PANAME MODULE REFERENCE MANUAL FOR THE HP/41C/41CV/41CX

Suite of Functions
Management of Matrices

Translated from the French

by

Sophie Ngozi Azorbo

English Translation Edited by
Jigekuma Ayebatari Ombu

Translation (c) 1990

F O R E W O R D

The PANAME module contains new functions for the HP-41, and applications of these functions are numerous and diverse. Like the other functions of the HP-41, those found in the original calculator as well as in other modules created by others, only the users, by the variety of their backgrounds and preoccupations are in a position to generate interest in the functions at their disposal.

However, it is useful to collect together the various examples of functions in order to increase the expertise of every user. One of the aims of the PPC Club is to support the dissemination of solutions to different problems to a wide audience and thus create permanent information for the benefit of every one of us. In this connection, it is incontestable that we owe a lot to our Club, without which the making of this module would not have been possible.

For instance, the PANAME module is provided with explanatory notes which we have endeavoured to make as explicit as possible. We however, are aware of our limitations and intend to take up this matter with every one of you, in order to create a document whose motivation will be close to that which gave rise to the making of the PPC module.

For those of you who are desirous of eventually receiving the reference manual for the PANAME module, we suggest that you contact, as soon as possible, Mr J J Dhenin, B C M W, 2 bis rue N Houel, 75005 PARIS. As soon as the manual becomes available, it will be sent to you.

How can we collectively bring the publication to fruition? Each one of us may notice in the document points that may require clarification. As far as possible, we will wait for your written comments, as these would invariably provide us with new ideas, records of which we would keep, that would enable us update our explanatory notes for the PANAME module. As much as possible, we will prefer that you submit ideas for the rewording of the various passages. Indeed, we are now so familiar with the use of the new functions that we are possibly no longer in a position to appreciate difficulties you may encounter in your use of the functions. It is you, and only you, who can say how the reference manual should be drawn up.

Finally, actual use offers the best possible explanation of a function, especially when the manual is written in a language one is not familiar with. We will therefore wait for you to inform us of details of your actual applications, which should be kept as short as possible.

Taking into account your suggestions and integrating these with our own work, we will be in a position, in the near future, to provide you with a more complete manual and above all one that would be closer to your expectations.

Happy programming!

P R E F A C E

When the HP-41 was introduced in 1979, it represented an outstanding advance in quality in its class; it introduced the handling and display of character strings. The way was paved for complete "dialogue" between the calculator and the user, a dialogue further enhanced by the sound (tones) capability of the

However significant these characteristics might be, they only represented the tip of the iceberg. The presence of an alphanumeric keyboard leads to other advantages, some immediately apparent and others more profound and richer lay beneath the internal structure of the calculator. The first characteristic is the "open architecture", as against code of programmed functions. The term "special machine language" used up to this point for programmable calculators would now be inappropriate. So much also can be said of the HP-41 assembly language. As a matter of fact, in developing this calculator, HEWLETT-PACKARD had put together an original language close to FORTH, which gave the calculator a set of properties that remain unique up to the present.

Understanding these capabilities depends on how one uses the HP-41.

THE LANGUAGE IS INTERPRETATIVE

The instructions set built into the memory of the machine are not directly handled by the microprocessor. Before being executed, the instructions are compiled and broken up into series of micro-instructions (sometimes very complex) which are then processed by the HP-41's CPU. It is this deciphering process that is termed "interpretation".

THE HP-41 ADOPTS A SYMBOLIC LANGUAGE

A CPU, whatever its configuration, most of the time is searching its memory for information to transfer elsewhere. It is possible that along the line some of the information may be modified, but this is not absolutely necessary. To be able to effect this transfer, the microprocessor ought to be capable of deciphering the origin and destination of the information, often referred to as the absolute address.
Two types of information exist:

- Data: numeric values or character strings.
- Instructions, a combination of which constitutes a programme.

In machine language these series of information are numerical. But the average user does not operate like a microprocessor. It is easier for him to remember words than numbers or even a series of instructions: programmers prefer symbols to numbers. The microprocessor is therefore left with the tedious task of matching the symbols with the absolute address in order to obtain the required information. It can do this through the absolute addresses, comparable in principle to the entries in a diary which makes it possible to find a telephone number through an individual's name. The level of development of a machine language can be measured by it's degree of symbolism.

HP-41 LANGUAGE IS MODULAR

In the world of micro-information, the HP-41 is one of the rare machines that permit the use of an indefinite number of programs. Each one can be created, modified, deleted and (if one possesses mass storage peripherals) read or stored independently.

Against this physical independence stands a logical dependence. Any program can call for the execution of instructions in another program; it suffices for the instructions to be executed to begin with a label: LBL "X..." and end with a RTN or an END instruction.

It is thus possible to break up a complex program into subroutines (these themselves could be broken up into simpler routines) in accordance with the sound principles of developing programs with decreasing levels of complexity. The problems posed by simple manipulation of data are thus thrown back at various levels of sophisticated calls and these in turn would not interfere with the logical structure of programs with clearly defined levels of calls. This offers multiple advantages:

- It makes it unlikely to encounter errors of conception in the ordering of necessary operations for resolving programming problems. If in spite of this errors occur, the problem is easily detected within a limited number of instructions.

- It is equally possible to test each subroutine in order to verify that the result obtained corresponds with expected values of a set of data to be input.

Finally, some of the routines could be so useful that one may wish to make an assembler version out of them. This module is an illustration of such an idea. Almost all the functions contained in the "PANAME" module were conceived as independent routines, written initially in "user" language and published in 1982. The idea of handling matrices and different functions dates from the beginning of that period.

PROGRAMMING THE HP-41

There are three levels of programming on the HP-41:

- PROGRAMS proper, made up of successive routines interspersed with tests, and always aimed at solving a specific problem. Programs embody the "strategic" part of the art of the programmer, and are easily comprehensible through annotations or commentaries given by the programmer as to the inner workings of a particular program.

- PROCEDURES (or subroutines). These represent the "tactical" side of programming. A procedure (routine) is usually, short, precise and economizes to the limit programmable memory available, but does not interfere with the variables themselves. Procedures are concerned with the execution of specific tasks.

It is generally possible for a subroutine to be called several times by the main program, and indeed by other programs distinct from the main program. It is therefore often retained permanently in the computer memory. This requires standardization in programming methods, an important effort-saving measure. If a routine meets such criteria it could be regarded as a new function of the HP-41 language set; a demonstration of the quintessence of the language : its flexibility and evolutionary nature.

- THE FUNCTIONS IN THE FORM OF AN ASSEMBLER. They are the elements that make up the language. A function ought to be much more than a routine in general use. The two authors of the module have achieved the feat of putting at our disposal a coherent collection of more that 120 functions.

A FIRST CONCEPT: HANDLING OF PERIPHERAL EQUIPMENT

It is necessary to try out the various functions to be able to appreciate their simplicity. Whether it is the video interface functions or the printer functions, they constitute appreciable time- saving devices during programming, as well as during the actual execution of programs. In regard to the "escape" sequence , for example, the function "CLEAR" in the module, enables us to input an equivalent of the function "SIN", where we would have previously input the two-byte instruction "31 04". In the same not only more explicit, but presents a considerable advantage over a function in the trace mode. In this way, the "PANAME" module brings with it flexibility and solves very easily problems that were regarded as insurmountable before now.

ANOTHER CONCEPT: HANDLING OF ARRAYS

How often have we expressed regrets on knowing that the HP-41 does not in its bare form possess the capability of operating on matrices. Today each one of us can determine at what point our knowledge of the usage of battery operated equipment prepared us towards the development of these new functions. The great strides that the FORTH language has made may be additional proof, if indeed necessary, for the present concept. Those who would have a chance to use RPN logic, will find firm footing in a language that has relegated BASIC to the realms of antiquity.

