since you don't necessarily know what type of cell this is, you must find out by typing:

```
CELLTYPE(3,9)
```

The CELLTYPE function returns 2, which means that the cell contains a formula. Knowing that, the cell can be examined with user-defined headers:

```
GETFORMULA$(3,9,1)
```

or with default headers:

```
GETFORMULA$(3,9,0)
```

The result can be examined in formatted form:

```
GETVALUE$(3,9,0)
```

or unformatted form:

```
GETVALUE$(3,9,1)
```

The local format can also be examined:

```
GETFORMAT(3,9)
```

These commands have counterparts called PUTFORMULA, PUTVALUE, and PUTFORMAT that allow you to change the cell contents. For example, PUTFORMAT 3,9,2 will put the Integer format (/FI) into the cell.

This simple example is just the beginning. The VisiCalc keywords provide the ability to do any operation performed by the VisiCalc interactive program—insert, delete, and move columns and rows, change display modes, etc. The descriptions in the pages that follow will show you the range of capabilities available. Beyond that, your imagination is the only limit to the applications of BASIC programs using the VisiCalc spreadsheet language.

Keyword Descriptions

This is the description of the keywords provided by the VisiCalc module. First, the name of each keyword is given, and whether the keyword is a function or a statement. Then the expected parameters are listed. Optional parameters are surrounded with brackets ([ ]). Bulleted items are any cautions that should be observed when using the keyword, and are listed next. Finally, a list of errors that may be reported by the keyword is given. For details on the specific conditions that can cause the listed errors to occur, refer to appendix A, "VisiCalc Errors and Warnings."

ACTIVE$ — function

```
ACTIVE$
```

Returns the file name of the active worksheet, as long as that worksheet is in RAM, and the scratch space needed by VisiCalc exists. In all other cases, ACTIVE$ will return a null string.

Errors: none

AVERAGE — function

```
AVERAGE(list)
```

Returns the average of the items in the *list* when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. The average of the items in the *list* is the sum of the items divided by the number of numeric parameters in the *list*. If the number of numeric parameters is zero, returns ERROR. Because AVERAGE computes using internal 16-digit math routines, it will yield a more precise result than if you compute the average manually. AVERAGE is the same as MEAN.

Errors: none

BLANK — statement

```
BLANK column coordinate , row coordinate
```

Blanks the designated cell and reclaims the space used by that cell, unless the cell contains a local format. If there is a local format, the space used by the cell can only be reclaimed by resetting the local format to /FD with PUTFORMAT either before or after using BLANK.

Errors: 200, 201, 202

CALLVC

statement

CALLVC '*file name*'

Performs a STOVAR and a CALL. Does not turn on the PRGM annunciator.

- If the BASIC program to be called using CALLVC is in a module, that program must be the first non-LEX file in the module. This applies particularly to visi-commands, since every visi-commands program is called by VisiCalc using CALLVC. If the visi-commands program is in a module, the program must be the first non-LEX file in the module.
- See the cautions for STOVAR.

Errors: 16, 200, 202, same as the HP-75 CALL statement

CELLTYPE

function

CELLTYPE(*column coordinate*,*row coordinate*)

Returns a number representing the type of the designated cell. Cell types are indicated by the following numbers:

Number	Cell Type
1	value
2	formula
3	formula with external references
4	label
5	repeating label
6	blank*
7	column or row default header
8	column or row user-defined header

* A cell located beyond (MAXCOL,MAXROW) is considered to be blank.

Errors: 200, 201

CHOICE$ function

```
CHOICE$(message number)
```

Returns the legal choices from the designated message. The choices are the underlined characters in the message, returned without underlines. If there are no choices, returns a null string. CHOICE$ looks for the designated message first in the text file VISIMSGS. If the file does not exist in user memory, or if it exists and is not a text file, the choices from the corresponding message in the VisiCalc module are returned. CHOICE$ will skip over a single leading blank in a message in case the text file contained a blank, for readability, between the line number and the message.

Errors: 16, 42

CLRLCD statement

```
CLRLCD [start position [,end position]]
```

Clears the LCD from the *start position* to the *end position*. This does not clear the input buffer—only the LCD. Used to selectively clear portions of the LCD, particularly when different size LCD windows are being used (refer to SETWIN for a discussion of LCD windows). CLRLCD with no parameters is equivalent to CLRLCD 1,32. CLRLCD with one parameter is equivalent to CLRLCD *start position*,32.

Errors: 89

CLRSCR statement

```
CLRSCR
```

Clears the current DISPLAY IS device by sending ESC H ESC J. If used in visi-commands, sends the same escape sequence to the current PRINTER IS device.

Errors: none

COL function

```
COL('VisiCalc coordinate')
```

Returns the number of the column coordinate that the *VisiCalc coordinate* represents—the inverse of COORD$. Accepts coordinates with default or user-defined headers. Trailing blanks are ignored, and brackets are required for user-defined headers. Returns 0 if the coordinate is a row header, the header is

not present, or the coordinate is an illegal syntax or an external coordinate. Therefore, a particular coordinate does not exist if both COL and ROW return 0.

Errors: 200

COLWIDTH$ function

COLWIDTH$(*column coordinate*)

Returns the local column width of the designated column. If there is no local column width for that column, returns a null string.

Errors: 200, 201

COORD$ function

COORD$(*column coordinate*, *row coordinate*, *specifier*)

Returns a VisiCalc coordinate given the numeric *column* and *row coordinates*. The *specifier* is a number that specifies the type of coordinate desired, as follows:

Specifier	Coordinate Type
0	default headers
1	user-defined headers
2	user-defined headers with no brackets (only if both headers are user-defined)
3	user-defined headers with nulls
4	user-defined headers with nulls and no brackets (type 2 and 3 combined)
5	type 4 with a space instead of a caret

A null user-defined header is created by using PUTLABEL to put a null string into the header area (column or row 0), or by typing [ ] as a header when the /HC or /HR command is used from VisiCalc. Below are examples of the different specifiers for the SAMPLE worksheet:

SAMPLE	A Jan	B	C (null)
1 2 Sales 3 (null)			

Specifier 0	Specifier 1	Specifier 2	Specifier 3	Specifier 4
0,0,0 = *error*	0,0,1 = *error*	0,0,2 = *error*	0,0,3 = *error*	0,0,4 = *error*
0,1,0 = 1	0,1,1 = 1	0,1,2 = 1	0,1,3 = 1	0,1,4 = 1
0,2,0 = 2	0,2,1 = [Sales]	0,2,2 = Sales	0,2,3 = [Sales]	0,2,4 = Sales
0,3,0 = 3	0,3,1 = 3	0,3,2 = 3	0,3,3 = []	0,3,4 = *null*
1,0,0 = A	1,0,1 = [Jan]	1,0,2 = Jan	1,0,3 = [Jan]	1,0,4 = Jan
1,1,0 = A1	1,1,1 = [Jan]1	1,1,2 = [Jan]1	1,1,3 = [Jan]1	1,1,4 = [Jan]1
1,2,0 = A2	1,2,1 = [Jan^Sales]	1,2,2 = Jan^Sales	1,2,3 = [Jan^Sales]	1,2,4 = Jan^Sales
1,3,0 = A3	1,3,1 = [Jan]3	1,3,2 = [Jan]3	1,3,3 = [Jan]	1,3,4 = Jan
2,0,0 = B	2,0,1 = B	2,0,2 = B	2,0,3 = B	2,0,4 = B
2,1,0 = B1	2,1,1 = B1	2,1,2 = B1	2,1,3 = B1	2,1,4 = B1
2,2,0 = B2	2,2,1 = B[Sales]	2,2,2 = B[Sales]	2,2,3 = B[Sales]	2,2,4 = B[Sales]
2,3,0 = B3	2,3,1 = B3	2,3,2 = B3	2,3,3 = B	2,3,4 = B
3,0,0 = C	3,0,1 = C	3,0,2 = C	3,0,3 = []	3,0,4 = *null*
3,1,0 = C1	3,1,1 = C1	3,1,2 = C1	3,1,3 = 1	3,1,4 = 1
3,2,0 = C2	3,2,1 = C[Sales]	3,2,2 = C[Sales]	3,2,3 = [Sales]	3,2,4 = Sales
3,3,0 = C3	3,3,1 = C3	3,3,2 = C3	3,3,3 = []	3,3,4 = *null*

A sixth column, titled Specifier 5, would be just like the Specifier 4 column, except that `1,2,5 = Jan Sales`.

Errors: 200, 201

CURSOFF

statement

```
CURSOFF
```

Turns off the cell cursor on the video by sending an escape sequence for cursor addressing followed by the cell contents, formatted according to cell type, format, and column width. Uses the column and row coordinates of the current cell and of the upper-left corner of the video (status 5, 6, 7, and 8), the local and global column widths, and header suppression (status 13) to determine how to address the cursor and where to send the cell contents. Sends nothing to the video if the column is too wide to fit.

- The escape sequence and formatted cell contents are always sent to the current `PRINTER IS` device. If the current `PRINTER IS` device is not an HP 82163 Video Interface (or equivalent), it may respond differently to the escape sequences sent by `CURSOFF`.

Errors: 200, 202

CURSON

statement

```
CURSON
```

Turns on the cell cursor on the video by sending an escape sequence for cursor addressing followed by the highlighted cell contents, formatted according to cell type, format, and column width. Uses the column and row coordinates of the current cell and of the upper-left corner of the video (status 5, 6, 7, and 8), the local and global column widths, and header suppression (status 13) to determine how to address the cursor and where to send the cell contents. Sends nothing to the video if the column is too wide to fit.

- The escape sequence and formatted cell contents are always sent to the current PRINTER IS device. If the current PRINTER IS device is not an HP 82163 Video Interface (or equivalent), it may respond differently to the escape sequences and highlighted characters sent by CURSON.

Errors: 200, 202

DECOMP$

function

```
DECOMP$(specifier)
```

Takes the result of the PARSE function (i.e., the internal form of a formula in VisiCalc's scratch space) and returns it as a formula. Equivalent to GETFORMULA$ for a formula in VisiCalc's scratch space. If the *specifier* is 0, the formula will be returned with coordinates containing default headers. If the *specifier* is 1, the formula will be returned with coordinates containing user-defined headers.

- See the cautions for MARK.

Errors: 16, 89, 200, 202

DELCOL

statement

```
DELCOL column coordinate
```

Deletes the specified column, causing all columns to the right of the designated column to move to the left one column position. The user-defined header for that column will also be deleted. Formulas are adjusted so that any cell references (except external and underlined) will still reference the correct data. Local column widths will also be adjusted.

When a column is deleted, any references to cells in the deleted column will be displayed with default headers and underlined, and the value of the cell containing the deleted references will be ERROR.

DELCOL 0 is a special case equivalent to using BLANK on every row header. The other columns remain unmoved, and no cell references or column widths are adjusted. Because of this, DELCOL 0 is not the reverse of INSCOL 0.

Errors: 200, 201, 202

DELROW

statement

DELROW *row coordinate*

Deletes the specified row, causing all rows below the designated row to move up one row position. The user-defined header for that row will also be deleted. Formulas are adjusted so that any cell references (except external and underlined) will still reference the correct data.

When a row is deleted, any references to cells in the deleted row will be displayed with default headers and underlined, and the value of the cell containing the deleted references will be ERROR.

DELROW 0 is a special case equivalent to using BLANK on every row header. The other rows remain unmoved, and no cell references are adjusted. Because of this, DELROW 0 is not the reverse of INSROW 0.

Errors: 200, 201, 202

DIR$

function

DIR$(*specifier*,'*file type*')

Returns file names of the designated type chronologically, or a null string if there are no files of the designated type. Returns names of user memory files only. The *specifier* determines which file name will be returned, as follows:

Specifier	File Returned
1	next newer file
2	next older file
3	newest file
4	oldest file

- *Specifier* 3 performs initialization, and must be used once before using *specifiers* 1, 2, or 4. If the file catalog is altered after initialization (file name changes, or files added, deleted, or timestamped), DIR$ must be reinitialized using *specifier* 3.

