Agilent 3396 Series Integrator

BASIC Language Quick Reference Guide

Agilent Technologies

Agilent 3395/3396 Integrators

Manuals

These manuals may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemicals analysis businesses are now part of Agilent Technologies. The HP 3395/3396 Integrator referred to throughout this document is now the Agilent 3395/3396 Integrator.

Agilent Technologies

Contents

CONVENTIONS

[ ]	Optional terms.
Italics	Supply a value.
....	More of the same
{ }	Select one of the items (| is separator).

LOG-ON PROCEDURES

1. Type BA at the system prompt (*). BASIC Is now actlve on the HP 3396 SERIES II keyboard.
2. Type BX at the system prompt. BASIC is now active on an external display terminal. See Section 15 of the User Manual for connection instructions. The HP 3396 SERlES ll keyboard is inactive until External BASIC is exited, except that CTRL-BREAK maybe used to reset the instrument.

LEAVING BASIC

To exit BASIC (either local or external), type EXIT or just E. If the message KEEP PROGRAM IN WORKSPACE [Y*/N]: appears, respond Y (or just ENTER) to return to system mode without erasing the workspace program (it will be ready for use the next time BASIC is entered) or N to clear the workspace and return to system mode.

HELP MESSAGES

The HELP reminder (TYPE "H" FOR HELP) appears immediately after logon. Help may be requested whenever the BASIC prompt (>) or the break mode prompt (ENTER BREAK COMMAND) is present by typing H or HELP.

The HP 3396 SERIES II displays a list of section headings. The first four sections list names used in the program. Note that names no longer used are removed only when RUN is executed. The remaining sections contain listings of functionally related BASIC commands, keywords, functions and parameters. The message ENTER SECTION NUMBER follows the list.

Respond with the number of the desired section. Additional information for that section is displayed. The same result is obtained by typing H section number.

HALTING A RUNNING PROGRAM

In local (HP 3396 SERIES II keyboard) BASIC, the BREAK key suspends program execution. preserves all variables, and enters break mode. In external BASIC (using an external display terminal), the BREAK key is not active on the terminal; use CTRL Y instead. See the DEBUGGING AND RUN CONTROL section.

ABORTING A COMMAND

The BREAK key halts command execution as soon as safely possible and returns the BASIC prompt >. If a command or BASIC line is typed in, but the BREAK key is used before ENTER, the input is ignored.

OPERATORS

Execution is from left to right for operators at the same priority level. Use parentheses to override the priority order. Nested parentheses are evaluated from the inside out; six levels of nested parentheses are permitted. Where alternate syntactical forms are shown here or elsewhere in this manual, it means that all forms are accepted as input, but are converted to the preferred form (given first).

In Decreasing Priority Order

^ (or**)	Exponentiation,
NOT, unary -	Logical negation, arithmetic negation.
*, /, DIV, MOD	Multiplication, real division, Integer division no remainder, (modulo).
+, -	Addition, subtraction.
& (or +)	String concatenation.
Relationals	=, #, <, >, <=, >=; <> and >< are acceptable alternate forms of "not equal to. "
AND	Logical conjunction.
OR, XOR	Logical disjunction, exclusive OR.

NUMERIC VARIABLES

Names must start with a letter and may contain up to 31 character, including letters, numbers, and the underscore character. All characters are significant. Lowercase letters are upshifted. Names cannot be the same as certain keywords and functions. Variables are handled as either 16-bit integers (-32768 to 32767) or 32-bit floating point numbers (-1.7E38 to 1.7E38). Conversions between the two forms are handled automatically.

TYPING TEXT

At turn on, the letter keys print in uppercase (capital letters). Holding down the SHIFT key while pressing a letter key prints in lowercase. Pressing CTRL C (hold CTRL down and press C] reverses the current sense of the SHIFT key for letters, i.e., if an unshifted letter key printed uppercase before CTRL C was pressed, it will print lowercase afterward. The SHIFT key then selects uppercase. For keys other than letter keys, the SHIFT key always selects the superscripted symbol on the keycap regardless of whether CTRL C has been used: the lower symbol is printed if the SHIFT key is not used.

LABELS

A label is defined by its appearance as the first or only item in a line, and it must be followed by a colon (or a @ sign). A label name follows the same set of rules that are followed for simple numeric variable names. Labels may be used as transfer destinations before they are defined.

A label can be used instead of a line number as the destination of a statement such as GOTO or GOSUB. Statement 350 below causes control to be transferred to the label appearing in line 300.

```
300  LABEL1: PRINT X

310  Statement

320  Statement

       :

350  GOTO LABEL1
```

When forming a list of destinations, line number and label parameters may be mixed. For example,

```
100 ON X GOTO 100, HERE , 200, THERE
```

MULTIPLE STATEMENTS ON A LINE

Two or more statements may be used on the same program line, provided they are separated by colons(:), semicolons(;), @ signs, or commas(,). There are some exceptions. Note that the restriction on ELSE only applies when it is part of a multi-line IF THEN ELSE construct. A label may be used in front of all statements.

Alone	First on line	Last on Line
`DATA`	`DO`	`ELSE`
`DIM`	`DO WHILE`	`USE`
`DEF`	`DO UNTIL`	`END IF`
`INTEGER`	`FOR`	`LOOP`
`IMAGE`	`Labels`	`NEXT`
`OPTION BASE`	`WHEN EXCEPTION IN`	`REM,!`

STRINGS AND SUBSTRINGS

Strings are sets of characters enclosed by single or double quote pairs. String length (number of characters) is limited only by the amount of memory available. String variable names consist of up to 31 characters (as for numeric variables), followed by a dollar sign ($). Unlike numeric variables and arrays, a simple (non-array) string variable and a string array may not have the same name. All strings must be dimensioned in DIM statements which specify the maximum number of characters, or physical length, of the string. The actual number of characters is the logical length (logical length of A$, below, is 6). If an assignment exceeds the physical length, the string is not changed and an exception is generated.

```
1    DIM A$(9) , ASTRING$(5)
2    LET A$ = "string"
3    ASTRING$ = ‘az"by ’
```

Strings can be compared by relational operators on an ASCII numerical basis, and concatenated by the & (or +) operator.

Character positions in strings are numbered from leftmost = 1. Substrings (parts of a string) are specified by one or two numbers or expressions in parentheses. The first number specifies a beginning character. Two numbers separated by a colon specify beginning and ending characters respectively. If there is no second number, the substring extends to the end of the parent string. Note that the colon must be present (line 10).

Two numbers separated by a semicolon specify the beginning character and the number of characters in the substring respectively. If A$ is a parent string of 5 characters, then B$, C$, and D$ are identical.

```
10 B$ = A$(3:)
11 C$ = A$(3:5)
12 D$ = A$(3;3)
```

Assume a substring expression SUB$(X:Y). If X is greater than Y, a null string is returned. See line 103. The examples assume the parent string SUB$ = "ABCDEFGH". Asterisks enclose the substrings.

```
100 PRINT "*" &SUB$(1:1)&"*" *A*
101 PRINT "*" &SUB$(2:5)&"*" *BCDE*
102 PRINT "*" &SUB5(2:)&"*"  *BCDEFGH*
103 PRINT "*" &SUB$(4:1)&"*"  **
```

When a substring appears on the left side of a LET assignment, the parent string characters presently in the positions specified by the substring are replaced by the assigned value. If the first subscript is greater than the second subscript, as in line 203, the value assigned is inserted starting at the beginning position designated by the first subscript. Existing characters are pushed to the right to create space. If the resulting parent string exceeds its dimensioned length, an exception is generated.