We hope that all those who have been able to develop programs with ease on the HP-41 will be able to see the advantages derivable from possession of the "PANAME" module.

PHILLIPE DESCAMPS

PANAME

CAT2 (1 of 2) for XROM 05

Listing	XROM
-ADV PRT 3C	05,00
AID	05,01
ID	05,02
FINDAID	05,03
OUTAX	05,04
OUTCR	05,05
OUTLF	05,06
OUTLFX	05,07
OUTSPX	05,08
OUTXB	05,09
OUTYBX	05,10
OUTa	05,11
OUTaX	05,12
RCLSEL	05,13
-82163 FCNS	05,14
CLEAR	05,15
CLEARO	05,16
CSRDN	05,17
CSRHX	05,18
CSRL	05,19
CSROFF	05,20
CSRON	05,21
CSRR	05,22
CSRVX	05,23
CSRUP	05,24
CTYPE	05,25
HOME	05,26
SCRLDN	05,27
SCRLUP	05,28
SCRLX	05,29
XYTAB	05,30
-92162 FCNS	05,31

Listing	XROM
CLBUF	05,32
8BIT	05,33
ESCAPE	05,34
PARSE	05,35
STATUS	05,36
TABCOL	05,37
UNPARSE	05,38
-82905 FCNS	05,39
BELL	05,40
CHARSET	05,41
FFEED	05,42
FORMLEN	05,43
GRAPHX	05,44
MODE	05,45
SKIPOFF	05,46
SKIPON	05,47
TEXTLEN	05,48
VSPAC	05,49
-PLOT FCNS	05,50
AXIS	05,51
BACKSP	05,52
BACKSPX	05,53
BOX	05,54
COLOR	05,55
*CSIZE	05,56
*DRAW	05,57
*HOME	05,58
*LABEL	05,59
*LDIR	05,60
*LTYPE	05,61
*MOVE	05,62
*PLREGX	05,63

PANAME

CAT2 (2 of 2) for XROM 09

Listing	XROM
RDRAW	09,00
RESET	09,01
REVLF	09,02
REVLFX	09,03
RMOVE	09,04
SETORG	09,05
-UTILITIES	09,06
/MOD	09,07
AD-LC	09,08
ALENG	09,09
ANUM	09,10
ANUMDEL	09,11
APPX	09,12
AROT	09,13
ATOXL	09,14
ATOXR	09,15
ATOXX	09,16
BLDPT	09,17
BRKPT	09,18
CHFLAG	09,19
CLINC	09,20
COLPT	09,21
GETRGX	09,22
LC-AD	09,23
LINPT	09,24
NOP	09,25
OUT	09,26
POSA	09,27
PSIZE	09,28
READEM	09,29
RG	09,30
RG+-	09,31

Listing	XROM
RG*	09,32
RG/	09,33
RG+Y	09,34
RG*Y	09,35
RG/Y	09,36
RGAX	09,37
RGCOPY	09,38
RGINIT	09,39
RGNb	09,40
RGORD	09,41
RGXTR	09,42
RGSUM	09,43
RGVIEW	09,44
SAVERGX	09,45
SIZE?	09,46
SORT	09,47
STO>L	09,48
SUB$	09,49
TF55	09,50
VKEYS	09,51
WRTEM	09,52
X=NN?	09,53
X!NN?	09,54
X<NN?	09,55
X<=NN?	09,56
X>NN?	09,57
X>=NN?	09,58
X<>F	09,59
STOAL	09,60
XTOAR	09,61
Y/N	09,62
YTOAX	09,63

Summary of Functions

Overall Module Functions

AID : Returns into X the accessory identification number of the primary peripheral device.

ID : Returns into the alpha register the identification of the primary device.

FINDAID : Searches in the loop for a peripheral device with the accessory identification number (AID) (or of a class) specified in X (< 0 for a class) and returns its address into the X register.

OUTAX : Repetition of OUTA. |X| indicates the number of repetitions.

OUTCR : Sends code 13 (carriage return) to the primary device.

OUTLF : Sends code 10 (line feed) to the primary device.

OUTLFX : Sends one or more code 10 (carriage return) to the primary device. |X| indicates the number of times the code is sent to the device.

OUTSPX : Sends one or more code 32 (Space) to the primary device. |X| indicates the number of times the code is sent to the device.

OUTXB : Sends n octet (8-bit byte) to the primary device for the byte value in |X|.

OUTYBX : Send to the primary device one or more times an octet specified in |Y|. The number of repetitions is shown in |X|.

OUTa : Similar to OUTA but sets the bit value to 7 for all octets sent (for example for inverse video on the HP82163).

OUTaX : Repetition of OUTa (cf. OUTAX).

RCLSEL : Returns into X the address of the primary device. If the SELECT value is greater than the number of the device, RCLSEL returns 1.

FUNCTIONS FOR THE HP82163 VIDEO INTERFACE

CLEAR : Clears the screen.

CLEARO : Clears the screen starting from the cursor.

CSRDN : Moves the cursor one line down.

CSRHX : Moves the cursor horizontally by the number of positions indicated in X (to the left if X < 0, to the right if the reverse is the case).

CSRL : Moves the cursor one position to the left.

CSROFF : Turns off the cursor.

CSRON : Turns on the cursor.

CSRR : Moves the cursor one position to the right.

CSRVX : Moves the cursor vertically by the number of positions in X (upwards if X < 0, downward if the reverse is the case).

CSRUP : Moves the cursor one line up.

CTYPE : Cursor type selection.

HOME : Moves the cursor to the (0,0) position.

SCRLDN : Scrolls the display one line down.

SCRLUP : Scrolls the display one line up.

SCRLX : Scrolls the display by a given number of lines in |X| (X < 0 downward, X > 0 upwards).

XYTAB : Moves the cursor to the (|X|,|Y|) position.

FUNCTIONS FOR THE HP82162 THERMAL PRINTER

CLBUF : Clears the buffer of the printer.

8BIT : Selects 8-bit mode.

ESCAPE : Selects ESCAPE mode.

PARSE : Selects "word wrap" mode.

STATUS : Recalls into X and Y the 2 octet (8-bit) status of the printer.

TABCOL : Effects an absolute tabulation at the level of the point indicated in |X|.

UNPARSE : Selects parse ("word break") mode at the 24th character

FUNCTIONS FOR THE HP82905 PRINTER

BELL : Enables the buzzer capability of the printer.

CHARSET : Selects the character set of the printer: X = 0 primary; X = 1 secondary.

FFEED : Executes a page skip.

FOR L N : Shows the number of lines in a logical page (indicated in |X|).

GRAPHX : Indicates to the printer to interpret the next number (in X) of characters as codes for graphic data.

MODE : Selects printing mode: 0 = Normal, 1 = Expanded, 2 = Compressed, 3 = Compressed-Expanded, 9 = Emphasized

SKIPOFF : Disables skip-over-perforation function.

SKIPON : Enables skip-over-perforation function.

TEXTLEN : Show   he number of lines of text per logical page
(indicated in |X|).

VSPAC : Selects the spacing between two lines; |X| indicates the
number of lines per inch.

FUNCTIONS FOR MINI-PLOTTER

AXIS : Plots an axis in accordance with the following format:

t = size of half a division (unit)
dy = distance between 2 divisions taken on the y-axis
dx = distance between 2 divisions taken on the x-axis
n = number of divisions

[INSERT FIGURE SHOWN AT PAGE 14 OF THE MANUAL]

The given data being thus:

T = t; Z = dy, Y = dx, X = n

BACKSP : Recalls a code (escape sequence).