- If there is more than one file in user memory with the same creation time, the most recently created file may be the only file returned by DIR$.

Errors: 89, 200

DSP$ function

```
DSP$
```

Returns the name of the first DISPLAY IS device that was active when VisiCalc was started. If an ASSIGN IO has not been done, if there are no DISPLAY IS devices, or if an OFF IO has been done, a null string is returned. Refer to VC for more details on how VisiCalc handles display devices.

- DSP$ will always return the name of the first DISPLAY IS device while VisiCalc is running. When VisiCalc ends, this name will still be correct until the name or location of the first DISPLAY IS device changes. When this happens, DSP$ will continue to return the previous DISPLAY IS device name until VisiCalc is run again.
- In an extension function, it may be necessary to determine if a display device exists, and if it is active. DSP$ indicates the presence of a display device, and status 18 reflects whether or not it is active (/VY or /VN). GETSTATUS(18)*LEN(DSP$)>0 returns 1 if there is an active display device, and 0 if there is either no display device or an inactive one. For example, to clear the screen only if there is an active display device, use IF GETSTATUS(18)*LEN(DSP$)>0 THEN CLRSCR.

Errors: 200

DUMP function

```
DUMP(start column, start row, end column, end row, specifier)
```

Sends the active worksheet to the printer or video. The portion of the worksheet that will be printed or displayed has the upper-left corner at (*start column*, *start row*) and the lower-right corner at (*end column*, *end row*). Each cell is formatted according to cell type, format, and column width.

The specifier is a number that determines what type of device the worksheet is being sent to, whether or not to display or print the headers, how the headers will be set off from the rest of the worksheet, and whether or not the ATTN key can be used to interrupt the display or print operation:

Specifier	Device	Include Headers	How Set Off Headers?	ATTN Key Interrupt?
−1	video	yes	inverse video	no
0	printer	no		yes
1	printer	yes	dashes and bars	yes

For a *specifier* of −1 (video), DUMP will first send an escape sequence to home the cursor. Each row is sent with the proper number of cells followed by as many blanks as are needed to send a total of VWIDTH−1 characters.

For *specifiers* 1 and 0 (printer with and without headers), DUMP returns 1 if the ATTN key was pressed, or 0 if not. For a *specifier* of −1, DUMP always returns 0.

- The worksheet is always sent to the current PRINTER IS device. When using a *specifier* of −1 (video), if the current PRINTER IS device is not an HP 82163 Video Interface (or equivalent), it may respond differently to the highlighted characters sent by DUMP.
- Since headers are displayed or printed depending on the value of the *specifier*, the boundaries of the area to be displayed or printed should not include column or row 0.

Errors: 16, 200, 202

ENDVC

statement

```
ENDVC
```

Ends VisiCalc when used in visi-commands or in an extension function. Resets the active worksheet name to null as if the WORKSHEET statement with no parameter was executed. Restores the original values of the HP-75 global conditions that are altered by VisiCalc (refer to VC for further details). Has no effect from the keyboard or from any BASIC program not called from VisiCalc.

Errors: 200

ERROR

function

```
ERROR
```

Returns the value of ERROR when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. If ERROR and NA are both in a formula, the result is whichever of them appears first.

Errors: 200, 202

FIRSTREF$ function

FIRSTREF$(*column coordinate*, *row coordinate*, *specifier*)

Returns either the VisiCalc coordinate of the first cell reference (except external and underlined) in a formula, or a null string if the formula contains no cell references. If the specifier is 0, coordinates containing default headers will be returned. If the specifier is 1, coordinates containing user-defined headers will be returned.

- See the cautions for MARK.

Errors: 200, 201, 202

GETFORMAT function

GETFORMAT(*column coordinate*, *row coordinate*)

Returns a number representing the local format from the designated cell. Local formats are indicated by the following numbers:

Number	Format
0	/FD
1	/FG
2	/FI
3	/FL
4	/FR
5	/F$
6	/F*

Errors: 200, 201

GETFORMULA$ function

GETFORMULA$(*column coordinate*, *row coordinate*, *specifier*)

Returns the formula from the designated cell. If the cell is a value, returns the value as a string. Returns a null string if the cell is not a value or a formula. If the *specifier* is 0, formulas will be

returned with coordinates containing default headers. If the *specifier* is 1, formulas will be returned with coordinates containing user-defined headers.

- Refer to "Warnings When Formulas Are Returned" in appendix A, and the cautions for MARK.

Errors: 16, 200, 201, 202

GETLABEL$ function

```
GETLABEL$(column coordinate, row coordinate)
```

Returns the label or repeating label from the designated cell. Returns a null string if the cell is not a label or repeating label.

Errors: 200, 201

GETSETUP$ function

```
GETSETUP$
```

Returns the printer setup string. Returns a null string if there is no setup string.

Errors: 200

GETSTATUS function

```
GETSTATUS(status number)
```

Returns a number representing the designated status item. The status information returned corresponds to the following status numbers:

Status Number	Description	Possible Values
0	unused	0-255
1	Recalculation mode	0 = Automatic 1 = Manual
2	recalculation order	0 = by columns 1 = by rows
3	Global format	1 = /GFG 2 = /GFI 3 = /GFL 4 = /GFR 5 = /GF$ 6 = /GF*
4	global column width	0-255
5	column coordinate of current cell	1-255
6	row coordinate of current cell	1-255
7	column coordinate of upper-left corner of video	1-255
8	row coordinate of upper-left corner of video	1-255
9	column coordinate of window access cell	1-255
10	row coordinate of window access cell	1-255
11	Formula/Result Display mode	0 = Result 1 = Formula
12	VisiCalc level—two-way communication path between VisiCalc and an extension function called from cell zero*	
	• passed from VisiCalc to the extension function	1 = start-up 2 = cell entry 3 = exit
	• passed by the extension function to VisiCalc	0 = update video if necessary 1 = update video
13	suppress default headers	0 = no 1 = yes

* For a discussion of the use of status 12, refer to "Cell Zero."

Status Number	Description	Possible Values
14	Coordinate Display mode for cell contents field of cell display	0 = default headers 1 = user-defined headers
15	Coordinate Display mode for cell coordinates field of cell display	0 = default headers 1 = user-defined headers
16	Display Format mode	0 = Formatted 1 = Unformatted
17	coordinate display specifier—determines the type of coordinates that will be displayed in the cell coordinates field of the cell display (used by COORD$ at Top level)	0 = default headers 1 = user-defined headers 2 = user-defined headers with no brackets 3 = user-defined headers with nulls 4 = user-defined headers with nulls and no brackets (type 2 and 3 combined) 5 = type 4 with a space instead of a caret
18	video active	0 = inactive 1 = active
19	external cell references in worksheet	0 = no external references 1 = at least one external reference

Errors: 89, 200

GETVALUE$ function

GETVALUE$(*column coordinate*, *row coordinate*, *specifier*)

Returns, as a string, the numeric value of the designated cell. If a cell is nonnumeric (label, repeating label, blank, or header), it will return zero. If the value of the cell is ERROR or NA, the strings ERROR or NA will be returned. If the *specifier* is 0, values will be returned formatted according to the cell format. If the *specifier* is 1, values will be returned unformatted.

Errors: 16, 200, 201

GETWIDTH

function

```
GETWIDTH(column coordinate)
```

Returns the numeric value of the column width for the designated column. If there is no local column width, returns the global column width.

Errors: 200, 201

HIGH$

function

```
HIGH$('string')
```

Returns the *string* with each character highlighted—underlined in the LCD or inverse video on the display device. This is equivalent to adding 128 to the numeric value of each ASCII character. If a character is already highlighted, leaves it unchanged.

Errors: none

INCAT

function

```
INCAT('file name','file type')
```

Returns a number that describes the existence, in user memory only, of the *file name* with the designated *file type*. The number returned has the following meaning:

Number	Meaning
1	File is nonexistent.
2	File exists and is the designated file type.
3	File exists and is not the designated file type.
4	Invalid file name.

Errors: 89

INLINE$

function

```
INLINE$('input string',window start,cursor position,'terminator string',cursor type)
```

Extends the capabilities of the HP-75's INPUT statement and KEY$ function to provide greater flexibility in accepting keyboard input. INLINE$ allows the user to press any combination of keys for

input and editing, just like the INPUT statement. While INPUT terminates execution only when specific keys are pressed, such as [RTN], any number of different keys can be defined that will cause INLINE$ to terminate execution and exit. These keys can be in addition to or instead of the set of keys that terminate the INPUT statement. When one of these terminating keys is pressed, INLINE$ returns the ASCII character corresponding to that key, just like KEY$.

The five parameters for INLINE$ have the following meanings:

Parameter	Meaning
Input string	The string that is being edited.
Window start	The first character position of the input string that appears at the left end of the LCD window (as specified by SETWIN).
Cursor position	The location within the input string where the cursor appears.
Terminator string	The string of ASCII characters representing the keys that, when pressed by the user, will cause INLINE$ to terminate.
Cursor type	0 = replace cursor, 1 = insert cursor

INLINE$ can best be illustrated with an example. During value entry while running VisiCalc, there are five keys that cause VisiCalc to take some action. All other keys behave as they would during normal HP-75 input and editing: character entry, insert/replace mode, delete, backspace, clear, arrows, shift-arrows, control-arrows, etc. The five terminator keys defined during value entry are as follows:

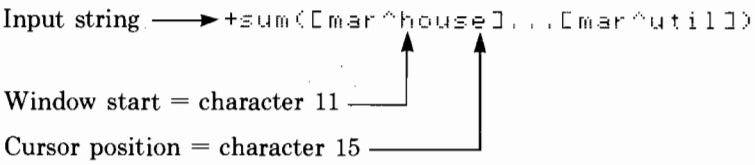
Terminator Key	Meaning
[RTN]	Accept entry and return to Top level.
[ATTN]	Return to Top level.
[TIME]	Toggle cell coordinates between default and user-defined headers.
[!]	Calculate the entry.
[SHIFT] [ATTN]	Exit VisiCalc.

Suppose that while typing in the sum([mar^house],,,[mar^util]) formula in the BUDGET worksheet, you have a display that looks like this:

```
Mar^TotalExp(V): hous◆],,,[mar^u
```

Because Mar^TotalExp(V): is 17 characters long, the LCD window is set to 18,32 with SETWIN 18 (refer to SETWIN for a discussion of LCD windows). The h is the first character of the

input string that appears in this LCD window. The five parameters for INLINE$ that set up this display are as follows:



Terminator string: CHR$(13) ([RTN]), CHR$(128) ([ATTN]), CHR$(129) ([TIME]), [!], CHR$(160) ([SHIFT][ATTN])

Cursor type = 1 (insert cursor)

When one of the terminator keys is pressed, INLINE$ returns the ASCII character corresponding to that key. As the visi-commands examples illustrate, the easiest way to process that key in BASIC is to find the position of that key in the original terminator string used by INLINE$, and use that position in an ON GOTO or an ON GOSUB.

Unlike INPUT, which assigns the input to variables in the input list, INLINE$ only returns the key that was pressed. The input itself must be read using separate functions that return the *input string*, *window start*, *cursor position*, and *cursor type*, as follows:

Parameter	Keyword
Input string	INPBUF$
Window start	INPWIN
Cursor position	INPPTR
Cursor type	IRFLAG

- The *input string* is limited to 96 characters. INLINE$ will not function properly if the *input string* is more than 96 characters long.
- There are 17 terminating keys already defined by the HP-75 that will still cause INLINE$ to exit unless they are included in the *terminator string*: [ATTN], [SHIFT][ATTN], [TIME], [SHIFT][TIME], [APPT], [SHIFT][APPT], [EDIT], [↑], [SHIFT][↑], [↓], [SHIFT][↓], [FET], [SHIFT][FET], [CTL][FET], [RTN], [RUN], and [SHIFT][RUN]. If these keys are included in the *terminator string*, they will not cause INLINE$ to exit. You must then either provide some action for these keys, or intercept all of them and ignore specific ones. (During value entry, VisiCalc actually intercepts all these keys, but only defines action for four of them.)
- It is possible to prevent INLINE$ from exiting by providing a *terminator string* containing ASCII characters that are not on the keyboard. Be sure to include in the *terminator string* at least one character that can be produced by pressing a key from the keyboard.