When only one position is specified in the substring, characters in the parent string are completely replaced, beginning at the designated position. Note in line 204, that there are more parent string characters to be replaced than there are characters to fill the positions. The extra characters in the parent string, specifically "G" and "H", are deleted.

The following examples all operate on the parent string "ABCDEFGH".

Statement	Resulting SUB$
`201 SUB$(2:2) = "Q"`	`"AQCDEFGH"`
`202 SUB5(2:2) = "QR"`	`" AQRDEFGH "`
`203 SUB$(3:2) = "X"`	`" ABXCDEFGH "`
`204 SUB$(3:)  = "Acus"`	`"ABACUS"`
`205 SUB$(2:3) = " "`	`" ADEFGH"`
`206 SUB$(2:3) = "123"`	`"A123DEFGH"`
`207 SUB$(2:O) = "12"`	`"A12BCDEFGH"`

NUMERIC ARRAYS

Array storage is allocated by the DIM statement (arrays of numbers which may be integer or floating point) or the INTEGER statement (arrays of integers). An array name may be the same as a simple (non-array) variable name in the same program and follows the same rules. The first parameter in a dimensioning statement (DIM or INTEGER) defines the number of rows in the array. Second and third parameters, if present, define the number of columns and number of "layers" (the third dimension) respectively.

```
100 DIM B1(1OO), BA_OH(2,15)
101 INTEGER E(5,50,2)
```

STRING ARRAYS

String arrays can be one-or two-dimensional. A string array name follows the same form as for a single string. Line 200 defines a one-dimensional array with 2 elements, each element may contain up to 10 characters. Line 210 fills the array from a (not shown) DATA statement. In line 220, the subscript (1) specifies the element number to be printed.

```
200 DIM A1$(2)(10)

210 READ A1$(l), A1$(2)

220 PRINT Al$(l)
```

Line 300 defines a two-dimensional array with 3 rows and 2 columns. Each of the 6 elements may contain 5 characters. Lines 310 through 350 fill the array from a (not shown) DATA statement. The subscripts that identify the array element must be enclosed in parentheses.

```
300 DIM A2$(3,2) (5)

310 FOR I=1 TO 3

320 FOR J=l TO 2

330 READ A2$(I, J)

340 NEXT J

350 NEXT I
```

Substrings of elements are specified by one or two additional subscripts in a second set of parentheses.

```
400 SUB1$ = A1$[2) (3:)

401 SUB1$ = A1$(2)(6:6)

402 SUB2$ = A2$(2,1) (2;5)
```

AUTOCALL PROGRAMS

A program assigned to key 0 (zero) is an Autocall program and will run automatically at the end of each analytical run. Autocall programs cannot accept keyboard input and certain statements are not allowed. They are

```
INPUT                     START RUN_NOW [END]
LINE INPUT                START SEQ_LATER [END]
START RUN _LATER [END]    START SEQ_NOW [END]
```

DEVICE ADDRESSES

Devices attached to the HP 3396 SERIES II are identified by addresses. These are numbers in the range -2 to 30, some of which are reserved (or pre-defined):

Address	Device Type
-2	Raster Mode
-1	RS-232C
0	HP 3396 internal printer, alphanumeric
1-7	Reserved for HPIB devices
8-30	HPIL loop devices

FILE SPECIFIERS AND TYPES

A complete file specifier (abbreviated filespec) is D: NAME. EXT

D: is the disk specifier (abbreviated diskspec), and consists of a disk drive name followed by a colon (:). The default disk specifier used (for commands only) is M:, or may be set by typing in the disk drive name followed by a colon. Legal disk drive names are the capital letters A through Z, with these exceptions: H names the Host, M names the HP 3396 SERIES II Internal Memory, and E, G, I, O and Q may not be used. A disk specifier must have a connected device associated with it.

NAME, the name of the file, may contain up to 8 characters if an extension is used or 10 characters without an extension. Extensions must consist of exactly 3 characters. Some commands and keywords add default extensions if none is explicitly provided.

Valid extensions, file types, and file descriptions are shown on the following page. "HP 3396 SERIES II" files are in an LIF format which is unique to this instrument.

BAA	LIF ASCII	BASIC program file
BAS	HP 3396	BASIC program file in memory image format
DAT	HP 3396	BASIC data file
MET	HP 3396	Method file
SEQ	HP 3396	Sequence file
CAL	HP 3396	Calibration file
RAW	HP 3396	Unbunched signal data file
PRO	HP 3396	Processed peak file from an analytical run
PRA	HP 3396	Processed peak file from an ANALYZE command
RPT	LIF ASCII	Report file from an analytical run
RPA	LIF ASCII	Report file from an ANALYZE command
BNC	HP 3396	Bunched data from an analytical run
BNA	HP 3396	Bunched data from an ANALYZE command
UA1	LIF ASCII	User-defined type
UA2	LIF ASCII	User-defined type
UA3	LIF ASCII	User-defined type
UD1	HP 3396	User-defined type
UD2	HP 3396	User-defined type
UD3	HP 3396	User-defined type

Files with no extension are in LIF ASCll format, LIF binary files may be read but are not created by the HP 3396 SERIES II.

COMMAND RANGE

The term *range*, which appears as a parameter for several BASIC commands, is defined as

line number	[TO	*line number*]
FIRST	-	LAST
F	/	L

Some examples are:

```
55    20/50       FIRST TO 20   F-L
```

EDIT AND LIST COMMANDS

Commands may be abbreviated by entering enough of the initial letters to distinguish between commands, with the exceptions of FORMAT, NOLIST and SERIALIZE.

A "current line" pointer is kept internally which provides a default line for many of these commands.

The HELP, LIST and PRINT commands found in this section can also be used in break mode (see DEBUGGING AND RUN CONTROL section).

AUTO_NUMBER

Supply line numbers automatically as program is entered into workspace. A dialog requests the parameters.

DATE [MM/DD/{YY}YY]

With no parameter, lists the current date and time from the HP 3396 SERIES II internal clock. With the parameter, sets the clock to the specified date. The first two digits of the year are optional until the year 2000.

D[ELETE] [*range*] [,*range*] [...

Remove one or more lines from the workspace program. Default: delete the current line.

DQ [*range*] [,*range*] [...

Delete Quietly one or more lines from the workspace program. Lines removed are not listed. Default: delete the current line.

EXIT

Terminate BASIC and return to system mode.

HELP

See the introductory material in this manual.

L[IST] [*range*] [*range*] [...

Print all or part of the workspace program on the internal printer. Default: print entire program.

MODIFY [*line number*]

List specified line for editing; use R, D, or I to replace, delete, or insert text. Default: edit the current line; insert characters entered. Position left edge of print head below first character to be modified (or to have text inserted in front of). Terminate modify with ENTER.

N[EXT]

Increment the " current line" pointer and list the new current line.

NOLIST

Prevent workspace program from being listed or revised to provide security. Use before SAVE: permits program to be saved and run in BAS format, but not listed or revised. Scans program to check for certain errors, so that checking need not be repeated for subsequent runs. Before using NOLIST, it is usually desirable to load a copy of the program into workspace using the LOAD command to strip off remarks.

P[RINT] [*expression list*]

Evaluate and print the *expression* (*s*). If no *expression* is given, list the current line. *expressions* can consist of or contain BASIC program variables if the variables have allocated data space. Data space is allocated during and after a run until a change is made in the program or an EXIT from BASIC is executed.

```
    PRINT (X+Y) /0.01, A$
```

PROMPT {ON | OFF}

Turns printing of BASIC prompt on or off.

RENUMBER

Renumber all or part of the workspace program. A dialog requests parameters. This also allows blocks of lines to be moved.