BACKSPX : Recalls a code one or more times (|X| indicates the number of codes).

BOX : Draws a rectangle in which the two opposite sides have the coordinates (x1,y1) and (x2,y2), with following stack contents: y2 in T, x2 in Z, y1 in Y and x2 in X.

COLOR : Selects the colour of the of the sketch.

***CSIZE :** Selects the size of the characters.

***DRAW :** Traces a line through the (X,Y) coordinate.

***HOME :** Returns the pen to the origin (0,0).

***LABEL :** Prints the contents of the ALPHA register (prints the text in graphic mode in the direction defined by LDIR).

***LDIR :** Determines the direction of drawing for LABEL.

***MOVE :** Moves the pen through the (X,Y) coordinate.

***LTYPE :** Determines the line type for DRAW and RDRAW (|X| = 0 through 15)

***PLREGX :** A pointer value (bbb,eee) placed in X traces a dotted line passing through the points [(Rbbb),(Rbbb+1)],[(Rbbb+2),(Rbbb+3)] .... [(Reee- 1),(Reee)].

RDRAW : Traces a line to the coordinate (X,Y), relative to the actual position of the "pen".

RESET : Resetting function; moves the pen to the left margin and selects text mode.

REVLF : Effects reverse linefeed once.

REVLFX : Effects reverse linefeed one or more times indicated in |X|.

RMOVE : Moves the "pen" to the (X,Y) coordinate relative to
it's actual position.

SETORG : Redefines the origin (0,0) as the actual position of the "pen".

UTILITIES

/MOD : Returns the quotient into Y and the remainder into X of the Euclidean division Y/X (i.e. the function calculates the quotient and the modulo of the Y/X division in one operation).

AD-LC : Returns the coordinates (line,column) of an element of an array given its address.

ALENG : Returns (into X) the length of the string in the ALPHA register.

ANUM : Places in X the first numeric value in the string contained in the ALPHA register.

ANUMDEL : Similar to ANUM but deletes the string from the beginning up to the numeric value.

APPX : Places a string representation of the value in X without a decimal separator.

AROT : Effects rotation of the string in the ALPHA register by the number of positions in X.

ATOXL : Places in X the decimal code of the left-most character in the ALPHA register and deletes this character from the ALPHA register.

ATOXR : Places in X the decimal code of the right-most character in the ALPHA register and deletes this character from the ALPHA register.

ATOXX : Places in X the decimal code of a character, the position of which is specified in X.

BLDPT : Builds a special character in X from the data in the Z, Y and X registers. If X > 0, X = zzz.yyy. If X < 0, X = a matrix code such that Z = the first element of the array, Y = the number of rows, X = the number of columns.

BRKPT : Breaks up the contents of the X register into three numeric values (the reverse of BLDPT).

CHFLAG : In the course of programming, the user determines the state of the HP- 41 while using the regular functions in the normal mode. In program mode executing "CHFLAG" in a program two lines:

```
    01 CHFLAG

    02 "...." (a non-regular alpha string)
```

During execution of the program, the two lines reset the calculator to its state at the time the two lines were inserted.

CLINC : Clears the X register of increments (i.e. from the 4th figure after the comma).

COLPT : Constructs the column code of a matrix from data given in Y and the matrix code in X.

GETRGX : Copies into the memory specified in X, the registers of a given file which conforms with 2 increments.

LC-AD : Returns the address (register number) of a matrix element given a line number and a matrix code.

LINPT : Constructs the line code of a matrix given a line number and a column number.

NOP : No operation.

OUT : Prefix to facilitate entry of related functions.
POSA : Returns the position in an ALPHA string of a character specified in X.

PSIZE : Allocates for data storage the number of registers indicated in X.

READEM : Copies from mass storage the XMEMORY file indicated in the ALPHA register. cf. WRTEM.

RG : Prefix to aid the input and execution of related functions.

RG+- : Effects addition (or subtraction) of data in the registers indicated in Y and X.

RG* : cf RG+-, effects multiplication (of data in the registers indicated in Y and X).

RG/ : cf RG+-, effects division (of data in the registers indicated in Y and X).

RG+Y, RG*Y and RG/Y : Effects arithmetic operation on data in X, with the operand placed in Y.

RGAX : If X < 0, copies the contents of the ALPHA register into registers given by a block of 6 characters. **I**f X > 0 , places to the right of the string in the ALPHA register the contents of the register indicated in X, and up to the end of the string as if it had been filled by RGAX.

RGCOPY : If X > 0, copies the contents of the block of registers indicated in X into the registers specified in Y. **I**f X < 0, the blocks are exchanged. Permits increment of the registers.

RGINIT : If X > 0, places the value 0 in the registers (i.e. clears the registers) designated in X. If X < 0, places numbers (1,2,3...n) in the registers.

RGNb : Returns the number of arrays indicated by the code ddd,fffii in X.

RGORD : Replaces each value in the specified registers, by the range (order) of their contents.

RGSUM : Returns the sum of the values specified by the code in X. **I**f X < 0, calculates the sum of the absolute values.

RGVIEW : Entry (of values in) or display of registers. Details required.

SAVERGX : Reverse function of GETRGX. Copies the contents of the registers indicated in X, in accordance with a given index, from the pointer and following the increment j: X = bbb,eeeiijj

SIZE? : Returns the number of registers allocated for data storage.

SORT : Sorts into ascending (X > 0) or descending (X < 0) order the contents of the registers indicated in X. Sorts alpha and numeric data.

STO > L : Copies the value in X to the register specified by the address in L and with an increment value of L.

SUB$: Extraction and/or justification of a sub-string.

T55 : Enables or disables printer.

VKEYS: Lists key assignments.

WRTEM : Writes an extended memory file on to mass storage. This function is "WRTA" for extended memory.

Y/N : Simplifies programs which when run, request the user to indicate YES or NO.

X...NN : [No explanation given in the manual. Similar to function in HP-41CX].

Detailed Information

OVERALL MODULE FUNCTIONS

AID — Accessory Identification

AID (**A**ccessory **ID**entity) enables the determination of the Accessory Identity of the primary device. Accessory Identity is a number between 0 and 255 which identifies the type of device.

For example, the Accessory Identity of the Thermal Printer HP82162A is 32. If the primary device is an HP826161A printer, the AID function returns 32 in the X register.

DETAILED INSTRUCTIONS FOR AID

The AID function returns into the X register, if the level of the batteries permit, a number representing the Accessory Identity of the primary device. To know the Accessory Identity value of a given device, refer to the "Send Accessory Identity" section of the manual for the device.

If the primary device does not possess an Accessory Identity, the error message NO RESPONSE is displayed.

Related Functions :

In the I/O module : FINDAID, ID
In the HP-IL module : FINDID, SELECT, AUTOIO, MANIO
In the PANAME module : RCLSEL

APPENDIX C

The Table below shows for each class of device, its name, the range of the value of its Accessory Identification Number and the corresponding number to be placed in X when searching for the class of the device employing the function FINDAID.

CLASS	AID	CLASS IDENTIFIER
Control Device	0 to 15	-1
Mass Storage	16 to 31	-16
Printer	32 to 47	-32
Video Display	48 to 63	-48
Interface	64 to 79	-64
	80 to 95	-80
Graphic Device	96 to 111	-96
	112 to 127	-112
	128 to 143	-128
	144 to 159	-144
	160 to 175	-160
	176 to 191	-176
	192 to 207	-192
	208 to 223	-208
	224 to 239	-224
	240 to 255	-240

ID Device Identity

ID (Device **ID**entity) enables the determination of the identity of the primary device. Device Identity is a string of characters which identifies the device and generally the manufacturer and the device reference number.