- If the four keywords INPBUF$, INPWIN, INPPTR, and IRFLAG are not used immediately after INLINE$ exits, it is possible for that information to be lost. It is therefore recommended that the four keywords be used immediately after INLINE$ exits.

Errors: 16, 89, 200

INPBUF$ function

```
INPBUF$
```

Returns the entire contents of the input buffer. The input buffer is the 96-character location where all keyboard input is typed during editing, command entry, INPUT, and INLINE$. Used mainly in conjunction with INLINE$ to establish the entire string that was input by the user.

Errors: none

INPPTR function

```
INPPTR
```

Returns the position of the edit cursor within the input buffer. Used mainly in conjunction with INLINE$.

Errors: none

INPWIN function

```
INPWIN
```

Returns the position of the first character of the input buffer that appears in the LCD window (as specified by SETWIN). Used mainly in conjunction with INLINE$.

Errors: none

INSCOL statement

```
INSCOL column coordinate
```

Inserts a blank column at the specified column position, causing all columns including and to the right of the designated column to move to the right one column position. The user-defined header for that column will also be moved. Formulas are adjusted so that any cell references (except external or underlined) will still reference the correct data. Local column widths are also adjusted.

INSCOL 0 is a special case equivalent to moving the row headers into column 1 of the worksheet.

Errors: 16, 89, 200, 201, 202

INSROW

statement

INSROW *row coordinate*

Inserts a blank row at the specified row position, causing all the rows including and below the designated row to move down one row position. The user-defined headers for those rows will also be moved. Formulas are adjusted so that any cell references (except external and underlined) will still reference the correct data.

INSROW 0 is a special case equivalent to moving the column headers into row 1 of the worksheet.

Errors: 16, 89, 200, 201, 202

INTERP$

function

INTERP$

Takes the result of the PARSE function (i.e., the internal form of a formula in VisiCalc's scratch space), calculates it, and returns the result (unformatted) as a string. Equivalent to RECALC followed by GETVALUE$ with a specifier of 1 for a formula in VisiCalc's scratch space. If the value of the formula is ERROR or NA, the strings ERROR or NA will be returned.

- INTERP$ must only be used in one of the following two ways: DISP INTERP$ or A$ = INTERP$, where A$ is any string variable. INTERP$ will not work properly if it is used in any other manner such as with string concatenation.
- Refer to "Warnings When Formulas Are Calculated" in appendix A.

Errors: 16, 89, 200, 202

IRFLAG

function

IRFLAG

Returns the cursor type that was in effect when INLINE$ exited (0 = replace cursor, 1 = insert cursor).

- Because IRFLAG is used only in conjunction with INLINE$, the value it returns has no relationship to the cursor type currently being displayed.

Errors: 200

LEGAL

function

```
LEGAL(coordinate)
```

Verifies that a *coordinate* has correct syntax. Accepts coordinates with default or user-defined headers. Trailing blanks are ignored, and brackets are required for user-defined headers. Returns 0 if the *coordinate* is an illegal syntax or an external coordinate, 1 if it is a legal column or row header only, and 2 if it is a legal coordinate with both column and row headers. Does not check for the existence of any headers contained in the *coordinate*.

Errors: 200

MARK

statement

```
MARK
```

Marks a cell reference as relative. The cell reference that will be marked is the one just returned from FIRSTREF$ or NEXTREF$. It is used on a coordinate or a range. To use on a range, use MARK for the first part of the range, NEXTREF$ to get the second part of the range, and MARK again for the second part of the range. References are unmarked when REPLICATE is executed, or by using the UNMARK command.

- None of the keywords that return formulas or portions of formulas (DECOMP$, GETFORMULA$, FIRSTREF$, and NEXTREF$) work properly on formulas containing cell references that were marked as relative using MARK. If a formula contains marked cell references, you will not be able to examine that formula until the cell references are unmarked using UNMARK.

Errors: 200, 202

MAXCOL

function

```
MAXCOL
```

Returns the right-most active column. If there have been no columns entered into the worksheet, returns a 1.

Errors: 200

MAXL

function

```
MAXL(list)
```

Returns the maximum of the items in the *list* when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. Zero is used as the value for labels, repeating labels, and empty cells.

Errors: none

MAXROW

function

```
MAXROW
```

Returns the bottom-most active row. If there have been no rows entered into the worksheet, returns a 1.

Errors: 200

MEAN

function

```
MEAN(list)
```

MEAN is the same as AVERAGE; refer to AVERAGE for more details.

Errors: none

MESSAGE$

function

```
MESSAGE$(message number)
```

Returns the designated message. If there is no message, returns a null string. MESSAGE$ looks for the designated message first in the text file VISIMSGS. If the file does not exist in user memory, or if it exists and is not a text file, the choices from the corresponding message in the VisiCalc module are returned. MESSAGE$ will skip over a single leading blank in a message in case the text file contained a blank, for readability, between the line number and the message.

Errors: 16, 42

MINL

function

```
MINL(list)
```

Returns the minimum of the items in the *list* when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. Zero is used as the value for labels, repeating labels, and empty cells.

Errors: none

MOVCOL

statement

```
MOVCOL source column , destination column
```

Moves the *source column* to the *destination column* position. All columns between the source and the destination, including the destination, will move one position toward the source, and the source will be placed in the destination's previous location. The user-defined headers for the columns will also be moved. Formulas are adjusted so that any coordinates or ranges (except externals and underlined) will still reference the data in its new location. Local column widths are also adjusted.

Errors: 16, 200, 201, 202

MOVROW

statement

```
MOVROW source row , destination row
```

Moves the *source row* to the *destination row* position. All rows between the source and the destination, including the destination, will move one position toward the source, and the source will be placed in the destination's previous location. The user-defined headers for the rows will also be moved. Formulas are adjusted so that any cell references (except external and underlined) will still reference the data in its new location.

Errors: 16, 200, 201, 202

NA

function

```
NA
```

Returns the value of NA when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. If NA and ERROR are both in a formula, the result is whichever of them appears first.

Errors: 200, 202

NEXTREF$

function

```
NEXTREF$
```

Returns either the VisiCalc coordinate of the next cell reference (except external and underlined) in a formula, or a null string if there are no remaining cell references. The coordinates will be returned according to the specifier used in the FIRSTREF$ function. The next cell reference is after the one returned from either NEXTREF$ or FIRSTREF$. Returns a one character string if the cell reference is the 2nd part of a range.

- FIRSTREF$ must be used before using NEXTREF$.
- See the cautions for MARK.

Errors: 200, 202

PARSE

function

```
PARSE('formula')
```

Converts the *formula* into its internal form, and puts it into VisiCalc's scratch space. Returns the character position where the first syntax error occurred, or 0 if there were no syntax errors. Trailing blanks are ignored, and underlines are removed from any underlined characters.

Errors: 16, 42, 200, 202

PRT$

function

```
PRT$
```

Returns the name of the first PRINTER IS device that was active when VisiCalc was started. If an ASSIGN IO has not been done, if there are no PRINTER IS devices, or if an OFF IO has been done, returns a null string. Refer to VC for more details on how VisiCalc handles print devices.

- PRT$ will always return the name of the first PRINTER IS device while VisiCalc is running. When VisiCalc ends, this name will still be correct until the name or location of the first PRINTER IS device changes. When this happens, PRT$ will continue to return the previous PRINTER IS device name until VisiCalc has been run again.

Errors: 200

PRWIDTH? function

```
PRWIDTH?
```

Returns the current PWIDTH. If PWIDTH was INF, returns 9.9999999999E499.

Errors: none

PUTFORMAT statement

```
PUTFORMAT column coordinate, row coordinate, format
```

Puts the *format* into the designated cell. Refer to GETFORMAT for the meanings of the different cell formats.

Errors: 16, 89, 200, 201

PUTFORMULA statement

```
PUTFORMULA column coordinate, row coordinate [, 'formula']
```

Puts the *formula* at the designated cell. Converts the *formula* into its internal form, calculates it, and puts the *formula* and its result into the designated cell. If the optional *formula* is not present, assumes the *formula* has already been converted into its internal form by the PARSE function. Thus PARSE(*formula*) followed by PUTFORMULA C,R has the same effect as PUTFORMULA C,R,*formula*.

PUTFORMULA can also be used to enter a value (as a string) into the worksheet. If a value is entered, it is given type 1 (see the CELLTYPE function). A formula is type 2, and a formula containing external cell references is type 3. ERROR and NA are considered to be formulas.

- Refer to "Warnings When Formulas Are Calculated" in appendix A.

Errors: 16, 42, 89, 200, 201, 202

PUTLABEL

statement

```
PUTLABEL column coordinate, row coordinate, 'label'
```

Puts the *label* into the designated cell. If labels containing highlighted characters (underlined in the LCD, inverse video on the video) are put in the header area (column or row 0), the highlight will be removed.

Errors: 16, 42, 89, 200, 201, 202

PUTRLABEL

statement

```
PUTRLABEL column coordinate, row coordinate, 'repeating label'
```

Puts the *repeating label* into the designated cell. Only one copy of the *repeating label* is stored in the cell because the number of times to repeat the label varies depending on the column width. It is the responsibility of the DUMP function to display the correct number of repetitions of a repeating label.

Errors: 16, 42, 200, 201, 202

PUTSETUP

statement

```
PUTSETUP ['setup string']
```

Puts the printer *setup string* into the active worksheet. If the optional string is not present, removes the *setup string* from the worksheet.

Errors: 16, 42, 200

PUTSTATUS

statement

```
PUTSTATUS status number, value
```

Sets the status item designated by the *status number* to the designated *value*. Refer to GETSTATUS for the meanings and possible values of the status numbers.

Errors: 89, 200, 201, 202

PUTVALUE

statement

```
PUTVALUE column coordinate, row coordinate, value string
```

Puts the value into the designated cell. Note that the value is input as a string, so that ERROR or NA can be input to the worksheet with this keyword. If a value is entered, it is given type 1 (see the CELLTYPE function). ERROR and NA are given type 2, since they are stored internally as formulas.

Errors: 16, 89, 200, 201, 202

PUTWIDTH

statement

PUTWIDTH *column coordinate* [, *width*]

Puts the *width* into the designated column. If the optional *width* is not present, resets the width of the designated column to the global column width.

Errors: 16, 89, 200, 201

RCLVAR

statement

RCLVAR

Recalls the values for 19 BASIC integer variables from VisiCalc's scratch space and places them back into the variables. Also turns off the PRGM annunciator.

- STOVAR must be used before using RCLVAR.
- If RCLVAR is used from the keyboard, unpredictable results may occur.
- RCLVAR blindly copies the contents of VisiCalc's scratch space into the first 19 variables of the BASIC program, whether those variables are integers or not. If the first 19 variables are not integers, unpredictable results may occur.

Errors: 200, 202

RECALC

statement

RECALC [*column coordinate*, *row coordinate*]

Calculates the value of the designated cell and puts the new value there. If the optional column and row coordinates are not present, calculates the entire worksheet (by columns or by rows, depending on the order of recalculation).

- Refer to "Warnings When Formulas Are Calculated" in appendix A.

Errors: 16, 200, 201, 202

REPLICATE

statement

```
REPLICATE
```

Performs replication. Copies the source range defined by the SOURCE statement to the target range defined by the TARGET statement. If the cells copied contain formulas, makes marked cell references relative, calculates the new value of the formula, and puts the formula with its result into the new cell. Also copies cell formats. Unmarks marked cell references in the source range when done.

- SOURCE, TARGET, and MARK must all have been used to set up the cells being replicated.
- Refer to "Warnings When Formulas Are Calculated" in appendix A.