SCRATCH

Erase the entire workspace program. Same as DQ F/L.

SERIALIZE

Reserved for future use. See the BASIC Reference Manual. CAUTION: Use of this command without proper hardware may make the program in the BASIC workspace impossible to run.

TIME [HH:MM:SS]

With no parameter, lists the current time from the HP 3396 SERIES II internal clock. With the parameter, sets the clock to the specified time (24 hour format).

XADDRESS [*(device address)*]

Specify the device for XLIST and XMARGIN. Default device: first printer on HPIL loop.

XLIST [*range*] [,*range*] [...

Print all or part of the workspace program on the device specified by XADDRESS. Default: print entire program.

XMARGIN [*column number*]

Set right margin on device specified by XADDRESS. Default: no margin (infinite iine Iength allowed).

FILE AND DEVICE COMMANDS

Copy *filespec 1*, *filespec2*
Copy contents of *filespec1* to *filespec2*. *filespec2* is created by
Copy; an error occurs if it already exists.

CREATE *filespec, filesize*

Create a file of the size specified (in bytes) on the specified or defaulted disk, An error occurs if file cannot fit on the disk if filesize is negative, CREATE builds a file as large as possible.

DIRECTORY [{*diskspec* | *filespec*}]

Print the directory for all files on a disk (use disk specifier only) or for a specific file (use file specifier).

FORMAT *diskspec*, *volume*, *numberfiles*

Prepare a disk to record information and assign It a volume name. Maximum length for volume is 6 characters, number files can be any integer; it is rounded up to the next increment of 8 when formatting floppy or hard disks and to the next increment of approximately 11 for the memory disk (M).

```
FORMAT A: ,Vol 5,64
```

GET *filespec*

Erase workspace and load specified program file. Valid extensions are BAA and BAS; if none is provided, first BAS and then BAA are tried as defaults. Only the part of a BAA file that fits in available memory is loaded. If a BAS file does not fit, an error occurs.

INET_CONFIGURATION

Prints a table showing the characterisics of the INET devices on the HPIL/INET loop.

JOIN *filespec*

Merge a BAA file with the workspace program. If line number duplication occurs, the line from *filespec* takes precedence; non-conflicting lines interleave.

LOAD *filespec*

Erase the workspace, load a BAA file and strip of all REM or ! comments and blank lines.

PACK *diskspec*

Recover fragmented, unused space on a disk.

PURGE *filespec*

Delete a file from a disk.

RENAME *filespec1, filespec2*

Change *filespec1* to *filespec2*. An error occurs if *filespec2* already exists. Both files must be on the same disk.

R[UN] [{*filespec* | *line number*}]

Fetch and execute program *filespec*, if specified. Valid extensions are BAA and BAS. if no extension is given, first BAS and then BAA are tried as defaults. If no file specified, run the workspace program starting at *line number*. If no line specified, run starting at the first line.

SAVE *filespec*

Store the workspace program on the default or specified disk Valid extensions are BAA and BAS. Default extension: BAS.

SET EXT [-] *event number*

Set an external event on an HP 19405A S/ECM module on the HPIL loop to ON or OFF. *event number* must evaluate to an integer between 1 and 8, inclusive; a negative value indicates OFF state.

SET MARGIN *columnnumber*

Define the right margin for printing, initial default is 80: range is 1 to 256; values >255 give no margin (infinite line length allowed).

SYSTEM

Print a table showing HP 3396 SERIES II configuration, such as available devices on HPIL.

TERMREAD

After opening a channel with NAME "-1" (-1 is the RS-232C port), may be used to read from the port buffer into A$ until an end condition occurs.

DEBUGGING AND RUN CONTROL COMMANDS

These commands are valid responses to the break mode prompt

ENTER BREAK COMMAND:

The BREAK, DEBUG, STEP and TRACE commands can also be used outside of break mode. The LIST and PRINT listing commands may also be used in break mode, as can the SET run parameter command.

Variable values may be listed using the PRINT command to aid debugging.

ABORT

Terminate program execution and return the BASIC prompt.

BREAK {*line number* | OFF}

Set a breakpoint at *line number*. Program suspends just before executing the specified line and break mode is entered. Only two breakpoints are permitted; if a third one is specified, the older of the two existing breakpoints is cancelled. BREAK OFF cancels all existing breakpoints.

DEBUG [{*ON* | OFF}]

DEBUG ON, or just DEBUG alone, activates BREAK and TRACE statements within the program to help isolate programming errors; DEBUG OFF disables them.

LET variable name = value

Assign a value to a variable. Attempted multiple assignment (LET A=B=5) is treated as LET A=(B=5), which sets A=1 if B=5 and A=0 if B#5.

RESUME

Exit break mode and continue program execution.

SET *variable name* = *value*

Assign a value to a variable.

STEP [{*ON* | OFF}]

STEP ON (or STEP alone) sets break points at the beginning of every program line; STEP OFF deactivates them.

TRACE *{range* | OFF*}*

For the *range* specified, list each line number as it is executed. Only two *ranges* are permitted; if a third *range* is specified, the older of the two existing *ranges* is cancelled. TRACE OFF cancels all existing *ranges*.

GENERAL PROGRAM STATEMENTS

BEEP

Emit a beep.

BREAK — Not valid in Autocall Programs

Place a breakpoint in the program. Program suspends at BREAK statement and break mode is entered. See DEBUG statement in this section.

```
190 BREAK

195 IF I=100 THEN BREAK
```

DATA *value* [,*value*] [...

Define data list for the READ statements.

```
200 DATA "VALUES", 1, 2, 3
```

DEBUG {ON | OFF} — Not valid in Autocall Programs

DEBUG ON activates BREAK and TRACE statements; DEBUG OFF disables them. Initial state is OFF.

DEF *function name (1 to 3 parameters)* = *function definition*

Define a single line function (maximum of 3 parameters). The parameters are local to the DEF line and do not conflict with program varlables of the same name.

```
260 DEF CUB(X) = X

360 DEF CUBES(X,Y,Z) = X+Y+Z
```

DIM *array or string name* (*size*) [...

Reserve storage for numeric arrays, string arrays, and string variables. Initializes numeric values to a detectable undefined state, and strings to null strings.

```
120 DIM A$(10), B(15,20), ARR$(5,10) (20)
```

END

Terminate program execution and return to the BASIC prompt. Same as STOP. An implied END is present at the end of any BASIC program.

IMAGE: *format string*

Template (printout pattern) for PRINT USING and PRINT USING # statements. In an IMAGE statement, the template is an unquoted *format string*.

```
100 IMAGE:Date of ### %% 19%%
```

INPUT *variable name list* Not valid in Autocall Programs

Print a ? prompt; accept input from the keyboard and assign it to variables listed. Multiple input responses on the same line should be separated by commas.

```
170 INPUT A$, B(3,1), C$, X
```

INTEGER {*variable* | *array, size*} [...

Declare integer variables or reserve space for integer arrays; initalize values to 0.

```
310 INTEGER A, B(6,6)
```

[LET] *variable name* = *expression*

Assign a value to a variable. Attempted multiple assignment (LET A=B=5) is treated as LET A=(B=5), which sets A=1 if B=5 and A=0 if B#5.

```
300 LET A$ = "STRING"
```

```
300 X = 100/Y
```

LINE INPUT *string variable* Not valid in Auto call Programs

Accept an entire line of input, including leading and trailing spaces, from the keyboard and assign it to *string variable*. No prompt is issued.

```
10 LINE INPUT A$
```

OPTION BASE {0 | 1}

Set lower limit for array indices. Default state: 1.