For example, the Device Identity of the HP-IL RS232C Interface is "HP82164A". If the primary device is the HP-IL RS232C interface, the ID function places in the ALPHA register the string HP82164A.

DETAILED INSTRUCTIONS FOR ID

The ID function places in the ALPHA register the Device identity of the primary device. To know the corresponding string for the identity of a given device, refer to the message "Send Device Identity" in the manual for the device.

If the primary device does not have a Device Identity, the error message NO RESPONSE is displayed.

Related Functions :

In the HP-IL module: FINDID, SELECT, AUTOIO, MANIO
In the PANAME module: AID, FINDAID, RCLSEL

OUT The OUT - Function as a Prefix Key

[OUT] is a function intended to facilitate the entry of functions beginning with OUT from the keyboard. This function is particularly useful if assigned to a key. For example, OUT assigned to the [LN] key.

ASN "OUT" 15

(Press [yellow key] [ASN] [ALPHA] [O] [U] [T] [ALPHA] [LN]). Place the calculator in USER mode. Subsequently, to execute or enter into a programme a function beginning with OUT (for example OUTAX), press:

[OUT] (the LN key) [ALPHA] A X [ALPHA]

The above sequence is equivalent to:

[XEQ] [ALPHA] [O] [U] [T] [A] [X] [ALPHA]

Thus in effect saving three pressings on the keyboard each time a function beginning with the three letters OUT is required.

DETAILED INSTRUCTIONS FOR OUT

1) Assign [OUT] to a key and place the calculator in USER mode.
2) To execute or programme a function with the prefix OUT, successively press on:

[OUT] (previously assigned to a key) [ALPHA]
..... characters of the name of the function
..... excluding the first three characters (i.e O U and T)
..... (for example YBX for the function OUTYBX) [ALPHA]

OUTAX — OUTA with repetitions indicated in X

OUTAX carries out OUTA operations once or several times, and outputs the contents of the ALPHA register on to the primary device. The number of times OUTA operation is desired is given by the absolute value of the contents of the X register.

If Flag 17 is cleared, a linefeed code (carriage return (CR), code 13 or linefeed (LF), code 10) is sent following every transmission of an ALPHA string in the HPIL loop.

If Flag 17 is set, the string is sent several times without separation of the strings.

DETAILED INSTRUCTIONS FOR OUTAX

Place the string, to be sent several times, in the ALPHA register, the number of times the string is to be sent in the X register, then set or clear Flag 17 as desired (cf. above) and execute OUTAX.

ILLUSTRATION OF OUTAX APPLICATION

To draw a broken line of 40 strings of "-*" on an HP82905B printer, follow the sequence (it is assumed that the printer has been declared the primary device by a previous operation):

```
"-*" SF 17 40 OUTAX ADV
```

RELATED FUNCTIONS :

All functions beginning with OUT on the HPIL system, MANIO and SELECT which are necessary in the declaration of the status of the device.

OUTCR — Transmission of the CR (Carriage Return) code

OUTCR (**OUT**put **C**arriage **R**eturn) sends CR (carriage return, decimal code 13) code to the primary device.

OUTLF — Transmission of the LF (Line Feed) code

OUTLF (**OUT**put **L**ine **F**eed) sends LF (line feed, decimal code 10) code to the primary device.

OUTLFX — Transmission of one or more LF (Line Feed) code

OUTLFX (**OUT**put **L**ine **F**eeds by **X**) sends one or more LF (line feed,decimal code 10) codes to the primary device, the number of times is indicated by the absolute value of the quantity in the X register (this should be between 0 and 999).

DETAILED INSTRUCTIONS FOR OUTLFX

Place the number of times the LF code is to be sent in the X register and execute OUTLFX.

OUTSPX Transmission of one or more "space" codes

OUTSPX (**OUT**put **SP**aces by **X**) sends one or more "space" codes to the primary device, the number of times the code is sent being specified by the absolute value of the quantity in the X register (this should be between 0 and 999).

DETAILED INSTRUCTIONS FOR OUTSPX

Place the number of times the "space" code is to be sent in the X register and execute OUTSPX.

ILLUSTRATION OF OUTSPX APPLICATION

Example: Some printers do not have tab instructions. The **OUTSPX** function permits quite easily simulation of such instructions. As an illustration, the short program below sends to the primary device a string of fixed length L made up of the contents of the ALPHA register complemented with sufficient number of spaces. If the length of the string in the ALPHA register is greater than L, the string is reduced to code L (1).

The maximum length of the ALPHA register restricts the use of this program to strings of not more that 24 characters.

Method of application of the **OUTAT** function;

- Place the number L in the X register;
- Place the string to be sent in the ALPHA register;
- Execute **OUTAT**.

The program clears the contents of the X, T, and LASTX registers; it also sets Flag 17.

Important Note: The parameter L to be placed in X must be positive and greater than or equal to 1.

Listing of the OUTAT program:

```
01 LBL "OUTAT"      08 OUTA          15 1 E2
02 ALENG            09 OUTSPX        16 /
03 X>Y?             10 RTN           17 SUB$
04 GTO 01           11 LBL 01        18 GTO 02
05 -                12 DSE Y         19 END
06 LBL 02           13 NOP
07 SF 17            14 CLX
```

NB: The text is left justified. To make the text right justified, the OUTA and OUTASPX functions should be used.

OUTXB Transmission of a character given its decimal code

OUTXB sends to the primary device, the character whose decimal code is specified by the absolute value of the quantity in the X register. This value usually lies between the interval 0-255.

DETAILED INSTRUCTIONS FOR OUTXB

Place the code for the character to be sent to the primary device in the X register and execute OUTXB.

EXAMPLE OF OUTXB APPLICATION

Example: To send to send the character "\" (inverted slash, decimal code 92) to the primary device, use the function sequence: 92 OUTXB.

OUTYBX Transmission of one or more characters by their codes

OUTYBX sends one or more times to the primary device identical characters whose decimal code is specified by the absolute value of the quantity in the Y register. The absolute value of the quantity in the X register specifies the number of characters sent to the primary device.

Restrictions: 0 <= ABS(X) <= 999 and 0 <= ABS(Y) <= 255

DETAILED INSTRUCTIONS FOR OUTYBX

Place the decimal code of the character to be sent to the primary device in the Y register, the number of times the character is to be sent in the X register and execute **OUTYBX**.

EXAMPLES OF OUTYBX APPLICATION

Example 1 : To send a series of twenty " ' " characters (apostrophe, decimal code 39), use the following sequence: 39 ENTER_ 20 OUTYBX.

Example 2 : "FMTNBZ" (ForMaT NumBer with Zero).

This program simulates formatting of numbers without the suppression of leading zeros. It is effected as follows:

- Place in the X register the number to be printed;
- Place in the Y register the maximum number of characters that can be printed (the size of the printing area to be occupied by the number of characters);
- Select the mode of printing required;
- Execute FMTNBZ.

If the number of characters required to represent the content of the X register is greater than the value placed in the Y register, the corresponding position is filled with " * ".

After execution of the program, the contents of the X, Y, LASTX and ALPHA registers are lost.

Listing of the FMTNBZ program:

```
01 LBL "FMTNBZ"
02 CLA              10 -48         18 OUTXB
03 ARCL X           11 X<>Y        19 RTN
04 X<0?             12 OUTYBX      20 LBL 01
05 XEQ 00           13 OUTA        21 CLX
06 CLX              14 RTN         22 42
07 ALENG            15 LBL 00      23 X<>Y
08 X>Y?             16 CLX         24 OUTYBX
09 GTO 01           17 ATOXL       25 END
```

OUTa — OUTA with the bit value set to 7

OUTa functions like OUTA, but with the following difference: the bit value of 7 transmitted with each octet forces the bit value to 1 (in other words, 128 is added to the bit values less than or equal to 128), with the exception of the two bits which constitute linefeed or carriage return codes (CR and LF, codes 13 and 10 respectively) sent to the end of the ALPHA string when Flag 17 is clear.