Errors: 16, 200, 202

ROW

function

```
ROW(coordinate)
```

Returns the number of the row coordinate that the *coordinate* represents—the inverse of COORD$. Accepts coordinates with default or user-defined headers. Trailing blanks are ignored, and brackets are required for user-defined headers. Returns 0 if the *coordinate* is a column header, the header is not present, or the *coordinate* is an illegal syntax or an external coordinate. Therefore, a particular coordinate does not exist if both COL and ROW return 0.

Errors: 200

SETWIN

statement

```
SETWIN [start position [, end position]]
```

Specifies the portion of the LCD that will be the window into the input buffer. Normally, the LCD is a 32-character window into the 96-character input buffer. SETWIN allows you to specify a smaller area, anywhere on the LCD, as the window into that same 96-character input buffer. The LCD window is set to be from the *start position* to the *end position*. SETWIN with no parameters is equivalent to SETWIN 1,32. SETWIN with one parameter is equivalent to SETWIN *start position*,32.

SETWIN allows a BASIC program to "freeze" a prompt on the LCD and still accept a full 96-character input. In the example used in the description of INLINE$, this would be done with the following commands:

Command	Action
SETWIN	Sets the LCD window to 1,32.
DISP 'Mar^TotalExp(V): ';	Displays the prompt to be frozen on the LCD. The semicolon is needed so that the next item sent to the LCD will not erase the fixed prompt.
SETWIN 18	Sets the LCD window to 18,32, since the prompt itself occupies positions 1-17.

Now all input and editing will be done in the 15-character window between character positions 18 and 32 on the LCD.

- When working with windows of different sizes, WIDTH INF is advised to avoid conflicts between the window size and width.
- Cursor addressing using ESC % is always relative to the start of the window.

Errors: 89

SOURCE

statement

SOURCE *column 1*, *row 1*, *column 2*, *row 2*

Specifies the source range for the REPLICATE statement. *Column 1* and *row 1* are for the first half of the range, and *column 2* and *row 2* are for the second half of the range.

- *Column 1* must be less than or equal to *column 2*, *row 1* must be less than or equal to *row 2*, and the range must be legal (only a column or row).

Errors: 200, 201, 202

SPEW

statement

SPEW [*column coordinate*, *row coordinate*]

Displays the internal form of the formula at the designated cell. If the optional column and row coordinates are not present, displays the internal form of the formula in VisiCalc's scratch space. Each byte of the formula is displayed in decimal form, separated by colons, with the first two bytes representing the formula length. SPEW is strictly a diagnostic tool.

Errors: 16, 200, 201

STOVAR

statement

```
STOVAR
```

Stores the values for 19 BASIC integer variables from the current program into VisiCalc's scratch space.

- If STOVAR is used from the keyboard, unpredictable results may occur.
- STOVAR blindly copies the values of the first 19 variables of the BASIC program into VisiCalc's scratch space, whether those variables are integers or not. If the first 19 variables are not integers, unpredictable results may occur.

Errors: 200, 202

SUM

function

```
SUM(list)
```

Returns the sum of the items in the *list* when used in a VisiCalc formula. Cannot be used anywhere except in a VisiCalc formula. Because SUM computes using internal 16-digit math routines, it will yield a more precise result than if you compute the sum manually.

Errors: none

TARGET

statement

```
TARGET column 1,row 1,column 2,row 2
```

Specifies the target range for the REPLICATE statement. *Column 1* and *row 1* are for the first half of the range, and *column 2* and *row 2* are for the second half of the range.

- *Column 1* must be less than or equal to *column 2*, *row 1* must be less than or equal to *row 2*, and the range must be legal (only a column or row).

Errors: 200, 201, 202

UNMARK

statement

```
UNMARK
```

Unmarks all marked cell references in the source range.

- Must be used if MARK is used and no replication is performed.
- SOURCE must be used before using UNMARK.

Errors: 200, 202

UPCOL

function

```
UPCOL(current column, direction, specifier)
```

Computes the left and right boundaries of the worksheet to be displayed or printed, and returns this boundary as a number from 0 to 255. The local column widths are added together starting at the *current column* until the first column is found that will not fit within VWIDTH−1 or PWIDTH, depending on the specifier (−1 = video, 0 = printer without headers, 1 = printer with headers, just like DUMP). On the video, header widths and column widths that are too wide to fit are ignored. On the printer, header widths too wide to fit are ignored, but at least one column will be included in the boundary to be printed, even if the column is greater than PWIDTH.

The *direction* indicates which direction to sum the local column widths. If the right boundary is to be found (the *current column* is the left boundary to be displayed or printed), the *direction* is 1. If the left boundary is to be found (the *current column* is the right boundary to be displayed or printed), the *direction* is −1.

- If the *direction* is −1, and there are no columns to the left of the *current column* that will fit on the video, the first left boundary found may be to the right of the *current column*.

Errors: 200

UPROW

function

```
UPROW(current row)
```

Computes the lower boundary of the worksheet to be displayed, and returns this boundary as a number from 1 to 255. The *current row* is added to VLENGTH and adjusted for the rows of default and user-defined headers at the top of the video.

- If the result of UPROW is greater than 256, the screen is cleared, and 255 is returned.

Errors: 200

VALCHK

function

```
VALCHK('string')
```

Returns the numeric value of a *string* composed of digits, decimal point, and/or exponent. If the numeric value is between 0 and 255, inclusive, returns the number. If not, beeps and returns −1.

Errors: none

VC

statement

```
VC [specifier]
```

Executes VisiCalc. The optional *specifier* is a number that causes certain normal start-up procedures to be bypassed, as follows:

Specifier	Start-up Procedures Bypassed
0	Copyright notice
1	Copyright notice and worksheet catalog
2	Copyright notice, worksheet catalog, and calculation of cell zero

Specifiers greater than 0 assume that an active worksheet already exists (i.e., the WORKSHEET statement has been executed).

Certain HP-75 global conditions are altered while VisiCalc is executing, as follows:

Condition	Value During VisiCalc
DELAY	DELAY 0
DEFAULT OFF/ON	DEFAULT OFF
DISPLAY IS device	PRINTER IS device
MARGIN	various
PRINTER IS device	not used unless printing
TRACE FLOW/VARS	TRACE OFF
WIDTH	WIDTH INF
Window start	various
Window end	various

All these conditions are saved when VisiCalc starts, changed to what VisiCalc needs while it is executing, and then restored to their original states when VisiCalc exits.

When an extension function is executed, the original states are restored, so the extension function sees the same configuration of the HP-75 as before VisiCalc was run. Upon return from the extension function, VisiCalc will ignore any changes made by the extension function to these global conditions. When VisiCalc exits, the original states saved when VisiCalc was run will be restored, even if these were changed by the extension function.

The only exception to this is for the HP-IL assignments. When VisiCalc starts, the names of the user's DISPLAY IS and PRINTER IS devices are saved. Then the user's DISPLAY IS device is made into VisiCalc's PRINTER IS device. This is done to allow separating the behavior of the video from the behavior of the LCD.

The user's DISPLAY IS and PRINTER IS devices are restored when an extension function is called. However, since an extension function could change the HP-IL configuration, it may not be possible to restore the DISPLAY IS and PRINTER IS assignments to their original values. Therefore, if an extension function changes the HP-IL assignments, the changed assignments will be restored when VisiCalc exits.

The original global conditions are only restored when an extension function is called or when VisiCalc exits. When a visi-commands program is called, it will inherit the same global conditions that are active during VisiCalc. If the program changes any of these conditions, it should restore them to what VisiCalc expects before returning control back to VisiCalc.

- If VisiCalc is invoked programmatically, any timers in effect in the program containing the VC statement will not be honored while VisiCalc is running.

Errors: 16, 202, same errors as HP-75 CALL statement.

VLENGTH

statement

VLENGTH *length*

Sets the video length (the number of lines on the video) to the designated *length*.

- VLENGTH less than 2 may behave unpredictably.

Errors: 16, 89

VWIDTH

statement

```
VWIDTH width
```

Sets the video width (the number of columns on the video) to the designated *width*.

- VWIDTH less than 2 may behave unpredictably.

Errors: 16, 89

WAITKEY$

function

```
WAITKEY$(wait time)
```

Halts program execution until a key is pressed or until the specified *wait time* expires, whichever occurs first. Returns the ASCII character corresponding to the key pressed. If no key is pressed before the *wait time* expires, returns a null string. If the *wait time* is negative, waits forever until a key is pressed. Unlike KEY$, WAITKEY$ will intercept the [ATTN] key.

Errors: none

WINSIZ

function

```
WINSIZ
```

Returns the size of the current window, as specified by SETWIN, where size = (end position − start position + 1).

Errors: none

WORKSHEET

statement

```
WORKSHEET ['file name']
```

Specifies which file will be the active worksheet file, and timestamps it. If the file does not exist, creates it. If the optional *file name* is not present, sets the active worksheet name to null. If the VisiCalc module has just been plugged into the HP-75, reserves some scratch space for use by VisiCalc if there is room.

When a new worksheet is created, its status is initialized as follows:

Status	Initial Value
Recalculation mode	automatic
Recalc order	by columns
Global format	/GFG
Global column width	7
Row header width	5
Current cell position	A1
Upper-left corner of video	A1
Window access cell	A1
Formula/Result Display mode	result
VisiCalc level	exit (3)
Suppress default headers	no
Coordinate Display mode for cell contents field	user-defined header
Coordinate Display mode for cell coordinates field	user-defined header
Display format mode	formatted
Coordinate display specifier	user-defined headers with no brackets (type 2)
Video active	yes

Errors: 16, 63, 68

Section 3

VisiCalc® Extensions

Extending the VisiCalc® Command Set

The spreadsheet language enhancements of the VisiCalc module provide the capability for the user to access worksheets using BASIC commands from the keyboard or in a program. For example, assume that a clear worksheet feature is needed that will clear out all cells but keep any user-defined headers intact. The program might look like this:

```
10 FOR X=1 TO MAXCOL
20 FOR Y=1 TO MAXROW
30 BLANK X,Y @ PUTFORMAT X,Y,0
40 NEXT Y
50 NEXT X
```

As written, the user would execute this program outside of VisiCalc. If the file name of this program were CLEARWS, and the worksheet to be cleared were named SAMPLE, the user would type from the keyboard:

```
WORKSHEET 'SAMPLE' @ RUN 'CLEARWS'
```

and then enter VisiCalc to observe the result.

A multitude of additional features can be created using the spreadsheet language enhancements: clearing data only, numeric sorting, alphabetizing, translating worksheets to /SS or DIF file types, merging data from other worksheets; the number of possibilities is enormous.

As a feature becomes more complex it becomes desirable to execute it while in VisiCalc. In the above example only the worksheet is specified, but for a numeric sort the user may need to specify information about the columns and rows of the worksheet as well. This level of prompting is better done if the user is in VisiCalc and viewing the active worksheet.

To accommodate this need, the capability exists for integrating additional features into the VisiCalc command set so that they may be accessed through the command key /. The procedure for adding commands to VisiCalc involves two steps: creating new messages in the text file VISIMSGS, and developing a BASIC program (visi-commands) to perform the desired operations.

Creating New Messages

In the VisiCalc module, the main command prompt is made by combining messages 21 (Command) and 22 (DFGHIMPRVW-). Message 21 is the level prompt and message 22 is the choice prompt. This combination yields the prompt:

Command: DFGHIMPRVW-

when the / key is pressed from Top level of VisiCalc. Any additions to the command set must be reflected in message 22.

For example, suppose that several command extensions are envisioned for VisiCalc, and that the user enters this command level by pressing the key sequence / E. Altering message 22 is done by editing the text file VISIMSGS (for visi-messages) and entering DFGHIMPRVW-E into line 22. (To create an underlined character, first press CTL I/R and then the desired character.)

Now when the / key is pressed from Top level of VisiCalc, the altered command prompt appears:

Command: DFGHIMPRVW-E

When E is pressed by the user to access the extended command set, VisiCalc searches in user memory or plug-in modules for a BASIC program named VISICM1 (visi-commands one). If the file VISICM1 does not exist, VisiCalc exits and a file not found error is issued.