PRINT [*expression*] [{, | ;} *expression*] [...

Evaluate the *expression(s)* and send output to the current printer. The , separator spaces values apart in 14 character wide fields, while ; separator places values as close as possible. The ; separator used at the end of a line suppresses carriage return and line feed. Floating point numbers are printed with a decimal point or in exponential format; they are rounded to 6 digits and trailing zeros in the fractional part of the representation are dropped. Numbers which are in integer form are printed without a decimal point.

```
120 PRINT 5;"A STRING","ANOTHER"
```

PRINT USING {*string* | *line number* | *label*}: *expression list*

Evaluate the *expression(s)* and send formatted output to the current printer. Format is controlled by a format string, which may be supplied as a quoted string, as a string variable to which the format string has been assigned, or as a line number or label which contains an IMAGE statement. In the following examples, lines 100, 202, and 301 all produce the same printed result.

```
100 PRINT USING "#.###^^^^": X/Y

200 DIM A$(11)

201 LET A$="#,###^^^^"

202 PRINT USING A$: X/Y

300 IMAGE:#.###^^^^"

301 PRINT USING 300: X/Y
```

Each character in the format string represents one character position in the output line. Characters are either formatting characters (see the table) which control the appearance of items from the list of items being formatted, or literal characters (any not in the table), which are output unchanged. Format items are sequences of formatting characters, which must be in the order shown (they may be omitted or in some cases repeated):

Justifiers	> <
Floating characters	+ - $
Digit places	* # %
Period	.
Number signs	#
Circumflex accent	^

A format item ends when a left-to-right scan of the format string encounters an out-of-order formatting character, any non-formatting character, or the end of the format string. If the output line exceeds the margin setting for the device, the line is broken into two lines.

Formatting Numbers

Leading characters are generated according to this table:

Floating characters		Character generated	
First	Last	non-negative	negative
-	$	"$"	"-$"
$	-	"$"	"$-"
-	none	" "	"-"
+	$	"+$"	"-$"
$	+	"$+"	"$-"
+	none	"+"	"-"
$	none	"$"	"$-"
none	none	" "	"-"

A format item may contain only one of each type (sign or $) of floating character. Digit places are represented by the #, and * characters. Unused leading digit places are replaced by spaces (the # character), zeros (the % character), or asterisks (the * character). Floating characters float to the right over spaces but not over zeros or asterisks.

An integer format item consists of 1 or more of the same type of digit place character. If a decimal point is added (creating a floating point format item), then all digit places to the right of the decimal point must be the # character. Exponential notation is produced by appending four circumflex accents (^^^^) to an integer or floating point format item. The four accents represent the E, + or -sign, and a two digit exponent. Exponential format prints one digit to the left of the decimal point (if any), so at least one digit place should be present in this position.

Numbers are rounded as necessary to fit the specified format. If a number will not fit the specified format, an attempt is made to print the information by moving the decimal point, changing to exponential notation (if not specified), and/or dropping leading $ or + characters. If this fails, the format item field is filled with asterisks.

Justifiers may be used to mark the start of a new numeric format item, since only 1 justifier is allowed per format item, The justifier is replaced by the character to its right. Numbers are always right justified.

Formatting Strings

A string may be output using any format item valid for numbers. This allows the format item to be used for both a column heading and the column data. If the format item begins with a justifier, spaces are added on the right (<justifier) or the left (> justifier) to fill the format item width. If there is no justifier, the string is centered with any leftover space placed on the right.

RANDOMIZE

Generate a new seed (initial value used to generate a random sequence) for the RND function.

RESTORE *line number* [*label*]

Reset data pointer to the first item in the DATA statement specified by line number or label. If no line number or label is specified, reset to the first item in the first DATA statement

READ *variable list*

Read values from the DATA list and assign them to the variables.

```
130 READ X, Y, Z, A$
```

REM [*any text*] or ! [*any text*]

Program remarks and comments; ! can be substituted for REM. REM statements may not be used on the same line following certain key words that do not permit trailing statements: however, ! without a statement separator maybe used in these cases.

SET *runparameter value*

Set a *runparameter* to the specified value. *runparameter* choices are AR_REJ, ATT2, CHT_SP. PK_WD, THRSH and ZERO. See the Run Parameters section for limits.

```
150 SET CHT_SP 5.0
```

SET EXT [-] *event number*

Set an external event on an HP 19405A S/ECM module on the HPIL loop to ON or OFF. *event number* must evaluate to an integer between 1 and 8, inclusive: a negative value indicates OFF state.

```
100 SET EXT 1
```

SET FONT *n*

Where *n* is a numeric expression. If *n* equals 0, the font will be small. If *n* is a non-zero, the font will be large.

SET MARGIN *columnnumber*

Define the right margin for printing initial default is 80; range is 1 to 255; values > 255 give no margin (infinite line length allowed).

SET RUNNUM *runnumber*

Set the run number.

STOP

Terminate program execution and return to the BASIC prompt.

Same as END.

TAB (*colmnnumber*)

Used in PRINT statements to move printhead to specified column.

```
   380 PRINT "LEFT"; TAB(33) : "RIGHT"
```

TRACE {*ON* | *OFF*}   Not valid in Autocall Programs

TRACE ON lists each statement number as it is executed. TRACE OFF cancels the trace. See DEBUG statement in this section.

LOOPING AND CONTROL FLOW STATEMENTS

DO

DO marks the beginning of a non-indexed loop and a LOOP, LOOP UNTIL, or LOOP WHILE statement marks its end.

DO UNTIL *test*

DO UNTIL marks the beginning of a non-indexed loop with a test condition and a LOOP, LOOP UNTIL, or LOOP WHILE statement marks its end. If *test* is true, passes control to the line following the LOOP, LOOP UNTIL, or LOOP WHILE statement. If *test* is false, proceeds to the immediately following statement.

DO WHILE *test*

DO WHILE marks the beginning of a non-indexed loop with a test condition and a LOOP, LOOP UNTIL, or LOOP WHILE statement marks its end. If *test* is false, continues to the line following the LOOP, LOOP UNTIL, or LOOP WHILE statement. If *test* is true, proceeds to the immediately following statement.

ELSE

Separates the action and alternative parts of an IF THEN ELSE statement.

EXIT DO

When executed, transfers control to the line following the LOOP, LOOP UNTIL, or LOOP WHILE statement for the present DO. DO UNTIL, or DO WHILE loop.

EXIT FOR

When executed, transfers control to the line following the NEXT statement for the present FOR loop.

FOR *index* = *first* TO *last* [STEP *size*]

FOR controls the repeated execution of a group of statements bounded by FOR and NEXT statement which have the same *index* (a simple numeric variable). Execution is repeated until any EXIT FOR condition is satisfied or the value of *index* exceeds *last*. Then control is transferred to the statement immediately following the NEXT statement of the FOR loop. If *index* is omitted in the NEXT statement, NEXT is paired with the most recent FOR that is still unpaired. Default STEP *size* is +1 *last* and *size* are evaluated dynamically with every iteration. They, along with *index*, may be changed within the loop.

```
10 FOR I=1 TO N

     Statements

30 IF  test THEN EXIT FOR

     Statements

50 NEXT I
```

GOSUB {*line number* | *label*}

Transfer unconditionally to subroutine at *line number* or label.

GOTO {*line number* | *label*}

Unconditional transfer to specified destination,

IF *test* THEN *action* [ELSE *alternative*]

END IF

IF starts a conditional transfer or execution. *test* is evaluated. and if true, the *action* is executed. If *test* is false, action is ignored and *alternative*, if present. is executed. If *test* is a numeric expression rather than a relation, *test* is true if the expression is non-zero. *action* and *alternative* may be executable statements or transfer destinations (line numbers or labels). A dangling THEN or ELSE sets up a multi-line IF, and a multi-line IF must be ended by ENDIF. A nested IF cannot be on the same line as the enclosing IF.