DETAILED INSTRUCTIONS FOR OUTa

Place the string to be transmitted in the ALPHA register, set or clear Flag 17 as desired (cf. above) and execute OUTa.

EXAMPLES OF OUTa APPLICATION

Example 1: To display a string of characters in "inverse video" on the HP82163 video interface, add 128 to the character codes of the string before sending the string on the interface loop. This operation is achieved automatically using the OUTa function. To display a string of characters in "inverse video" it suffices to declare the video interface the primary device, place the string in the ALPHA register and execute OUTa. The status of Flag 17 determines the transmission or otherwise of a linefeed code.

Example 2: Some printers possess the capability of underlining characters automatically, this requires the addition of 128 to the code of each character that one desires to underline. The OUTa function achieves enormous simplification of the underlining operation with such printers.

Example 3: Access to the special characters of the HP82905B printer can be achieved in two ways:

- By employing the "secondary character game" which gives new meanings to the character codes between 32 and 127.
- By employing character codes beyond the range 0 to 127.

The second possibility is particularly simplified through the use of the OUTa function.

RCLSEL Recalling the address of the primary device

RCLSEL places in the X register, if the level of the batteries permit, a whole number representing the HPIL address of the primary device. Furthermore **RCLSEL** verifies the completeness of loop and that all the devices have adequate power supply (the effect on devices possessing the "STANDBY" mode is similar to that of the PWRPUP function; see the description of this function in the manual of the HP82160A module). Unlike the function of the same name in the HP82183A Extended I/O module, RCLSEL function of the PANAME module returns a value which may be different from the last address specified by the SELECT function. This occurs when the user specifies a parameter for SELECT which is greater than the number of devices on the loop; in that case, the address returned by RCLSEL is equal to 1. This characteristic is particularly useful in programs comprising a loop executed once for each device in the loop. A simple test comparing the address of the device to be treated and the value returned by RCLSEL permits one to know if all the devices have been treated. As an illustration, the LOOP programs (given as example of AID and ID application) and FNDAIDN (given as example of FINDAID application) apply this method.

DETAILED INSTRUCTIONS FOR RCLSEL

Execute **RCLSEL** and a number representing the address of the primary device is recalled into the X register in accordance with the procedures detailed above.

EXAMPLE OF RCLSEL APPLICATION

RCLSEL can be applied in a program that modifies the selection of the primary device to permit the restoration of this selection at the end of the program. The initial value of the address of the primary device is stored in a register at the beginning of the program by the sequence RCLSEL STO nn and restored when needed by RCL nn SELECT.

HP-82163 VIDEO INTERFACE GROUP

The functions of the above Group are aimed at facilitating the use of the HP82163 Video Interface. The functions make it possible for the user to control the interface without employing escape sequences or of control characters required by the interface to implement a function, such as clearing the display or moving the cursor downward. For example, for these two functions, the CLEAR and CSRDN (Cursor Down) functions are used respectively.

All these functions require the interface to be declared the primary device. (Refer to the application instructions for FINDAID (in this manual) and FINDID (in the HPIL HP82160A manual) to find out the different methods for selecting a particular device.

In AUTOIO mode, if the primary device does not have an Accessory Identification equal to 48 (Standard Video Interface), the error message AID ERR is displayed.

Nevertheless, the functions do not implement the foregoing verification process in MANIO mode, thus permitting their usage, for example, with the Mountain Computer MC 00701 Video Interface (with Accessory Identification 50), PAC-TEXT Interface (48) or with the KRISTAL MINITEL[*] Interface (48).

For more technical details about the sequences sent by these functions, refer to Appendix V.

----

[*] KRISTAL Chemin des Clos Zirst, 38240 MEYLAN (FRANCE), an interface systems, technical applications and instrumentation company is an HP approved original equipment manufacturer (OEM).

CLEAR — Clearing the Display

CLEAR clears the display, places the cursor at the (0,0) position and selects replacement mode of the cursor.

DETAILED INSTRUCTIONS FOR CLEAR

Execute CLEAR.

EXAMPLE OF CLEAR APPLICATION

The sequence ESC E sent by the CLEAR function is used by the HP82905B printer as a "reinitilisation" sequence. The CLEAR function can thus be used to reinitialise this printer, the MANIO mode being necessary to prevent the CLEAR function from verifying that the Accessory **I**dentity of the primary device is equal to 48.

CLEARO — Clearing the Display from the Cursor

CLEARO clears the display from the cursor position to the end of the screen. The cursor position and the cursor type (i.e. replace or insert) is not changed.

DETAILED INSTRUCTIONS FOR CLEARO

Execute CLEARO

CSRDN — Movement of the Cursor Downward

CSRDN (**C**ur**S**o**R** **D**ow**N**) moves the cursor one line downward. If the cursor is on the last line of the display, the cursor is not moved.

CSRHX — Movement of the Cursor Horizontally

CSRHX (Move **C**ur**S**o**R** **H**orizontally by **X**) moves the cursor horizontally, by the number of positions specified by the absolute value of the contents of the X register and in the direction specified by the sign of the value in the X register: if X < 0 the movement is to the left, and to the right if X >= 0. For example, -1 CSRHX is equivalent to CSRL and 1 CSRHX is equivalent to CSRR.

DETAILED INSTRUCTIONS FOR CSRHX

Place the desired number of movements in the X register in line with the details given above and execute CSRHX.

CSRL — Movement of the Cursor to the Left

CSRL (**C**ur**S**o**R** **L**eft) moves the cursor one position the left. If the cursor is on the first column of a line, it is moved to the last column of the preceding line; if it is at the (0,0) position, no movement takes place.

CSROFF — Suppression of the Cursor

CSROFF suppresses the cursor display. The cursor would not be visible until the next execution of CLEAR or CSRON or the subsequent initialisation of the interface (due to a reset, or cleared by an HP-IL device (DCL) or cleared by a selected device
(SDC).

CSRON — Display of the Cursor

CSRON activates the cursor display. This display can be suppressed by executing the CSROFF function.

CSRR — Movement of the Cursor to the Right

CSRR (**C**ur**S**o**R R**ight) moves the cursor one position to the right. If the cursor is at the last column of a line, it is moved to the first column of the next line, except that in the case where it is at the (31,15) position, no movement takes place.

CSRVX — Vertical Movement of the Cursor

CSRVX (Move **C**ur**S**o**R** Vertically by **X**) moves the cursor vertically by the number of positions specified by the absolute value of the contents of the X register and in the direction specified by the sign of the value in the X register: if X < 0 the movement is downward, and upward if X >= 0. For example, -1 CSRVX is equivalent to CSRUP and 1 CSRVX is equivalent to CSRDN.

DETAILED INSTRUCTIONS FOR CSRVX

Place the desired number of movements in the X register, in line with the above details and execute CSRVX.

CSRUP — Movement Upward of the Cursor

CSRUP (**C**ur**S**o**R UP**) moves the cursor one line up. If the cursor is on the first line of the display, no movement takes place.

CTYPE — Selection of Cursor Mode

CTYPE (**C**ursor **TYPE**) selects the "cursor mode" in accordance with specified value in the X register.

- For X = 0, selects the "insert mode" (blinking arrow) cursor;
- For X = 1 or -1, selects the "replacement mode" (blinking solid bar) cursor.