The text file VISIMSGS is used to store the additional command prompts that are accessed by VISICM1. In building the VISIMSGS text file, the following guidelines should be observed to be consistent with VisiCalc:

- The level and choice prompts should be separate messages.

  Example: Command and DFGHIMPRVW- are separate messages.

- Capitalization should occur on the first character of a level prompt, and on the first character of an input prompt. Only one character of each choice in the choice prompt should be underlined, and is usually capitalized.

  Examples: level prompt—Headers
  choice prompt—Col Row Width Suppress
  input prompt—Width row headers

The colon (:) that appears between a level prompt and a choice prompt, or after an input prompt, is displayed automatically by VisiCalc, and should not be included in the messages.

Continuing the example, assume that the extended command set is to include the additions Clear and Sort. To support these additions, the prompts are composed of messages from the VisiCalc module (message 23, Yes No, and message 25, Col Row), and additions to the text file VISIMSGS. A list of the VISIMSGS text for these additions would appear as follows:

Line	Prompt Type	Display Prompt
22 DFGHIMPRVW-E	choice	Command: DFGHIMPRVW-E
70 Extra command	level	
71 Clear Sort	choice	Extra command: Clear Sort
72 Clear	level	
73 Worksheet Data	choice	Clear: Worksheet Data
74 Clear worksheet	level	Clear worksheet: Yes No
75 Clear data	level	Clear data: Yes No
90 Sort	level	Sort: Col Row
91 Numeric Alphabetic	choice	Sort: Numeric Alphabetic
92 Ascending Descending	choice	Sort: Ascending Descending
93 Select col	input	Select col: *default*
94 Select row	input	Select row: *default*
95 First col	input	First col: *default*
96 First row	input	First row: *default*
97 Last col	input	Last col: *default*
98 Last row	input	Last row: *default*

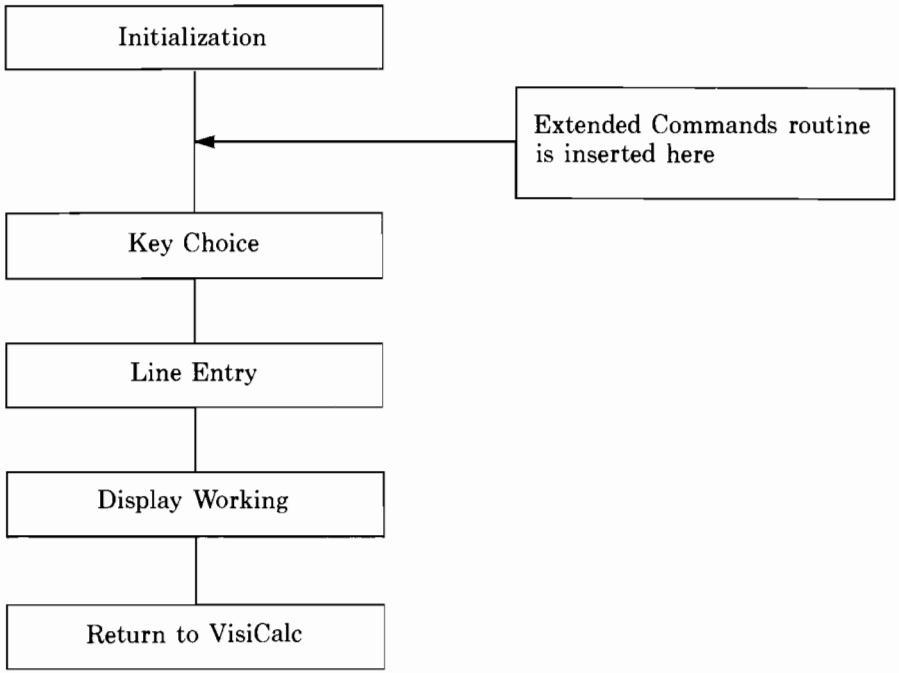
These messages are provided on the VISIMSGS magnetic card.

The Visi-Commands Shell

The BASIC program implementing the extended command set is named VISICM1. Generally, this is a program that utilizes the language enhancements of the VisiCalc module to interact with worksheets. As with the message file, a number of guidelines must be adhered to when developing the visi-commands program.

A universal shell is required so that the user-developed visi-commands program can successfully interact with the VisiCalc module. This shell is a set of five BASIC program routines: Initialization, Key Choice, Line Entry, Display Working, and Return to VisiCalc; the visi-commands program is inserted into this shell. Once the shell has been created, the BASIC code for the VisiCalc command set is inserted between the Initialization and Key Choice routines.

Visi-Commands Program Structure



Together these routines define a visi-commands program. This program is excuted out of user memory or a plug-in module. If in a plug-in module, visi-commands must be the first non-LEX file in the module. The text file VISIMSGS must always be in user memory.

The following sections describe the functions of each routine in the shell.

Initialization Routine. This routine must be the first part of the program.

```
10 INTEGER P,C,R,F5,V5,V0,F1,X0,Y0,X1,Y1,Z0,Z1,Z2,S1,S4,S5,S6,S7
20 RCLVAR
30 T2$=CHR$(13)&Δ_°αβ$%!÷Σσ")■-I @ M2=0
40 DIM B0$[96]
```

Line 10 declares a list of integer variables whose values are passed to the visi-commands program from VisiCalc. The INTEGER statement must be the first statement of the visi-commands program, and it must contain exactly 19 variable names. The definition of each of these variables is listed below. For the variables that represent status items in the worksheet, the status number is shown in parentheses. Note that if any of these variables are changed, the corresponding status location must be changed also, and vice versa.

Variable	Meaning
P	Flag used to indicate if [ATTN] (P>0) or [SHIFT] [ATTN] (P=0) exits visi-commands.
C	Column coordinate of current cell (status 5).
R	Row coordinate of current cell (status 6).
F5	For VisiCalc internal use.
V5	Video update. 0 = update video if necessary 1 = update video 2 = do not update video 3 = update current cell only
V0	Video active flag whose value is determined in VisiCalc by the statement: V0=GETSTATUS(18)*LEN(DSP$)>0 0 = inactive 1 = active
F1	Cell type of current cell. 1 = value 2 = formula 3 = formula containing external references 4 = label 5 = repeating label 6 = blank
X0	Column coordinate of upper-left corner of video (status 7).
Y0	Row coordinate of upper-left corner of video (status 8).
X1	Column coordinate of lower-right corner of video.
Y1	Row coordinate of lower-right corner of video.
Z0	Flag indicating current viewing window. 0 = primary 1 = alternate
Z	Column coordinate of current cell of primary viewing window.
Z2	Row coordinate of current cell of primary viewing window.
S1	Formula/Result Display mode (status 11). 0 = result 1 = formula
S4	Coordinate Display mode for cell contents field of cell display (status 14). 0 = default headers 1 = user-defined headers

Variable	Meaning
S5	Coordinate Display mode for cell coordinates field of cell display (status 15). 0 = default headers 1 = user-defined headers
S6	Display format mode (status 16). 0 = formatted 1 = unformatted
S7	Coordinate display specifier (status 17). 0 = default headers 1 = user-defined headers 2 = user-defined headers with no brackets 3 = user-defined headers with nulls 4 = user-defined headers with nulls and no brackets (type 2 and 3 combined) 5 = type 4 with a space instead of a caret

Line 20 passes the values of the variables declared in line 10 to the visi-commands program.

Line 30 defines the string T2$ to contain the keys that terminate a line of input and initializes the message pointer M2 to zero. The terminator key string is used in the Line Entry subroutine to branch to appropriate places in the visi-commands program, and to screen out any HP-75 terminator keys that would halt a BASIC program during input. Below is a complete list of the terminator keys:

Position in T2$	Character	Key	ASCII Value
1	none	RTN	13
2		ATTN	128
3		SHIFT ATTN	160
4		TIME	129
5		↑	132
6		↓	133
7		SHIFT ↑	164
8		SHIFT ↓	165
9		SHIFT TIME	161
10		EDIT	131
11		APPT	130
12		FET	137
13		SHIFT APPT	162
14		SHIFT FET	169
15		RUN	141
16		SHIFT RUN	173
17		CTL FET	201

Line 40 dimensions the input buffer string to 96 characters.

Key Choice Subroutine: This subroutine uses WAITKEY$ to handle all single key responses to a choice prompt.

```
8795 IF NOT M2 THEN M2=M1+1
8796 SETWIN @ DISP MESSAGE$(M1);": ";MESSAGE$(M2)
     @ T$=UPRC$(CHOICE$(M2)) @ M2=0
8810 K$=WAITKEY$(-1) @ K=POS(T$,UPRC$(K$)) @ IF K THEN RETURN
8830 IF POS('_ ',K$) THEN POP ELSE 8810
8835 IF K$='_' THEN P=0
8852 GOTO 9990
```

Line	Description
8795	If the command level prompt and the choice prompt have consecutive message numbers, then M2 (choice prompt message number) is set to one greater than M1 (level prompt message number).
8796	Set the LCD window to 1,32. Display messages M1 and M2 separated by a colon. Set T$ to the choices of message M2. Reset M2 to zero.
8810	If a valid key choice is found then return.
8830	If [ATTN] or [SHIFT] [ATTN] are not pressed then branch and wait for another key to be pressed. Otherwise POP the return from the GOSUB that called this subroutine.
8835	If [SHIFT] [ATTN] is pressed then set P=0.
8852	Jump to Return to VisiCalc subroutine.

Line Entry Subroutine: This subroutine accepts line input using INLINE$.

```
9115 SETWIN @ DISP M$;": "; @ CLRLCD LEN(M$)+3
9121 W0,P0=1 @ F0=0
9123 SETWIN LEN(M$)+3
9125 K$=INLINE$(B0$,W0,P0,T2$,F0)
9130 B0$=INPBUF$ @ W0=INPWIN @ P0=INPPTR @ F0=IRFLAG
     @ L=LEN(B0$)+1
9180 K=POS(T2$,K$) @ IF K>3 THEN 9123
9210 IF K=1 THEN RETURN
9220 IF K=2 THEN POP @ GOTO 9990
9230 IF K=3 THEN POP @ P=0 @ GOTO 9990
9231 ! Insert other intercepts here and alter line 9180.
9399 RETURN
```

Line	Description
9115	Set the LCD window to 1,32. Display the input message M$. Clear the LCD from length (M$+3) to 32.
9121	Initialize variables: W0 = position of first displayed character of input buffer P0 = cursor position in input buffer F0 = cursor type: 0 = replace cursor, 1 = insert cursor
9123	Set the LCD window from length (M$+3) to 32.
9125	Accept keyboard input until a terminator key from string T2$ is pressed, and assign this key to K$.
9130	Preserve input buffer parameters: B0$ = contents W0 = position of first displayed character P0 = cursor position F0 = cursor type L = length of contents + 1
9180	Find the position of the terminator key K$ in terminator string T2$. If the key is not [RTN], [ATTN], or [SHIFT] [ATTN], then continue accepting keyboard input.
9210	Return if [RTN] was pressed.
9220	Exit if [ATTN] was pressed.
9230	Set P=0 and exit if [SHIFT] [ATTN] was pressed.
9399	Return for other possible intercepts.

Display Working Subroutine: This subroutine displays VisiCalc message 2 and turns off the cell cursor on the video.

```
9982 IF V0 THEN CURSOFF
9983 SETWIN @ DISP MESSAGE$(2) @ RETURN
```

Line	Description
9982	If a video is active, turn off the cell cursor.
9983	Set the LCD window to 1,32. Display message 2 (Working) and return.

Return to VisiCalc Routine: This routine is the exit from the visi-commands program.

```
9990 STOVAR @ END
```

This line passes the values of those variables declared in the INTEGER statement on line 10 back to VisiCalc.