Both *actions* and *alternatives* may consist of multiple statements on the same line, with : separators, if the *action* or *alternative* is the last item in the line. The entire group of statements is either executed or not executed, depending on the condition of *test*.

```
100 IF A=B THEN 200
150 IF A=B THEN PRINT B ELSE
160  A=-A : PRINT A
170 ENDIF
```

LOOP

Marks the end of a loop started by a DO, DO UNTIL, or DO WHILE statement. When executed, passes control to the DO, DO UNTIL, or DO WHILE statement which began the loop.

LOOP UNTIL *test*

Marks the end of a loop started by a DO, DO UNTIL, or DO WHILE statement, When executed. passes control to the DO, DO UNTIL, or DO WHILE statement which started the loop if *test* is false: otherwise transfers to the statement immediately following the LOOP UNTIL statement.

LOOP WHILE *test*

Marks the end of a loop started by a DO, DO UNTIL, or DO WHILE statement. When executed, passes control to the DO, DO UNTIL, or DO WHILE statement which started the loop if *test* is true; otherwise transfers to the statement immediately following the LOOP WHILE statement.

ON *index* GOSUB {*line number* | *label*} [,...

Indexed transfer to a subroutine. If *index* is not between 1 and the number of selections, generate an exception.

```
30 ON Y GOSUB 100,200, START
```

ON *index* GOTO {*line number* | *label*} [,...

Indexed transfer to a program line. If *index* is not between 1 and the number of selections, go to the next line.

```
20 ON X+1 GOTO 200,300, LOCATION
```

RETURN

Marks the end of a subroutine; causes control transfer to the statement immediately following the GOSUB which called the subroutine.

WAIT [DELAY] *numeric expression*

Program operation is suspended until *numeric expression* seconds have elapsed. *numeric expression* cannot be greater than 86,399 seconds. The DELAY keyword is automatically added if omitted.

```
970 WAIT DELAY AMOUNT OF TIME
```

WAIT TIME {*numeric expression* | *string expression*)

Suspends program operation until the time specified. *numeric expression* is expressed in seconds since the preceding midnight; if greater than one day (86,400 seconds) the value is reduced modulo 86,400. *string expression* is specified in 24-hour clock format ("HH:MM:SS"). If the time specified has already occurred in the present day, operation is suspended until that time the next day.

```
    200 WAIT TIME 11500
    210 WAIT TIME "05:30:30,,
```

EXCEPTION PROTECTION STATEMENTS AND FUNCTIONS

An exception is a deviation from the normal, or expected, program routine that requires special corrective action. Exception protection programming is used to separate the exception from the regular program routine and carry out appropriate actions to process it. Once the exception has been processed, normal program routine resumes when the exception block is properly exited.

CAUSE EXCEPTION *exception number*

Causes the exception indicated by *exception number*. Used to test exception protection routines.

CONTINUE

In a USE block, clear the exception and transfer to the line following the one which caused the exception.

END EXCEPTION

Used in a USE block to clear the exception state; must be followed by a GOTO to code outside the USE block.

END WHEN

Marks the end of the exception protection block which began at WHEN EXCEPTION IN. A GOTO on the same line may be used to transfer control to a line other than the one following END WHEN

EXLINE [({*line number* | *label*})]

With no parameter, return the line number where the last exception occurred, or 0 if none has occurred. With a *line number* or *label* para-meter, return 1 if the exception occurred in the specified line or 0 if it did not.

EXTEXT$ (*exception number*)

Return exception message corresponding to *exception number*. If no message, a null string is returned. To print a list of all exception messages, run the following program:

```
10 FOR I=1000 TO 13000
20 IF EXTEXT$(I)#" THEN
30 PRINT I;':'
40 PRINT EXTEXT$ (1]
50 PRINT
60 END IF
70 NEXT
```

EXTYPE

Return the exception number. The function EXTYPE returns 0 if used outside of an exception processing block.

RETRY [{ALL | ({*line number* | *label*})}]

In a USE block, clears the exception and transfers control: with no parameter, to the line which caused the exception; with the ALL parameter, to the WHEN EXCEPTION IN statement to retry the entire block; with a *line number* or *label* parameter, to the specified line, which must be in the protected block.

USE

Separates the block of statements being protected against exceptions from the correction statements. If no exception has occurred when USE is reached, control transfers to the line following the END WHEN statement.

WHEN EXCEPTION IN

Starts a block of statements for which exception protection is desired. When an exception occurs, statements following USE are executed to perform corrective action.

```
10 WHEN EXCEPTION IN
    Statement
50 USE
    Exception processing statements
90 END WHEN
```

FILE AND DEVICE STATEMENTS

channel may range from 0 to 99. Channel 0 is defined to be the device from which BASIC is being run; channel 0 is always open.

File specifiers, which appear as parameters in some of these statements, may be supplied as either quoted strings or string variables. A complete filespec is required: the diskspec portion cannot be defaulted.

LIF ASCII files (BAA, RPT, RPA, UA1 UA2, UA3, and files without extensions) and LIF binary files are read a record at a time; if the record is assigned to a numeric variable, conversion to a number is done if possible: otherwise an exception is generated. If the record read is assigned to a string variable but will not fit, an exception is also generated.

The size of elements in DAT files is 4 bytes for all numbers; strings use 4 bytes plus the number of characters in the string. Checks are done to see that requests to READ numbers or strings from DAT files are reading an item of the correct type.

Other HP 3396 SERIES II unique files are read n bytes at a time by reading a string or substring of length n. No checks are done regarding the format of what is read. It is also possible to read a 4 byte (floating point internal format) number directly from such a file by reading into a numeric variable.

ASK # *channel*: DATUM *string variable*

Assign the type of the next data item to be read from file *channel* to *string variable*. For DAT files, returns "STRING", "NUMERIC", "NONE" (end of file reached), or "UNKNOWN" (bad file format or incorrect record pointer position in file); for all other files and devices, returns "UNKNOWN".

```
40 ASK #5: DATUM A$
```

ASK # *channel*: LENGTH *numeric variable*

Assign the physical length in bytes of file *channel* to the variable. If not a file, return 0.

ASK # *channel*: LIFTYPE *numeric variable*

Assign the LIF file type number In file channel to the variable (LIF ASCII=l: LIF binary =-2; LIF HP 339 6SERlES II unique= -10241; if not a file, 0 is returned).

ASK #*channel*: RECORD *numeric variable*

Assign the current value of the record number pointer in file *channel* to the variable. This is the byte position in the file rather than a true record number. If not a file, return 0. For LIF ASCII (or LIF binary) files, note that the value returned includes bytes used for record size counts and padding, and so will not be the same as the total number of characters in the records that have been read.

```
40 ASK #6: RECORD R
```

ASSIGN *filespec, keynumber*

Assign the program *filespec* to the key *key number*, *filespec* is a string variable or quoted string; *key number* ranges from 0 to 9. The program is loaded and run when the key is pressed while in system mode. The program assigned to key 0 is loaded and run after each analytical run.

CHAIN *filespec*

Terminate execution of currently running program and clear the workspace; load and execute a program file. Only BAS files may be chained.

CLOSE # *channel*

Close access to file *channel*. If *channel* is zero or not in use, no action is taken.