DETAILED INSTRUCTIONS FOR CTYPE

Place the corresponding value of the cursor mode desired in the X register and execute CTYPE. Note that when using the Mountain Computer MC00701A Video Interface, selecting the "insert mode" cursor (blinking underline character) does not turn on the "insert feature" of characters or of lines.

HOME — Returning the Cursor to the (0,0) position

HOME moves the cursor to the (0,0) position.

SCRLDN — Scrolling down of the Display

SCRLDN (**SCR**oll **D**ow**N**) scrolls the display one line down (that is, the function causes the last line of the display to disappear and a new line appears at the top of the screen).

SCRLUP — Scrolling Up of the Display

SCRLUP (**SCR**ol**L UP**) scrolls the display one line up (that is, the function causes the uppermost line of the display to disappear and a new line appears at the bottom of the screen).

SCRLX — Scrolling the Display by given Parameter in X

SCRLX (**SCR**ol**L** as specified in **X**) scrolls the display up or down by the number of lines indicated in X.

- For X < 0, SCRLX scrolls the display up by (-X) lines (corresponds to the repeated execution of SCRLUP for the absolute value in X).

- For X >= 0, SCRLX scrolls the display down by the number of lines indicated in X (corresponds to the repeated execution of SCRLDN for the value in X).

DETAILED INSTRUCTIONS FOR SCRLX

Place the desired number of lines to be scrolled in line with the details above and execute SCRLX.

XTTAB — Movement of the Cursor to an (X,Y) Position

XTYAB ((**X**,**Y**) **TAB**ulate) moves the cursor to the (X,Y) position, the column number indicated by the absolute value of the contents of the X register, and the row number indicated by the absolute value of the contents of the Y register.

DETAILED INSTRUCTIONS FOR XYTAB

Place the column number in the Y register and the row number in the Y register and execute XYTAB.

APPENDIX V

Control Codes sent to the Primary Device by the Functions of the
82163 Group. "ESC" represents the "Escape" Symbol, Decimal Code 27.

Functions	Sequence	Character Codes
CLEAR	ESC E	27 69
CLEARO	ESC J	27 74
CSRDN,CSRVX for X >= 0	ESC B	27 66
CSRL,CSRHX for X < 0	BS	08
CSROFF	ESC <	27 60
CSRON	ESC >	27 62
CSRR,CSRHX for X >= 0	ESC C	27 67
CSRUP,CSRVX for X < 0	ESC A	27 65
CTYPE for X = 0	ESC Q	27 81
CTYPE for X = 1 or -1	ESC R	27 82
HOME	ESC H	27 72
SCRLDN,SCRLX for X >= 0	ESC T	27 84
SCRLUP,SCRLX for X < 0	ESC S	27 83
XYTAB	ESC % {c} {l}	27 37 {col}{row}

HP-82162 THERMAL PRINTER GROUP

The functions of this group are intended to facilitate the use of the HP-82162A Thermal Printer. These functions will enable the user to exploit fully all the facilities available in the printer, most of which are not explained in the manual. These facilities are:

- Two different character types
- A "word wrap" mode
- Possibility of tabulation at a point, independent of data present in the buffer of the printer.
- Possibility of knowing precisely the state of the printer.

These functions operate on the first HP8261A present on the loop after the primary device. If no HP8216A printer is found on the loop, the error message NO 82162 is displayed. The only exception to this rule concerns the STATUS function of the PANAME module; for more information, consult the detailed instruction for use of the STATUS function.

8BIT — Selection of "eight bits" mode

8BIT selects the "eight bits" mode which turns on the HP-41 character set. This mode is selected automatically during the execution of specific printing functions (functions appearing under the heading - PRINTER 2E - of the HPIL module). This function is therefore useful only if one is using the HP82162A printer with non-printing functions such as OUTA or OUTYBX.

ESCAPE — Selection of the "escape" mode

ESCAPE selects the "escape" mode, which activates the (non HP-41) ASCII character set. In this mode, characters sent to the printer cannot be printed by specific printing functions because these re-select the "eight bits" mode. However, certain applications require the use of the ASCII character set. The ESCAPE function enables the use of this set with such functions; printing ought therefore should take place using the OUTA function or other related functions which only enable the transmission of characters to the primary device. Note also that in this case, the printer ought to be declared the primary device which is not required with specific printing functions such as PRA.

PARSE — Selection of the "word wrap" mode

PARSE selects the "word wrap" mode, which enables the automatic printing of text without word truncation at the line feed point. A line feed is generated by the printer in between two words, if the word following the space between the two words cannot be printed entirely on the current line.

CLBUF — Clearing the Buffer

CLBUF places the printer in the same condition as when it was first turned on, that is:

- the print head is returned to the right
- the print buffer is emptied
- the active modes are: "escape", normal width, upper case and "word wrap" at the 24th character.

This function is primarily used to empty the buffer of its contents, this operation being impossible otherwise except by putting off the printer.

UNPARSE Selection of "word break" mode

UNPARSE cancels the "word wrap" mode enabled by the PARSE function.

TABCOL Tabulation by Columns

TABCOL permits unlimited tabulation at the level of column of points (as against SKPCOL which permits only relative tabulation).

In using TABCOL, one can easily print matrices of several columns (FMT permits only two columns).

DETAILED INSTRUCTIONS FOR TABCOL

Place the number of columns of tabulation desired (0 to 167) in the X register and execute TABCOL.

EXAMPLE OF TABCOL APPLICATION
To print the following matrix:

```
A =  123.00  FF
B =   23.95  FS
C = 1115.70  FB
```

One can use the sequence:

FIX 2		
CLBUF		
"A="	"B="	"C="
ACA	ACA	ACA
28 TABCOL	28 TABCOL	28 TABCOL
123 ACX	23.95 ACX	1115.7 ACX
91 TABCOL	91 TABCOL	91 TABCOL
"FF"	"FS"	"FB"
ACA	ACA	ACA
PRBUF	PRBUF	PRBUF

HP-82905B PRINTER GROUP

The functions of this group are intended to facilitate use of the 80- column HP82905B printer. The functions enable the user to completely control the printer without directly knowing the escape sequences and the character controls required by the printer in order to execute a given a task. These functions ease considerably the writing and legibility (of print-listed) programs, bringing into play the numerous modes of operation of the HP82905B printer.

All these functions require that the printer be declared the primary device. The user is advised to refer to use instructions for FINDAID (in this manual) and FINDID (in the manual of the HP82160A HPIL module) functions, in order to know the different methods for selection of particular devices as primary device.

In AUTOIO mode, if the primary device does not possess an Accessory Identity equal to 33, the error message AID ERROR is displayed. However, the functions do not carry out this verification in MANIO mode. This feature permits the use of these functions with other printers using the same escape sequences and character controls as the HP82905B printer.

For more technical details concerning the sequences sent by these functions, refer to Appendix B.

BELL — Buzzer Signal

BELL activates the buzzer of the printer for a second. This function can be used, for example, to alert the user of a particular condition requiring the user's intervention.

CHARSET — Selection of the Printers's Character Set

CHARSET selects the primary character set if X = 0 and the secondary set if X = 1; refer to the User Manual of the HP82905B printer for a list of the two character sets.

FFEED — Page Skip

FFEED sends a "page skip" (form feed) command to the printer which causes the paper to advance to the top of the next page. Note that the user is to position the paper correctly and define the number of lines per page corresponding to the paper in use (by means of the FORMLEN function) for the FFEED function to move the paper to the next page.

FORMLEN — Page Length

FORMLEN determines the number of lines in a logical page (this number depends on the type of paper used and the spacing between lines selected by the VSPAC function).

The contents of the X register (irrespective of sign) indicates the required number of lines, which ought to be between 1 and 128. When turned on or re-initialised with the CLEAR function (see the detailed instruction for use of the CLEAR function for a description of this possibility) the default number of lines is 66.