Creating a Visi-Commands BASIC Program

Single Visi-Commands Program: The user-developed Extended Commands routine consists of BASIC program lines developed around the spreadsheet language keywords contained in the VisiCalc module. By utilizing keywords to access and modify the data and parameters of worksheets, the [/] command set of VisiCalc is extended. Returning to the example in "Creating New Messages," assume that an extended command set includes Clear and Sort. The Extended Commands routine listed below utilizes the message additions outlined in that section.

The Clear command is designed to clear a worksheet via two options: clear all cell entries and their local formats, or clear only cells containing values or formulas with no cell references. In both cases the user-defined headers remain intact.

With VISIMSGS in user memory, the VisiCalc user presses [/] from Top level to access the Clear command:

```
Command: DFGHIMPRVW-E
```

Note that message 22 in VISIMSGS replaced message 22 in the VisiCalc module.

Pressing [E] causes VisiCalc to call the BASIC program VISICM1. Falling through the Initialization portion of the visi-commands program (lines 10-40), the first line of the Extended Commands routine is executed.

```
500 M1=70 @ GOSUB 8795 @ ON K GOTO 1000,1610
```

Using messages 70 and 71 from VISIMSGS, the display shows:

```
Extra command: Clear Sort
```

By pressing [C] or [S] the user may access each of the additional commands at lines 1000 or 1610. Pressing [ATTN] will return to VisiCalc. Pressing [SHIFT][ATTN] will exit VisiCalc. Note that when the HP-75 turns itself off, it does so by "pressing" the [SHIFT][ATTN] key.

The Clear command code is as follows:

```
1000 M1=72 @ GOSUB 8795 @ ON K GOTO 1020,1050
1010 ! Clear worksheet.
1020 M1=74 @ M2=23 @ GOSUB 8795
     @ IF K=2 THEN 9990 ELSE GOSUB 9982
1030 FOR X=1 TO MAXCOL @ FOR Y=1 TO MAXROW
1035 BLANK X,Y @ PUTFORMAT X,Y,0
1038 NEXT Y @ NEXT X
1040 V5=1 @ GOTO 9990
1049 ! Clear data.
1050 M1=75 @ M2=24 @ GOSUB 8795
     @ IF K=2 THEN 9990 ELSE GOSUB 9982
1060 FOR X=1 TO MAXCOL @ FOR Y=1 TO MAXROW
1065 T=CELLTYPE(X,Y) @ IF T>2 THEN 1080
1070 IF T=1 THEN BLANK X,Y @ GOTO 1080
1075 IF NOT LEN(FIRSTREF$(X,Y,0)) THEN BLANK X,Y
1080 NEXT Y @ NEXT X
1085 IF NOT GETSTATUS(1) THEN RECALC
1090 V5=1 @ GOTO 9990
```

Line	Description
1000	Displays the prompt Clear: Worksheet Data and records the choice.
	Clear worksheet
1020	Displays the prompt Clear worksheet: Yes No. Y calls Display Working subroutine, N returns to VisiCalc.
1030	Steps through each cell of the worksheet.
1035	Clears each cell and clears the local format.
1038	End of loop.
1040	Force the video to update upon return to VisiCalc.

Line	Description
	Clear data
1050	Displays the prompt `Clear data: Yes No. Y` calls Display Working subroutine, `N` returns to VisiCalc.
1060	Steps through each cell of the worksheet.
1065	Determine the cell type. Take no action if the cell contains a formula with external references (type 3), a label (type 4), a repeating label (type 5), or is blank (type 6).
1070	Clear the cell if it contains a value.
1075	Clear the cell if it contains a formula with no cell references.
1080	End of loop.
1085	Recalculate the worksheet if Automatic Recalculation mode is set.
1090	Force the video to update upon return to VisiCalc.

As with the Clear command extension, the Sort code utilizes the spreadsheet language and the visi-commands program modules. Appendix B gives a listing of the completed visi-commands program `VISICM1` including code for the Sort routines. This program, with the comments removed, is included on magnetic cards, as is the `VISIMSGS` file with the required messages.

Multiple Visi-Commands Programs: In the event that more than one visi-commands programs exists in user memory and/or plug-in module simultaneously, VisiCalc can access any of the programs. To reflect the existence of more than one visi-commands program, message 22 in the text file `VISIMSGS` must reflect the additions. This is done by adding choices to message 22. For example,

```
Command: DEGHIMPRVW-XYZ
```

indicates that three different visi-commands programs exist. Given their order in the command prompt, pressing `X` calls `VISICM1`, `Y` calls `VISICM2`, and `Z` calls `VISICM3`. The naming convention for a visi-commands program is `VISICM`*n* where *n* is 1-99, corresponding to the position of the nth choice after the `-` (repeating label) choice in message 22.

If several such programs exist and *n* is not consecutive, then the underlined block character (underlined CTL 8, ▒) is used to fill in the gaps. For example, assume visi-commands programs 2, 5, 6, and 8 exist in user memory. Then message 22 would be modified to produce the following command prompt:

```
Command: DEGHIMPRVW-▒A▒▒BX▒Y
```

Notice that numbers could be used instead of letters, so that the numbers imply which visi-commands program will be selected. The above example would then look like this:

```
Command: DEGHIMPRVW-▒2▒▒56▒8
```

In situations involving more than one visi-commands file, care must be taken to insure that the appropriate messages in VISIMSGS are present for use by VISICM*n*. One possible solution allows each visi-commands program to copy its own messages into VISIMSGS when it is called. Each visi-commands program would include the following line in its Initialization routine:

```
50 RENAME "VISIMSGS" TO "TEMPMSGS"
   @ RENAME "VISIMS1" TO "VISIMSGS"
```

and would replace its Return to VisiCalc routine with the following:

```
9990 RENAME "VISIMSGS" TO "VISIMS1"
     @ RENAME "TEMPMSGS" TO "VISIMSGS"
9995 STOVAR @ END
```

Using Coordinate Display Specifiers

When using VisiCalc, the user can modify the display by pressing the primary and shifted [TIME] and [APPT] keys:

Key	Action
[TIME]	Toggles between default and user-defined headers for cell coordinates.
[SHIFT] [TIME]	Toggles between default and user-defined headers for cell contents.
[APPT]	Toggles between Formula and Result Display modes.
[SHIFT] [APPT]	Toggles between Formatted and Unformatted Display modes.

In order for a visi-commands program to reflect the settings of these toggles when returning coordinates and cell contents, the program must use the proper specifier for the keywords COORD$, DECOMP$, GETVALUE$, GETFORMULA$, and FIRSTREF$. The variables S4, S5, S6, and S7 reflect the values of worksheet status 14, 15, 16, and 17, respectively, and should be used as the coordinate specifier as shown below:

Keyword	Specifier	Example
COORD$	S5*S7	COORD$(*column*, *row*, S5*S7)
DECOMP$	S4	DECOMP$(S4)
GETVALUE$	S6	GETVALUE$(*column*, *row*, S6)
GETFORMULA$	S4	GETFORMULA$(*column*, *row*, S4)
FIRSTREF$	S4	FIRSTREF$(*column*, *row*, S4)

The visi-commands program listing for Sort (beginning on page 69) demonstrates the use of the correct specifiers.

Debugging Visi-Commands

The developing and debugging of a visi-commands program requires a more careful approach and some different techniques than those used in developing most BASIC programs.

Using the [ATTN] Key: An important difference between visi-commands programs and other BASIC programs is the inability to halt visi-commands at any point with the [ATTN] key. Pressing [ATTN] is used to return to Top level in VisiCalc from the Key Choice and Line Entry subroutines of visi-commands. As a result, conditions that would cause visi-commands to become lost in an infinite loop need to be avoided—the only recovery is a system reset of the HP-75 using [SHIFT][CTL][CLR]. It is therefore recommended that files in user memory be backed up on mass storage before experimenting with visi-commands.

Halting the Program: A STOP statement is not recognized by VisiCalc as it executes a visi-commands program. Therefore, other techniques are required to halt the program and examine the current values of variables. One such method is to insert the statements SETWIN, DISP *variable list*, and WAITKEY$ where a STOP statement would be placed.

For example, the following line halts visi-commands and displays the variables X, Y, and T in the LCD until any key is pressed:

```
1075 SETWIN @ DISP 'X=';X;'Y=';Y;'T=';T @ Z$=WAITKEY$(-1)
```

If output to the video is desired, the video must be specified as the display device prior to displaying the variables, and then the video must be specified as the printer device afterwards. The following example uses the keyword DSP$ to return the name of the display device from VisiCalc:

```
1074 IF NOT LEN(DSP$) THEN DISPLAY IS DSP$
1075 SETWIN @ DISP 'X=';X;'Y=';Y;'T=';T @ Z$=WAITKEY$(-1)
1076 IF NOT LEN(DSP$) THEN PRINTER IS DSP$
```

Examining Program Errors: If a run-time error is encountered as VisiCalc executes the visi-commands program, an error message is displayed, the program is halted, and an exit from VisiCalc occurs. To examine the line number in visi-commands where the error occurred, type ERRL from the keyboard. The error number can be returned by typing ERRN.

Cell Zero

Cell zero is the cell in every VisiCalc worksheet whose column coordinate and row coordinate are both zero. This cell cannot be examined or modified from VisiCalc, but can be accessed using spreadsheet language commands. In this respect, cell zero is the same as any other cell. The following keywords can be used to examine and modify cell zero: BLANK, CELLTYPE, FIRSTREF$, GETFORMAT, GETFORMULA$, GETLABEL$, GETVALUE$, NEXTREF$, PUTFORMAT, PUTFORMULA, PUTLABEL, PUTRLABEL, PUTVALUE, and RECALC.

Although labels, repeating labels, values, and formats can be put in cell zero, it is most useful when it contains a formula with a call to an extension function. Cell zero is recalculated at three important times while VisiCalc is executing. If it contains a formula with a call to an extension function, that extension function will be executed at those times. This allows an extension function to take control away from VisiCalc, use the spreadsheet keywords to examine and modify the worksheet, and then return control back to VisiCalc.

Cell zero is only recalculated at start-up, cell entry, and when exiting VisiCalc. Start-up occurs after the worksheet has been selected, but before reaching Top level (after VC, VC 0, or VC 1). Cell entry occurs after the cell contents are changed because of value, label, or repeating label entry, or by clearing a cell. Exiting VisiCalc occurs after [SHIFT] [ATTN] has been pressed (or when the HP-75 turns itself off), but before returning to HP-75 EDIT mode.

Cell zero is not recalculated when the entire worksheet is recalculated, and VC 2 specifically bypasses the copyright notice, worksheet catalog, and calculation of cell zero.

Status 12 allows the extension function called by cell zero to identify which of the three conditions caused it to be executed:

Condition	Status 12
Start-up	1
Cell Entry	2
Exit	3

For example, suppose that the following extension function is used to intercept VisiCalc via cell zero:

```
10 ON GETSTATUS(12) GOTO 20,30,40
20 DISP "Start of VisiCalc" @ END
30 CLRLCD @ DISP "Cell entry at ";COORD$(GETSTATUS(5),
   GETSTATUS(6),GETSTATUS(15)*GETSTATUS(17))
35 WAIT 1 @ DISP "Cell type"; CELLTYPE(GETSTATUS(5),
   GETSTATUS(6)) @ END
40 DISP "End of VisiCalc" @ WAIT 1 @ END
```

Line 10 branches to the appropriate operation based on status 12. At start-up, line 20 is executed to indicate that VisiCalc has started. After cell entry, lines 30 and 35 are executed to display the location of the modified cell and describe its contents. Line 40 is executed when VisiCalc is about to exit.

Notice that the extension function does not put a value in line 0 of VISIDATA. This is not necessary in cell zero, since it cannot be examined or referenced while running VisiCalc. Since no result is placed in VISIDATA, cell zero's value will be ERROR (use GETVALUE(0,0,1) to verify this).

Status 5 and status 6 give the current column and row position of the cell being modified. After cell entry, those coordinates are used by COORD$ to display the VisiCalc coordinate of the cell being changed. The third parameter for COORD$ is the product of status 15 and status 17. This insures that the coordinate will be displayed either using default headers or using headers specified by status 17 (the coordinate display specifier).