COPY *filespec1* | *filespec2*

Copies the first file, creating the second file

CREATE *filespec, filesize*

Create a file of *filesize* bytes on the specified or default disk. An error occurs if the file will not fit. If *filesize* is negative, CREATE *builds* the largest possible file.

DIRECTORY {*diskspec* | *filespec*}

Print a directory of all files on a disk (use *diskspec* only) or the entry for a specific file (use complete *filespec*).

ERASE REST #*channel*

Set the end-of-file marker to the current position of the file pointer, effectively erasing the remainder of the file.

HPIL_IO RESERVED

INET_IO *address, command, string variable*

```
    20 INET 8, ''BOTTLE 100", A$
```

OPEN # *channel*: NAME {*filespec* | *device address*}

Assign a channel number to a file or an external device. If a file, open it.

```
20 OPEN #20: NAME D$
21 OPEN #30: NAME "E: AFILE. BAS"
22 OPEN B40: NAME "8"
```

PACK *diskspec*

Recover fragmented, unused space on a disk. *diskspec* must be a string.

PRINTER IS # *channel*

Define current output device for PRINT statements.

PRINT # *channel* : *expression list*

Write data to channel specified.

```
530 PRINT #12: B$,C$,D
```

PRINT # *channel* USING {*string* | *line number* | *label*}: *expression list*

Evaluate the *expression(s)* and send formatted output to the channel. If *expression list* contains more than one item and *channel* is a file, all formatted outputs are combined into a single record. Format is controlled by a format string, which may be supplied as a quoted string, as a string variable to which the format string has been assigned, or as a line number or label which contains an IMAGE statement. See the PRINT USING statement for the format string details.

```
120 PRINT #5 USING "%%":A
```

PURGE *filespec*

Delete a file from a disk.

READ # *channel* : *variable list*

Read data from the specified *channel*. Assign values to the variables listed.

```
    120 READ #24: X, Y, Z
```

RELEASE REST # *channel*

Return unused space at the end of the file (past the and-of-file marker) to the disk. For LIF ASCII files, the file pointer must be at the and-of-file (EOF) marker, or no action is taken. Even if the file is empty, a read must be done to return an EOF error before the system is aware of the EOF position and can execute this command; no action is taken if these condition are not met.

RENAME *filespec1, filespec2*

Change *filespec1* to *filespec2*. An error occurs if *filespec2* already exists.

SET # *channel*: MARGIN *column number*

Set right margin for *channel*. See SET MARGIN for range. If this statement is not used, the default is no margin (infinite line length allowed).

SET # *channel*: RECORD *numeric expression*

Set record pointer in file channel to record number numeric expression. Record number is a byte position in the file, rather than a true record number. Use RECORD 0 to set to the first byte. For LIF ASCII (or LIF binary) files, any non-zero value will set the file pointer to the physical end of the file; it is not possible to set the file pointer to a particular record except by reading all previous records.

```
    70 SET #9: RECORD 80
```

PLOTTING STATEMENTS

The chromatogram plotting operation which is invoked by the PLOT key may also be used from BASIC to plot any desired data This operation starts when a PLOT X,Y statement is executed and terminates when END PLOT is executed. While plotting data, all carriage returns and line feeds are stripped from printed material and compressed width printing is used; the printhead returns to the curve after each string.

PLOT X,Y

If not already plotting, begin plotting operation and turn PEN ON. Advance paper by X units; move the plotter printhead to Y. X range is 0 to 16500 units (1 unit = 1/660,000 cm). Y range is 0 to 1312 (full-scale). There are certain physical limits on the rapidity with which X and Y may change and yield accurate plots; see the BASIC Reference Manual for more information and specific limitations.

PEN OFF

Turn the plot dot OFF on the HP 3396 SERIES II plotter.

PEN ON

Turn the plot dot ON on the HP 3396 SERIES II plotter.

END PLOT

End plotting operation.

MATH AND STRING FUNCTIONS

Each function returns a value. x and y represent numeric expressions; n represents a numeric expression which is rounded to the nearest integer (if necessary) before use. All returned strings are shown between double quotes to mark the start and end; these quote characters are not part of the returned string.

ABS(*x*)

Absolute value of x.

ANGLE (*x,y*)

Angle (in radians) between positive x-axis and vector joining origin to point with coordinates (x, y).

`ANGLE(5,5)` returns `0.785398`

ATN(*x*)

Arc tangent (in radians) of x, i.e., value of the angle whose tangent is x.

BSTR$(*x,n*)

String representation of x using base n. (n must evaluate to an even integer from 2 to 72.)

`BSTR$ (3,2)` returns `"11"`

BVAL(*A$, n*)

Value of *A$* interpreted using base n. (n must evaluate to an even integer from 2 to 72.)

`BVAL("1F",16)` returns `31`

CHR$($x$)

ASCII character equivalent of x. See Table A

`CHR$ (90)` returns `"Z"`

COS(*x*)

Cosine of x radians.

DATE

Current Julian date from HP 3396 SERIES II internal clock. In YYDDD format.

`DATE` returns `85001 (for January 1, 1985)`

DATE$

Current date, in YYYYMMDD format.

`DATE$` returns `"19610911" (for September II , 1961)`

DATE2$

Current date, in MMM DD, YYYY format.

`DATE2$` returns `"SEP 11, 1961"`

DEVICE$(*x*)

Parameter x is an HPIL address from 8 to 30; a string is returned showing the characteristics of the device. Format is the same as in SYSTEM command listings. If no device is at that address, returns the null string (" ").

EPS(*x*)

Resolution error for x.

EXP(*x*)

e raised to the x power.

FLOAT (*four character string*)

Convert a *four character string* representing a floating point number in internal format (as may be read from a processed peak file) into a BASIC number. Reverse of FLOAT (X).

FLOAT (x)

Convert a floating point number into a *number in four character string* representing in a floating point number internal format. Reverse of FLOAT (*four character string)*

FP(x)

Fractional part of x.

INT(x)

Largest integer less than or equal to x.

INTRND (x)

Nearest integer to x.

IP(x)

Integer part (by truncating at the decimal point) of x.

KEY$

Returns one character for the last key hit on the HP 3396 SERIES II keyboard (or the terminal keyboard in External BASIC mode), then clears to zero (the NUL character). See Table A.

LCASE$(*A$*)

A$ with each uppercase letter shifted to lowercase

```
LCASE$ ("LOWERCASE") = "lowercase"
```

LEN (*A$*)

Number of characters (logical length) in *A$*.

LOG (x)

Natural (base e) logarithm of x.

LTRIM$(*A$*)

A$ with leading spaces deleted.

MAX(x, y)

Larger (more positive) of x and y.

MAXNUM

Largest floating-point value that can be represented (1,70141E+38).

MIN (x,y)

Smaller (more negative) of x and y.

MOD(x, y)

x modulo y: x-y*INT(x/y)

NUM(*A$*)

ASCII decimal equivalent of first character in *A$*. See Table A

`NUM ("*&#")` returns 42; `NUM ("*" )` returns 42

ORD(*A$*)

Ordinal position of *A$*. *A$* must be only 1 character, or else consist of one of the ORD mnemonics in Table A. For example, the ordinal position for backspace is 8 and the mnemonic is "BS".

```
ORD("BS ") returns 8
```

PI

Represented as 3.14159

POS(*A1$,A2$*)

Character position of the first character of the first occurrence in *A1$* of the string value *A2$*. If the value of *A2$* is not present in *A1$*, a zero is returned.

```
POS ("SUBSTRING","SUBSTRING") returns 4
```

REAL(*x*)

Converts an integer number into the real (floating point) format.

RND

Next number in a pseudo-random sequence.