GRAPHX Generation of Graphics

GRAPHX signals the printer to interpret the octets following, not as characters, but as binary data, each value corresponding to a column of points. Consult the User Manual of the printer for the correspondence between received values and the and result print (paragraph on "Graphics Mode").

The contents of the X register indicates the number of octets the printer is to interpret as graphics data (the sign of X is ignored).

MODE Printing Mode

MODE determines the printing mode in accordance with the value in the X register (the sign is ignored) in conformity with the following table:

Content of X	Mode	No. of char./line
0	Normal	80
1	Expanded	40
2	Compressed	132
3	Compressed-Expanded	66
9	Emphasized	80

The user can combine "0" and "1" or "2" and "3" modes on the same line; other combinations can give unexpected results.

If X contains a value other than 0,1,2,3 or 9, the calculator displays the message DATA ERROR.

SKIPOFF Disabling the Top-of-Form Function

SKIPOFF disables the SKIPON function.

SKIPON Activating the Top-of-Form Function

SKIPON enables the "skip-over-perforation" function. When this function is on, the printing of the last line of the text of a page (the number of lines of text in a page is determined by the TEXTLEN function) activates automatic advance of the paper to the top of the next page, thus avoiding printing on the perforations separating two pages.

The "skip-over-perforation" function is inactive when the printer is turned on, or when initialized by the CLEAR function (see detailed instructions on the use of the CLEAR function for details).

TEXTLEN Length of Text

TEXTLEN defines the number of lines of text in a logical page.

The contents of the X register (without taking into account the sign of the contents) indicates the number of lines of text desired, which should be between 1 and the number of lines on the page (determined by the FORMLEN function). When turned on or reinitialized by the CLEAR function (see the detailed instruction for the use of the CLEAR function for description of this possibility), the number of lines used by the printer is 60.

VSPAC Vertical Spacing

VSPAC defines the vertical spacing in number of lines per inch (1 inch = 2.54 cm) determined by the value in the X register (the sign being ignored). This value can only be 6, 8, 9, 12, 24, 36 or 72. If X contains a value different from the foregoing, the calculator displays the message DATA ERROR.

APPENDIX B

Escape Sequences sent to the Primary Device by the Functions of the HP82905 Printer Group

- ESC represents the "escape character" with decimal code 27
- {#} symbolises the ASCII representation of a number (parameter) and {by} the corresponding character codes

Function(s)	Sequence	Codes	Thinkjet
BELL	BEL	07	
CHARSET for X = 0	SI	15	Normal
CHARSET for X = 1 or –1	SO	14	Emphasized
FFEED	FF	2	FF
FORMLEN	ESC &l {#} P	27 38 108 {by}	FL
GRAPHX	ESC *b {#} G	27 42 98 {by}	
MODE	ESC &k {#} S	27 38 107 {by}	
0			Normal (80 c/l)
1			Expanded (40 c/l)
2			Compressed (142)
3			Expanded-Compressed
SKIPOFF	ESC &lOL	27 38 108 48 76	skipoff
SKIPON	ESC &l1L	27 38 108 49 76	skipon
TEXTLEN	ESC &l{#}F	27 38 108 {by}	textlen
VSPAC	ESC &l{#}D	27 38 108 {by}	vspac

MINIPLOTTER

Different models of mini-plotters which can be operated to much advantage with the HP-41 exist. TANDY, CANON, for example, employ the same mechanical design and the same modes of command like the HP82905 printer. It is, however, necessary to interface such mini-plotters with the HP-IL loop by means of the GPIO (HP82166A) converter.

Other manufacturers have also introduced into the market parallel converters that can be used to interface these mini-plotters with the HP-IL loop.

The main characteristics of these mini-plotters are:

- Four printing colours: black, red, blue, green; the tracing is carried out by a "tracing head" which in effect is a barrel containing a set of four ball points. The colour change can be carried out by a program or by pressing a switch in front of the plotter during the tracing process.

- Large standard roll paper of 11.4 cm size. It thus possible to trace or write in using the paper along its length and which permits the printing of matrices of large widths (27 or 29.7 cm).

- Along the width, one can print texts of 80 characters maximum per line.

- These mini-plotters can, of course, be controlled by other hardware similar to the HP-41 (HP-75, HP-85, HP-71) and consequently such equipment, like other HP-IL devices will not quickly become "obsolete".

AXIS — Plotting of the Axes

AXIS plots various types of axes on the mini-plotter.

DETAILED INSTRUCTIONS FOR THE AXIS FUNCTION

AXIS uses 4 parameters which the user places in the stack (of the calculator) before executing the function:

T: half the size of each division
Z; distance between 2 divisions vertically
Y: distance between 2 divisions horizontally
Z: the number of divisions

The axis is plotted from the current position of the pen, and the direction of the sketch depends solely on the Y and Z values. On the other hand the divisions are always either vertical or horizontal, their "direction" being determined by the inclination of the axis in relation to the horizontal (X axis): below 45* the divisions are vertical, above that angle they are horizontal.

The parameter in T facilitates the plotting of grids, for example, in matrices.

Example: The following program plots a matrix with 2 lines of columns. Each column having a width of L and each line having a height of H. To use it, it suffices to execute (XEQ) "TABLO" and to respond to the questions asked by supplying the corresponding values followed by R/S:

```
01 LBL "TABLO"
02 "HP82166"          Identification of GP-IO converter
03 FINDID             Searching for mini-plotter position
04 SELECT             Selection of the mini-plotter
05 RESET              Resetting
06 "NO. OF COLS. ?"
07 PROMPT             Entry of number of columns
08 STO 00             R00 = Number of columns
09 "COL. WIDTH ?"
10 PROMPT             Entry of width of columns
11 STO 01             R01 = Width of the columns
12 *                  First dimension of the matrix
13 "LINE HT. ?"
14 PROMPT             Entry of the height of each line
15 STO 02             R02 = Height of each line
16 ST*X               There are two lines, the second has dimension of 2*X
17 CHS                Downward movement
18 X<>Y
19 0
20 ENTER_
21 BOX                BOX uses the 4 parameters in T,Z,Y and X
22 RCL 02
23 CHS
24 0
25 *MOVE              Starting position
26 RCL 02
27 0
28 RCL 01
29 RCL 00
30 AXIS               Sketching of inner lines
31 END

XEQ "TABLO"
NO. OF COL. ?   To be printed
4,000 RUN
COL. WIDTH ?
100,000 RUN
LINE HT. ?
50,000 RUN
```

BACKSP Backspacing by one Character

BACKSP (**BACKSP**ace) moves the pen one character backwards.

BACKSPX Backspacing by several Characters

BACKSPX (**BACKSP**ace by **X**) moves the pen backwards by the number of characters specified in X. Only the absolute value of X is taken into account.

BOX — Sketching of Rectangle

BOX sketches a rectangle whose opposite corners have the coordinates:

(x1,x2) and (x2,y2) with T = y2, Z = x2, Y = y1 and X = x1

COLOR — Choice of Colour

COLOR selects one of the four colours according to the value in the X register.

CSIZE — Character Size

CSIZE (Characters SIZE) selects character size. The value could be between 0 and 63.

DRAW — Drawing a Segment

DRAW sketches a segment to the right from the current position of the pen to the coordinate (X,Y).

HOME — Returning the Pen to the Origin

HOME returns the pen to the (0,0) position.

LABEL — Printing the Contents of the ALPHA Register

LABEL prints the contents of the ALPHA register. This function is useful in that the printing can be done in 4 direction in text mode; these directions are defined by LD.