PUTFORMULA is used to enter a formula into cell zero. Assuming a file name INTRCEPT for this program, the command sequence for entering the formula into cell zero of worksheet SAMPLE is:

```
WORKSHEET 'SAMPLE' @ PUTFORMULA 0,0,'INTRCEPT'
```

At start-up, the message Start of VisiCalc will be displayed. After cell entry, Cell entry at and the VisiCalc coordinate will be displayed, followed by the type of the cell after modification. Just before VisiCalc exits, End of VisiCalc will be displayed.

If the extension function changes the worksheet status (with PUTSTATUS), VisiCalc must be signalled to read the new status (this is only necessary after cell entry). This signal is sent via status 12. If the extension function sets status 12 to 0 or 1, VisiCalc will read the current worksheet status.

The value of 0 or 1 for status 12 tells VisiCalc how to update the video if one is present. If status 12 is set to 0, the video will be updated only if necessary. Therefore, 0 should be used in those instances where the status has changed, but the changed status may not affect the video (status 0-2, 9-11, 14-17). If status 12 is set to 1, the video will always be updated. Therefore, 1 should be used in those instances where the status has changed and does affect the video (status 3-8, 13, 18), or if the status has not changed, but the video still needs to be updated (such as if a column width is changed).

Thus status 12 serves as a two-way communication path for passing information between an extension function called from cell zero and VisiCalc. VisiCalc sends the point at which cell zero is recalculated, and the extension function signals whether or not it changed the worksheet status, and how to update the video if one is present.

In the example program, messages will be displayed in the LCD the three times that cell zero is recalculated. If a video is being used, the messages will be displayed starting at the upper-left corner of the video. In most instances, this will overwrite a portion of the displayed worksheet. The display can be refreshed manually by the user with the /VY command, or automatically upon return from the extension function by setting status 12 to 1. The following additions to lines 30-41 will clear and update the video:

```
30 CLRSCR
31 CLRLCD @ DISP "Cell entry at ";COORD$(GETSTATUS(5),
   GETSTATUS(6),GETSTATUS(15)*GETSTATUS(17))
35 WAIT 1 @ DISP "Cell type";CELLTYPE(GETSTATUS(5),GETSTATUS(6))
36 PUTSTATUS 12,1 @ END
40 CLRSCR
41 DISP "End of VisiCalc" @ WAIT 1 @ END
```

The practical applications for cell zero include checking the validity of user inputs, moving the cell cursor position based on those inputs, and pre- and post-processing of a worksheet.

Other VisiCalc® Enhancements

Coordinate Display Specifier

The coordinate display specifier defines how headers will be displayed at Top level in VisiCalc. The specifier is 0-5, and the meanings of the different values are illustrated on page 14. When the user presses TIME at Top level, VisiCalc toggles the specifier between 0 (default headers) and the value defined by status 17. If status 17 is changed with `PUTSTATUS 17,n`, that specifier will be used when user-defined headers are displayed at Top level. For example, one effect of `PUTSTATUS 17,5` is to display a space instead of a caret as the separator between user-defined headers. Other uses of status 17 are related to null headers, discussed below. Notice that no matter which coordinate display specifier is in effect, the default headers will always be displayed when TIME is pressed.

Null Headers

Null headers provide the ability to suppress the column and row headers from the cell coordinates displayed at Top level. To enter a null column header, type just a left and right bracket (`[ ]`) as the header when using the /HC command, or execute `PUTLABEL c,0,''` (for column *c*). To enter a null row header, type just a left and right bracket as the header when using the /HR command, or execute `PUTLABEL 0,r,''` (for row *r*).

To display null headers, the coordinate display specifier must be changed to one that allows null headers to be displayed (3, 4, or 5—refer to `COORD$` for more details). Also, the Coordinate Display mode must be set to show user-defined headers. Either press TIME at Top level until the column header disappears, or execute `PUTSTATUS 15,1`.

For example, set the column A header to null, and the coordinate display specifier to 4. From Top level in VisiCalc, the cell coordinates will now be displayed as "line numbers:"

```
  1:
  2:
   .
   .
   .
255:
```

If row headers Income and Rent exist, they will be displayed in a way that looks more like prompts than cell coordinates:

```
Income:
Rent:
```

Application programs that prompt for data and calculate a result can be developed quickly and easily using VisiCalc. The "prompts" would be the row headers, the column header would be suppressed (null), and the coordinate display specifier would be 3, 4, or 5. Since the time-consuming task of developing the user interface is already done, the application program can be completed simply by selecting names for the "prompts" and writing the calculations using VisiCalc functions and extension functions.

Label Entry Message

If message 4, (L), is a null message, VisiCalc will enter a label when any key that starts cell entry is pressed. This prevents the keys that normally start value entry (+, −, ., (, [, or digit) from doing so. The null message is created by placing one space after line 4 in the text file VISIMSGS. Since the (L) message is suppressed, it will not appear when you begin the cell entry. (Note that putting the @ sign as the beginning character of the line still causes the label to be treated as a value.)

This has some possible applications for text entry. If a column is visualized as a page of text, with each row in that column corresponding to a line of text, a worksheet can be treated as a pseudo-text file. A page could have up to 255 lines of up to 95 characters each. All the existing VisiCalc commands and features for line editing, inserting, deleting, moving, clearing, replicating, and printing can be used to enter and edit this page of text, and the video can be an effective editing aid.

If all keys are forced to begin label entry only, text entry is simplified. As described previously, null headers can be used to display "line numbers." Since VisiCalc beeps at the local column width during label entry, an end of margin beep for text entry occurs by setting a local column width.

Because multiple worksheets can be stored in user memory, a separate worksheet should be used for each page of text. If you use each column of the same worksheet as a separate page, it will be very difficult to insert or delete lines on individual pages. However, a visi-commands program could be written to insert, delete, and move individual lines on separate pages.

Running VisiCalc® Programmatically

The VC keyword can be used in a program. This allows another option for applications built around VisiCalc. Programs can modify worksheets before and after VisiCalc is run, thereby tailoring worksheets for certain applications.

For example, a data collection program could prompt for data, and put that data in a worksheet. The program could set up certain items in the worksheet to make it more useful, such as headers, coordinate display specifiers, column and row position, location of window access cell, etc..

Since the worksheet is already known, VC 1 could be used in the program to bypass the copyright notice and worksheet catalog and enter VisiCalc directly (refer to VC for further details on its optional specifier). When the user presses [SHIFT] [ATTN], VisiCalc ends, and control is returned to the calling program. The program can then post-process the worksheet if desired, begin a new data collection session, etc.

Video Active

VisiCalc determines whether or not to display the worksheet on the video by both the presence of a display device and by the user's choice of the video being active or not. Status 18 saves the latter state, allowing an extension function to activate or deactivate the video with PUTSTATUS 18,1 or PUTSTATUS 18,0.

Ending VisiCalc

The ENDVC statement ends VisiCalc, and can be used in an extension function. This gives an extension function, particularly one called from cell zero, a great deal of flexibility. For example, an application could be written such that VisiCalc would end when a certain condition was reached, such as when the last data item was input, or an iterative procedure was completed.

Appendix A

VisiCalc® Errors and Warnings

Error Messages

Number	Message and Condition
16	not enough memory CALLVC, CHOICE$, DECOMP$, DUMP, GETFORMULA$, GETVALUE$, INLINE$, INSCOL, INSROW, INTERP$, MESSAGE$, MOVCOL, MOVROW, PARSE, PUTFORMAT, PUTFORMULA, PUTLABEL, PUTRLABEL, PUTSETUP, PUTVALUE, PUTWIDTH, RECALC, REPLICATE, SPEW, VC, VLENGTH, VWIDTH, WORKSHEET Not enough available memory exists.
42	string too long • CHOICE$, MESSAGE$: The message is longer than 32 characters. • PARSE, PUTFORMULA: The formula is longer than 91 characters. • PUTLABEL, PUTRLABEL, PUTSETUP: The string is longer than 255 characters.
63	invalid filespec WORKSHEET: The file name is invalid.
68	wrong file type WORKSHEET: The file already exists, but is not type W.
89	bad parameter • CLRLCD, SETWIN: The start or end position is less than 1 or greater than 32, or the end position is less than the start position. • DECOMP$, INTERP$, PUTFORMULA with no optional formula: The PARSE keyword is not used previously or is used on an invalid formula. • DIR$, INCAT: The file type is not a single character. • GETSTATUS: The status number is greater than 19. • INLINE$: The window start is less than 1 or greater than 96, the cursor position is less than the window start or greater than 96, the cursor position is greater than the input string length+1, or the cursor type is not 0 or 1.

Number	Message and Condition
	• INSCOL: The column 255 already exists, or there is a local column width defined for column 255. • INSROW: Row 255 already exists. • PUTFORMAT: The format is not 0-6. • PUTLABEL: There is an illegal character in the header ([,], or ^), or a duplicate header. • PUTSTATUS: The status number is greater than 19, status value is greater than 255 or less than 0, the global format = 0 or is greater than 6, the VisiCalc level is greater than 4, or the coordinate display specifier is greater than 6. • PUTVALUE: The string is not a number, ERROR, or NA. • PUTWIDTH: The width is not 0-255. • VLENGTH, VWIDTH: The lenth or width is not 1-255.
200	no worksheet • BLANK, CELLTYPE, COL, COLWIDTH$, COORD$, CURSOFF, CURSON, DECOMP$, DELCOL, DELROW, DUMP, FIRSTREF$, GETFORMAT, GETFORMULA$, GETLABEL$, GETSETUP$, GETSTATUS, GETVALUE$, GETWIDTH, INSCOL, INSROW, INTERP$, LEGAL, MARK, MAXCOL, MAXROW, MOVCOL, MOVROW, NEXTREF$, PUTFORMAT, PUTFORMULA, PUTLABEL, PUTRLABEL, PUTSETUP, PUTSTATUS, PUTVALUE, PUTWIDTH, RECALC, REPLICATE, ROW, SOURCE, SPEW, TARGET, UNMARK, UPCOL, UPROW, VC There is no worksheet file in user memory whose name matches the file name of the active worksheet, or the scratch space needed by VisiCalc has been purged. WORKSHEET'*file name*' must be executed to specify which worksheet file is to be modified. When an extension function is executed, the active worksheet will be the one containing the formula that called the extension function. The active worksheet is set to null when VisiCalc exits (as if the WORKSHEET statement with no parameters was executed) to prevent inadvertant modification of a worksheet. • CALLVC, DIR$, DSP$, ENDVC, ERROR, INLINE$, IRFLAG, NA, PARSE, PRT$, RCLVAR, STOVAR The scratch space needed by VisiCalc has been purged. WORKSHEET or WORKSHEET'*file name*' must be executed to recreate VisiCalc's scratch space.

Number	Message and Condition
201	bad coordinate • BLANK, CELLTYPE, FIRSTREF$, GETFORMAT, GETFORMULA$, GETLABEL$, GETVALUE$, RECALC, SOURCE, SPEW, TARGET: Either the column or row coordinate is less than 0 or greater than 255. • COLWIDTH$, DELCOL, INSCOL, GETWIDTH, PUTWIDTH: The column coordinate is less than 0 or greater than 255. • COORD$: Either the column or row coordinate is less than 0 or greater than 255, or both are equal to 0. • DELROW, INSROW: The row coordinate is less than 0 or greater than 255. • MOVCOL, MOVROW: Either the column or row coordinate is less than or equal to 0 or greater than 255, or both are equal to 0. • PUTFORMAT, PUTFORMULA, PUTLABEL, PUTRLABEL, PUTVALUE: Either the column or row coordinate is less than or equal to 0 or greater than 255. • PUTSTATUS: For the current cell, the upper-left corner of the video, or the window access cell, either the column or row coordinate is less than 0 or greater than 255, or both are equal to 0.
202	in extension fn • BLANK, PUTLABEL, PUTRLABEL, PUTVALUE: Used in an extension function. Writes over or deletes the cell containing the formula that calls that extension function. • CALLVC, CURSOFF, CURSON, DECOMP$, DELCOL, DELROW, FIRSTREF$, GETFORMULA$, INSCOL, INSROW, INTERP$, MARK, MOVCOL, MOVROW, NEXTREF$, PARSE, PUTFORMULA, RCLVAR, RECALC, REPLICATE, SOURCE, STOVAR, TARGET, UNMARK: Not allowed in an extension function. • DUMP: Not allowed in an extension function or in a formula. • ERROR, NA: Not allowed anywhere except in a formula. • PUTSTATUS: Cannot set the column or row coordinate of any of the following to 0: the current cell, the upper-left corner of the video, or the window access cell. • VC: Not allowed in an extension function or in visi-commands. Any command used during an extension function or visi-commands that causes it to be deleted or deallocated will cause VisiCalc to exit with error 202. This error condition cannot be trapped using ON ERROR.