ROUND (x, n)

x rounded to n places to the right of the decimal point, or -n places to the left if n < 0.

RTRIM$(*A$*)

A$ with trailing spaces deleted.

SERIAL_NUMBER$

Returns a 10-character serial number if an EPROM with a serial number is installed; otherwise returns a null string.

SGN *(x)*

Algebraic sign of x. Negative = -1; 0 = 0: positive= +l.

SIN(x)

Sine of x radians.

SQR(x)

Positive square root of x.

STR$ (x)

String equivalent to x as it would be printed from a simple PRINT statement.

```
    STR$(10) = "10"
```

TAN (x)

Tangent of x radians.

TIME

Current time, in seconds since the previous midnight. Resolution is 0.05 seconds.

TIME$

Current time, in 24-hour clock format (HH :MM:SS).

UCASE$(*A$*)

A$ with each lowercase letter in uppercase.

```
   UCASE$ ( "upper", ) returns UPPER"
```

UND(*numeric variable*)

1 if variable value is undefined, otherwise 0 is returned.

USING$(*A$,*x)

Formatted representation of number x, using *A$* as the template. See PRINT USING statement.

VAL(*A$*)

Number represented by *A$*. VAL ("10") returns 10

BINARY FUNCTIONS

These operate on the internal 16-bit 2's complement integer representation. MSB is bit 15; LSB is bit 0; there is no carry bit. Floating point numbers are first converted to the nearest integer. For SHIFT and ROTATE operations, + is to the right.

BINAND(x1,x2)

Bit-by-bit logical and of xl and x2.

BINCMP(x)

Bit-by-bit complement of x.

BINEOR(x1,x2)

Bit-by-bit exclusive-or of xl and x2.

BINIOR(x1,x2)

Bit-by-bit inclusive-or of xl and x2.

ROTATE (x, n)

Shift 16-bit binary representation of x, n bit positions; shift is performed with wraparound.

```
ROTATE (9,-2) returns 36
```

SHIFT (x, n)

Shift 16-bit binary representation of x, n bit positions; shift is performed without wraparound.

```
SHIFT (9,2) returns 2
```

CHROMATOGRAPHIC STATEMENTS AND FUNCTIONS

Statements

ANALYZE *filespec* [, |]   Not valid in Autocall Programs

Reintegrates raw or bunched run data in *filespec*, which must be a complete file specifier. If , | is used, uses the peak width profile from the original data storage for reintegration. Parameters must be in a single string variable or quoted string.

AREA_PERCENT

Generate Area % or Height % report (depending on mode selected in Option 4 of Report Options), using data and parameters from the current method.

BTL_BCD

Returns the bottle number (0-99) as read in the Binary Coded Decimal format from the SAMPLE connector on the rear of the integrator.

BTL_BIN

Returns the bottle number (0-255) as read in the binary format from the SAMPLE connector on the rear of the integrator.

GETCALIB *filespec*

Erase the current calibration and load the specified calibration file. ext type is automatically added if not present. *filespec* must be a quoted string.

GETMETH *filespec*

Erase the current method and load the specified method file. ext type is automatically added lf not present. *filespec* must be a quoted string.

GETSEQ *filespec*

Erase the current sequence and load the specified sequence file, ext type is automatically added if not present *filespec* must be a quoted string.

REPORT

Generate a report using the calculation procedure specified by the currently active method.

SAVEMETH *filespec*

Store the current method on the device and file specified. *filespec* must be a string variable or quoted string.

SAVESEQ *filespec*

Store the current sequence on the device and file specified. *filespec* must be a string variable or quoted string.

START RUN_LATER Not valid in Autocall Programs

Programmatic initiation of an analytical run. Run begins when [START] is pressed, and statements following START RUN_LATER are suspended until the run ends.

START RUN_LATER END *key number* Not valid in Autocall Programs

Same as START RUN_LATER except: The program containing this statement must be assigned to a numeric key and run by pressing the numeric key in system mode. When the statement is executed, the program containing it is cleared from the workspace. When the run ends, the program assigned to *key number* is loaded and executed. The *key number* program is cleared from the workspace when it ends.

START RUN_NOW Not valid in Autocall Programs

Programmatic initiation of an analytical run. Run starts immediately, and statements following START RUN_NOW are suspended until the run ends.

START RUN_NOW END *key number* Not valid in Autocall Programs

Same as START RUN_NOW except: The program containing this statement must be assigned to a numeric key and run by pressing the numeric key in system mode. When the statement is executed, the program containing it is cleared from the workspace. When the run ends, the program assigned to *key number* is loaded and executed. The *key number* program is cleared from the workspace when it ends.

START SEQ_LATER Not valid in Autocall Programs

Programmatic initiation of a sequence of analytical runs. Sequence begins when [START] is pressed, and statements following START SEQ_LATER are suspended until the sequence ends.

START SEQ_LATER END *key number* Not valid in Autocall Programs

Same as START SEQ_LATER except: The program containing this statement must be assigned to a numeric key and run by pressing the numeric key in system mode. When the statement is executed, the program containing it is cleared from the workspace. When the sequence ends, the program assigned to *key number* is loaded and executed. The *key number* program is cleared from the workspace when the sequence ends.

START SEQ_NOW Not valid in Autocall Programs

Programmatic initiation of a sequence of analytical runs. Sequence starts immediately, and program statements following START SEQ_NOW are suspended until the sequence ends.

START SEQ.NOW END *key number* Not valid in Autocall Programs

Same as START SEQ_NOW except: The program containing this statement must be assigned to a numeric key and run by pressing the numeric key in system mode. When the statement is executed, the program containing it is cleared from the workspace. When the sequence ends, the program assigned to *key number* is loaded and executed. The *key number* program is cleared from the workspace when the sequence ends.

Functions

AMT(peak number)

Concentration or amount determined by the calculation procedure.

AMT_LBL$

Amount label from Option 4 (Report Options) dialog. Maximum length: 10 characters.

AREA(*peak number*)

Baseline-corrected area, in 1/8 microvolt- seconds. of the specified peak.

CALAMT(*calibration number, level number*)

Amount of the calibration entry.

CALFIT$

Type of calibration fit. Returns "P" (=point-to-point), "L" (= linear regression), or "N" (= non-linear i.e., quadratic).

CALNUM (*peak number*)

Calibration table number of the specified processed peak. (Peaks are numbered in order of retention time.) This is the inverse of the PEAKNUM function.

CALRF(*calibration number, level number*)

Response factor at specified calibration point.

CALRT(*calibration number*)

Calibration table retention time of the calibrated peak.

CALTYPE$(*peak number*)

Type of calibration peak. Returns "R" (Reference Peak), "S" (Internal Standard Peak), "&" (Reference and internal Standard Peak), or Blank character (Other).

EXT$ (*event number*)

Indicates state ("ON" or" OFF") of the event specified. *event number* must round to an integer from 1 to 8.

GROUP_NAME$(*group number*)

Name of the specified group in the calibration table. Maximum length: 16 characters.

GROUP_SUM (*group number*)

Total of the calculated amounts of the peaks in the specified group.

HEIGHT (*peak number*)

Baseline-corrected height of the specified peak, in 1/8 microvolts.

IDENTIFIER$

Instrument identifier. Set using the system mode IDENTIFIER command. Maximum length: 12 characters.

INJTIME$

Date and time when sample was injected. Maximum length: 22 characters.

ISTDAMT

Amount of the internal standard.

ISTDNUM

Calibration number of the internal standard peak.

METHOD_NAME$

Most recently reported method file name (file name of the active method used in the most recent run). Maximum length: 14 characters. Returns null string if the active method is not stored in a file anywhere.

MULT

Value of the constant factor by which all calculation procedure results are multiplied (MULTIPLIER parameter).

NAME$(*calibration number*)

Name of the calibrated peak. Maximum length: 16 characters.

NUMCALB

Maximum calibration number used in the present calibration table.

NUMGRPS

Total number of peak groupings in a sample.

NUMLEV (*calibration number*)

Number of levels in the calibration for the calibrated peak.

NUMPEAKS

Number of peaks in the processed data file.

PEAKNUM (*calibration number*)

The peak number corresponding to the specified *calibration number*. This is the inverse of the CALNUM function.

PROC$

Calculation procedure in the current calibration table. Possible returns are NORM, ESTD, ESTD%, ISTD, and ISTD%, followed by either -AREA or -HEIGHT to show what measurement was used. If current method is not calibrated, returns AREA%.

PROCFILE$

Name of the processed data file. Maximum length: 14 characters.

REPORT_FILE$

Name of the report file. Maximum length: 14 characters.

REPORT_MEMO$

Maximum length: 126 characters. Can come from option 7 default report memo or from sample information table.

RF (*peak number*)

Response factor of the specified peak.

RT (*peak number*)

Measured retention time of the specified peak, in minutes.

RUNFILE$

Name of the unbunched or bunched data file. Maximum length: 14 characters.

RUNNUM

Current run number.

SAMPAMT

Amount of the sample (used with external and internal standard calculations).

SAMPNAME$

Name of the sample. Maximum length: 12 characters.

SAMPNUM

Number of bottle most recently injected or BCD number from the HP 3396 SERIES II back panel or from the signal data file after reintegration.

SEPCODE$(*peak number*)

Peak Separation Code for specified peak. Maximum length: 4 characters.

TITLE$

Report title from Report Options dialog. Maximum length: 42 characters.

UNCALRF

Response factor used for uncalibrated peaks.

WIDTH (*peak number*)

Width of the specified peak, in minutes.

SIGNAL DATA ACCESS STATEMENTS AND FUNCTIONS

Statements

INIT_ACCESS # *channel*

Set the stored slice access pointer to 1. Use after opening the file but before accessing stored data.

INC_SLICE_NUM (*numeric expression*)

Add *numeric expression* to the stored slice access pointer. Cause an exception if the pointed-to slice does not exist.

INC_SLICE_TIME (*numeric expression*)

Add (numeric expression /20) seconds to the current SLICE_TIME. If the new time does not fall within or on the end of an existing slice, cause an exception. If the new time is valid, adjust the stored slice access pointer to point to the slice which contains the new time.

Functions

SLICE_AREA

Area, in microvolt-seconds, of the slice pointed to by the stored slice access pointer.

SLICE_TIME

Time after start of run, in 1/20 seconds, at the end of the slice pointed to by the stored slice access pointer.

SLICE_WIDTH

Width, in 1/20 seconds, of the slice pointed to by the stored slice access pointer.

SLICE_NUM

Current value of the stored slice access pointer

SIGNAL_LEVEL

Voltage level of real-time input signal to HP 3396 SERIES II in millivolts. See the BASIC Reference Manual for more information and specific limitations.

RUN PARAMETERS

The Run Parameter functions return the current values of the run parameter settings. The current values can be changed using the SET command (which may also be used as a program statement) in this section.

Command or Statement

SET *runparameter value*

Set a *runparameter* to the specified value, *runparameter* choices are AR_REJ, ATT2, CHT_SP, PK_WD, THRSH and ZERO.

Functions or Parameters

AR_REJ

Area rejection level. Range is from 0 to 2,147,483,647 area counts.

ATT2

Chart attenuation in powers of 2, from -8 to 36.

CHT_SP

Chart speed from 0 to 30.0 cm/min for source or filtered plots, and from 0 to 3,0 mm/peak for Unigram plot.

PK_WD

Peak width setting, from 0.01 to 2.5 minutes.

THRSH

Threshold in powers of 2, from -6 to 28.

ZERO

Chart zero offset from -6 for the left margin to 100 for the right margin.

Table A HP 3396 Key and Character Set

ASCII Decimal	Graphic Display	ORD Mnemonic	HP3396 Key
0		NUL	CTRL + @
1		SOH	CTRL + A
2		STX	CTRL + B
3		ETX	CTRL + C
4		EOT	CTRL + D
5		ENQ	CTRL + E
6		ACK	CTRL + F
7		BEL	CTRL + G
8	(pad)	BS	BKSP
9		HT	CTRL + I
10		LF	CTRL + J
11		VT	CTRL + K
12		FF	CTRL + L
13		CR	ENTER
14		SO	CTRL + N
15		S	CTRL + O
16		DLE	CTRL + P
17		DC1	CTRL + Q
18	=	DC2	CTRL + R
19		DC3	CTRL + S
20		DC4	CTRL + T
21		NAK	CTRL + U
22		SYN	CTRL + V
23		ETB	CTRL + W
24		CAN	CTRL + X
25	BREAK	EM	BREAK
26		SUB	CTRL + Z
27	< =	ESC	ESC
28		FS	
29		GS	
30		RS	CTRL + ^
31		US	CTRL + _
32	(space)	SP	
33	!		!
34	"		"
35	#		#

ASCII Decimal	Graphic Display	ORD Mnemonic	HP3396 Key
36	$		$
37	%		%
38	&		&
39	’		’
40	(		(
41	)		)
42	*		*
43	+		
44	,		,
45	-		-
46	.		.
47	/		/
48	0		0
49	1		1
50	2		2
51	3		3
52	4		4
53	5		5
54	6		6
55	7		7
56	8		8
57	9		9
58	:		:
59	;		;
60	<		<
61	=		=
62	>		>
63	?		?
64	@		@
65	A		A
66	B		B
67	C		C
68	D		D
69	E		E
70	F		F
71	G		G
72	H		H
73	I		I
74	J		J

ASCII Decimal	Graphic Display	ORD Mnemonic	HP3396 Key
75	K		K
76	L		L
77	M		M
78	N		N
79	O		O
80	P		P
81	Q		Q
82	R		R
83	S		S
84	T		T
85	U		U
86	V		V
87	W		W
88	X		X
89	Y		Y
90	Z		Z
91	[		
92	\		
93	]		
94	^	^	
95	_	UND	_
96	‘	GRA	
97	a	LCA	a
98	b	LCB	b
99	c	LCC	c
100	d	LCD	d
101	e	LCE	e
102	f	LCF	f
103	g	LCG	g
104	h	LCH	h
105	i	LCI	i
106	j	LCJ	j
107	k	LCK	k
108	l	LCL	l
109	m	LCM	m
110	n	LCN	n
111	o	LCO	o
112	p	LCP	p
113	q	LCQ	q

ASCII Decimal	Graphic Display	ORD Mnemonic	HP3396 Key
114	r	LCR	r
115	s	LCS	s
116	t	LCT	t
117	u	LCU	u
118	v	LCV	v
119	w	LCW	w
120	x	LCX	x
121	y	LCY	y
122	z	LCZ	z
123	{	LBR	
124	|	VLN	
125	}	RBR	
126	-	TIL	
127		DEL	
225			LIST
226			DEL
227			ZERO
228			ATT2
229			CHT SP
230			AR REJ
231			THRSH
232			PK WD
233			EXT()
234			INTG()
235			STOP
236			START
237			PLOT
238			TIME
240			PREP
241			EDIT
242			STORE
243			LOAD
244			SEQ
245			METH
246			CALIB
247			REPORT
248			AREA %
249			OP ()

Agilent Technologies

08/97

Part Number
G1213-90120