Errors During Extension Functions

Errors that occur during an extension function do not cause the program to halt, but result in the value of ERROR being put in the worksheet in the proper cell. Consequently, if error 202 is reported during the extension function, the error will never be returned to the user, except in the form of the ERROR value appearing in the cell.

While debugging extension functions, ON ERROR statements used in conjunction with ERRN and ERRL can help trap the conditions that generated error 202.

Warnings When Formulas Are Calculated

When INTERP$, PUTFORMULA, RECALC, or REPLICATE calculate a formula, they do so using the current default setting of the HP-75. If a formula, when calculated, produces a defaulting math error (errors 1-8), the result will be ERROR only if the HP-75 is set to DEFAULT OFF. If the HP-75 is set to DEFAULT ON, the appropriate math warning message will be issued, and the result will be the corresponding default value for that math operation.

Warnings When Formulas Are Returned

When DECOMP$ or GETFORMULA$ return a formula longer than 91 characters, the string returned will be only 92 characters long, and the 92nd character will be a question mark (?). At the same time, the LCD will blank out. The WARNING: line too long message will not appear in the LCD, but will appear on any DISPLAY IS devices. Because the LCD blanks out, if any messages are being displayed while DECOMP$ or GETFORMULA$ are being used, it may be desirable to redisplay the messages if the formula length returned is greater than 91.

The VISICM1 Visi-Commands Program

```
0 ! VISI-COMMANDS: CLEAR and SORT
1 !
2 ! Delete lines 50 - 3000 to create the visi-commands shell.
3 !
4 ! The literalized string declarations in lines 30, 8830, and 8835
5 ! are described in the VisiCalc Programmer's Reference Manual.
6 !
7 ! Lines 10 - 20 must be the first lines and should not be altered.
8 !
9 ! Initialization.
10 INTEGER P,C,R,F5,V5,V0,F1,X0,Y0,X1,Y1,Z0,Z1,Z2,S1,S4,S5,S6,S7
20 RCLVAR
30 T2$=CHR$(13)&"              `   è" @ M2=0
40 DIM B0$[96]
50 ! --------------------
59 ! Additional string dimensions.
60 DIM M$[96],T$[96]
498 ! --------------------
499 ! Select clear or sort.
500 M1=70 @ GOSUB 8795 @ ON K GOTO 1000,1610
998 ! --------------------
999 ! Clear command.
1000 M1=72 @ GOSUB 8795 @ ON K GOTO 1020,1050
1010 ! Clear worksheet.
1020 M1=74 @ M2=23 @ GOSUB 8795 @ IF K=2 THEN 9990 ELSE GOSUB 9982
1030 FOR X=1 TO MAXCOL @ FOR Y=1 TO MAXROW
1035 BLANK X,Y @ PUTFORMAT X,Y,0
1038 NEXT Y @ NEXT X
1040 V5=1 @ GOTO 9990
1049 ! Clear data.
1050 M1=75 @ M2=23 @ GOSUB 8795 @ IF K=2 THEN 9990 ELSE GOSUB 9982
1060 FOR X=1 TO MAXCOL @ FOR Y=1 TO MAXROW
1065 T=CELLTYPE(X,Y) @ IF T>2 THEN 1080
1070 IF T=1 THEN BLANK X,Y @ GOTO 1080
1075 IF NOT LEN(FIRSTREF$(X,Y,0)) THEN BLANK X,Y
1080 NEXT Y @ NEXT X
1085 IF NOT GETSTATUS(1) THEN RECALC
1090 V5=1 @ GOTO 9990
1499 ! --------------------
1500 ! Sort command.
1502 ! The sorting routine sorts a worksheet by column or row.
1504 ! This routine sorts either numerically or alphabetically.
1506 ! This routine sorts numerically in ascending or decending order.
1507 !
```

```
1508 ! The variables used are:
1510 ! K9=1/2 defines a sort directed by a column/row.
1512 ! K8=0/1 defines a numeric/alphabetic sort.
1514 ! K7=0/1 defines an ascending/descending numeric sort.
1516 ! M1 is the value of the current test variable.
1517 ! M$ is the current test string in alphanumeric sorts.
1518 ! M2 is the value of the other variable under test.
1519 ! T$ is the string under test in alphanumeric sorts.
1520 ! C1 is the address of the first column/row in a sort.
1521 ! C2 is the address of the last column/row in a sort.
1522 ! C3 is the value of the address of the largest/smallest value found.
1523 ! C0 is the address of the column/row on which the sort takes place.
1524 ! Note that the variables M1, M2, M$, and T$ are carefully shared with
1525 ! the visi-commands shell to minimize memory use.
1529 !
1530 ! Method used to sort a worksheet:
1532 ! a) Find largest/smallest value in the range.
1534 ! b) Move the column/row to the end of the sort range.
1536 ! c) Decrease the sort range by decrementing C2.
1538 ! d) Continue until C2=C1.
1540 !
1580 ! --------------------
1600 ! Prompt for column or row sort.
1610 M1=90 @ M2=25 @ GOSUB 8795 @ K9=K
1620 ! Prompt for numeric or alphabetic sort.
1630 M2=91 @ GOSUB 8795 @ K8=K-1 @ IF K8=1 THEN K7=0 @ GOTO 1680
1640 ! Prompt for ascending or descending sort.
1650 M2=92 @ GOSUB 8795 @ K7=K-1
1660 ! Determine default column/row to sort.
1680 M$=MESSAGE$(92+K9)
1685 IF K8=1 THEN B0$=MESSAGE$(38) ELSE B0$=COORD$((K9=1)*C,(K9#1)*R,S5)
1690 ! Prompt for column/row to sort.
1700 GOSUB 9115 @ C0=0
1710 ! Check for 'Headers' message in alphabetic sort.
1720 IF K8=1 AND UPRC$(B0$)=UPRC$(MESSAGE$(38)) THEN 1780
1730 ! Determine column/row coordinate.
1740 IF K9=1 THEN C0=COL(B0$) ELSE C0=ROW(B0$)
1750 ! Check for valid column/row.
1760 IF NOT C0 THEN BEEP @ GOTO 1700
1770 ! Select default column/row for start of sort.
1780 M$=MESSAGE$(97-K9)
1785 IF K9=1 THEN B0$=COORD$(0,1,S5) ELSE B0$=COORD$(1,0,S5)
1790 ! Prompt for the start of the sort.
1800 GOSUB 9115
1810 ! Check the validity of the selected column/row.
1820 IF K9=1 THEN C1=ROW(B0$) ELSE C1=COL(B0$)
1830 IF NOT C1 THEN BEEP @ GOTO 1800
1840 ! Select the defaults for the end of the sort.
1850 M$=MESSAGE$(99-K9)
1855 IF K9=1 THEN B0$=COORD$(0,MAXROW,S5) ELSE B0$=COORD$(MAXCOL,0,S5)
1860 ! Prompt for the column/row to end the sort.
1870 GOSUB 9115
1880 ! Check the validity of the ending coordinates.
1890 IF K9=1 THEN C2=ROW(B0$) ELSE C2=COL(B0$)
1900 IF NOT C2 THEN BEEP @ GOTO 1870
1903 ! Turn the cell cursor off and display 'Working'.
```

```
1904 GOSUB 9982
1910 ! Check for a valid range; if invalid then return to VisiCalc.
1920 IF C2-C1<1 THEN 9990
1960 ! These loops find the location of the column/row to be moved.
1970 ! The first loop is for numeric sorting.
1980 ! Begin each sort loop by selecting numeric/alphabetic.
1990 C3=C1 @ IF K8=1 THEN 2140
2000 ! Initialize M1 to value of first cell in range.
2010 IF K9=1 THEN T$=GETVALUE$(C0,C1,0) ELSE T$=GETVALUE$(C1,C0,0)
2015 IF POS("NAERROR",T$) THEN M1=INF ELSE M1=VAL(T$)
2020 ! Loop through columns/rows searching for largest/smallest value.
2030 FOR I=C1+1 TO C2
2040 IF K9=1 THEN T$=GETVALUE$(C0,I,0) ELSE T$=GETVALUE$(I,C0,0)
2045 IF POS("NAERROR",T$) THEN M2=INF ELSE M2=VAL(T$)
2050 ! Determine if a new high/low value has been found.
2060 IF K7=0 AND M1<=M2 OR K7=1 AND M1>=M2 THEN M1=M2 @ C3=I
2080 NEXT I
2090 IF C3=C2 THEN 2110
2100 IF K9=1 THEN MOVROW C3,C2 ELSE MOVCOL C3,C2
2105 ! Check for the end of the sort.
2110 C2=C2-1 @ IF C1#C2 THEN 1990 ELSE 1040
2120 ! This loop locates the last alphabetic item.
2130 ! Extract the first alpha string to alphabetize.
2140 IF K9=1 THEN M$=UPRC$(GETLABEL$(C0,C1)) ELSE M$=UPRC$(GETLABEL$(C1,C0))
2150 ! Start of major loop.
2160 FOR I=C1+1 TO C2
2170 ! Get the string to compare against.
2180 IF K9=1 THEN T$=UPRC$(GETLABEL$(C0,I)) ELSE T$=UPRC$(GETLABEL$(I,C0))
2190 ! Compare the strings.
2200 IF M$<=T$ THEN C3=I @ M$=T$
2280 NEXT I
2287 ! The column/row to move has been found; now go move it.
2290 GOTO 2090
3000 ! --------------------
8790 ! Key choice.
8795 IF NOT M2 THEN M2=M1+1
8796 SETWIN @ DISP MESSAGE$(M1);": ";MESSAGE$(M2) @ T$=UPRC$(CHOICE$(M2)) @ M2=0
8810 K$=WAITKEY$(-1) @ K=POS(T$,UPRC$(K$)) @ IF K THEN RETURN
8830 IF POS("  ",K$) THEN POP ELSE 8810
8835 IF K$=" " THEN P=0
8852 GOTO 9990
9099 ! --------------------
9100 ! Line entry.
9115 SETWIN @ DISP M$;": "; @ CLRLCD LEN(M$)+3
9121 W0,P0=1 @ F0=0
9123 SETWIN LEN(M$)+3
9125 K$=INLINE$(B0$,W0,P0,T2$,F0)
9130 B0$=INPBUF$ @ W0=INPWIN @ P0=INPPTR @ F0=IRFLAG @ L=LEN(B0$)+1
9180 K=POS(T2$,K$) @ IF K>3 THEN 9123
9210 IF K=1 THEN RETURN
9220 IF K=2 THEN POP @ GOTO 9990
9230 IF K=3 THEN POP @ P=0 @ GOTO 9990
9231 ! Insert other intercepts here and alter line 9180.
9399 RETURN
```

```
9979 ! --------------------
9980 ! Display working.
9982 IF VO THEN CURSOFF
9983 SETWIN @ DISP MESSAGE$(2) @ RETURN
9988 ! --------------------
9989 ! Return to VisiCalc.
9990 STOVAR @ END
```

Keyword Index

HEWLETT PACKARD

Portable Computer Division
1000 N.E. Circle Blvd., Corvallis, OR 97330, U.S.A.

00075-90223 English

Printed in U.S.A. 5/83