STATUS — Recalling the Status of the Printer

STATUS places in the Y register an integer representing the decimal equivalent of the first octet of the printer status, and in the X register, an integer representing the decimal equivalent of the second octet of the printer status. The effect of the STATUS function on the stack depends on whether stack lift was enabled or disabled at the moment of execution of the STATUS function.

- If stack lift was enabled, the effect is as follows:

Before	After
T: t	T: y
Z: z	Z: x
Y: y	Y: status of first octet
X: x	X: status of second octet

- If stack lift was disabled, the effect is as follows:

Before	After
T: t	T: z
Z: z	Z: y
Y: y	Y: status of the first octet
X: x	X: status of the second octet

In both cases, the content of the LASTX register is not changed.

The STATUS function has a peculiar characteristic: in MANIO mode, it places two numbers representing the status of the primary device in the X and Y registers.

- If the primary device loses its octet status, STATUS places 97 in the X and Y registers.
- If the primary device is in a single octet status, STATUS places the decimal equivalent of this octet in the Y register and 64 in the X register.
- If the primary device is in two or more octets status, the STATUS function has the same effect on the primary device as it has on the HP82162A in the AUTOIO mode. Status octets beyond the second are ignored.

To determine the number and the octet status of a particular device, refer to the description of the HPIL message "Send Status" in the manual for the device.

The SI Appendix gives detailed definition of the two octet status of the HP82162A printer.

APPENDIX T2

Minimum Number of Commands required for complete handling of a 4-Colour Mini-Plotter by the Functions of PLOTTER FUNCTIONS (PLOT FCNS) Group.

To obtain detailed description of a mini-plotter refer to JPC, Number 15, June 1984.

Standards of Representation:

- # represents a numeric string of characters with an implied sign and at most four figures (examples: -230; 0024);
- The "syntax" column indicates the significant portion of each of the parameters.

Control Characters (Decimal Values):

17: Conversion to "Text" Mode
18: Conversion to "Graphic" Mode
11: Line feed backwards ("Text" Mode only)
08: Movement of the pen one character backwards ("Text Mode only)

Instructions in "Graphic" Mode

Syntax	Format	Action
A	A	Initialisation
H	H	Movement to (0,0) position
Mx,y	M#,#	Movement to (x,y) position
Dx,y	D#,#	Tracing (x,y) position
Rx,y	R#,#	Movement relative to (x,y)
Jx,y	J#,#	Tracing relative to (x,y)
P<string>	P<string>	Printing of a character string
Lx	L#	Selection of line type by x
Cx	C#	Selection of pen x (colour change)
Sx	S#	Selection of character size by x
Qx	Q#	Selection of print direction by x (For instruction P only)

Functions of the "Graphic" Mode

The plotting functions that correspond to these functions require that the "graphic" mode be active. The functions, therefore, put the plotter in the "graphic " mode before carrying out the operation required, and leave it in this mode after execution.

Functions of the "Variable" Mode

The plotting functions that correspond to these functions require that the "graphic" mode be active. The functions, therefore, put the plotter in the "graphic" mode before carrying out the operation required. Nevertheless, it is sometimes necessary to carry out the said operation during a printing sequence using the "text" mode. The user, therefore, has a choice as to the mode in which the plotter is left after executing these functions.

UTILITIES GROUP

The functions of this group are of varied uses:

- manipulation of strings
- manipulation of numerical or string matrices of one or two dimensions
- numeric or alphanumeric sorting
- extended memory management (HP822180A XFUNCTIONS and HP822181A XMEMORY modules)
- many other applications ... !

/MOD — Euclidean Division

/MOD (**D**ivide **MOD**) determines the remainder (modulo) and quotient of a division, that is performs Euclidean division in its entirety. It is an extension of the [MOD] function of Catalogue 3.

Example 1: Calculation of the quotient and the remainder (modulo) of the division 13 by 3.

Press ON:	Display	
13	13_	Input of the dividend
[ENTER_] 3	3_	Input of the divisor
[XEQ] /MOD	1.000	Remainder (Modulo)
[X<>Y]	4.000	Quotient of the division
[LASTX]	3.000	The divisor is preserved in the L register

DETAILED INSTRUCTIONS FOR /MOD

1. For calculating the remainder (modulo) and the quotient of the division Y and X

2. [XEQ] "/MOD". The quotient and the remainder (modulo) of the division are placed in Y and in X respectively. The divisor is preserved in L, the dividend is lost. The contents of the T and Z registers are not altered.

3. If the X register contains zero, the calculator displays DATA ERROR.

The Stack

Input	Output
T: T	T: T
Z: Z	Z: Z
Y: Dividend	Y: Quotient
X: Divisor	X: Remainder (Modulo)
L: L	L: Divisor

APPLICATION PROGRAMS FOR /MOD

Example 2: Following is a fast process for calculating the decimals of the division of A by B where A < B and B ends in 9:

Following may be used for calculating 153 divided by 209

```
01 LBL "DIV9"          10 LBL 01
02 10                  11 RCL 01
03 /                   12 /MOD
04 INT                 13 VIEW Y
05 1                   14 10
06 +                   15 *
O7 STO 01              16 +
08 RDN                 17 GTO 01
09 SF 21               18 END
```

153 divided by 209 = 0.732057...

Example 3: [/MOD] can be used in a simple routine such as "changing to a lower base"! This short program, YBX, (Y to base X) in simplifying the calculation, produces the different figures for the new value in reverse order, that is to say, from the lower to the higher base. X and Y have to be integers.

```
01 LBL "YBX"          07 CLX
02 SIGN               08 X#Y?
03 LBL 00             09 GTO 00
04 X<>L               10 BEEP
05 /MOD               11 END
06 STOP
```

For example 1103 [ENTER_] 8 [XEQ] "YBX" gives 7 [R/S] 1 [R/S] 1 [R/S] 2 [R/S] 0. This shows that 1103 (DEC) = 2117 (OCT). This result can be verified using the OCT and DEC functions.

N.B.: If one wants to reconstitute the dividend by X<>Y LASTX * + for a quotient > 0 and by X<>Y X<0? DSE X NOP * LASTX * + for a quotient not equal to 0, one cannot guarantee a null quotient.

AD-LC — Line, Column Address

[**AD-LC**] (**AD**dress-**L**ine, **C**olumn) determines the coordinate lines and columns of an element of a matrix from its real address and from the matrix pointer.

Example: Calculate the coordinates of register 36 in Matrix A (below) whose pointer 25.04405 is in the R00 register. R25 = Number of the first register and R44 = Number of the highest register.

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

Press ON	Display
36 [ENTER_]	36.0000 Input of register number
[RCL] 00	25.04405 Recalling of the matrix index
[XEQ] "AD-LC"	2.00000 Column No. 2
[RDN]	3.00000 Line No. 3
[LASTX]	25.04405 The index is saved in L.

DETAILED INSTRUCTIONS FOR AD-LC

To calculate the line and column coordinates of an element in a matrix when the matrix pointer and the register occupied by this element are known, [ENTER_] the number of the register, [ENTER_] the matrix pointer and [XEQ] "AD-LC". The column number is returned into X and that of the line into Y. The pointer is preserved in L and the Z and T registers remain unchanged.

The Stack

Input	Output
T:	T: t
Z: z	Z: z
Y: register	Y: Line number.
X: matrix	X: column number.
L: L	L: matrix index

NOTE: This function does not verify if the register is part of the matrix. If the X or Y registers contain an ALPHA string, the calculator displays ALPHA DATA error message.

ALENG ALPHA Length

[**ALENG**] (Alpha LENGth) places in the X register, the number of characters of the string present in the ALPHA register.

Example 1: In the course of a programme, the HP-41 stops to receive ALPHANUMERIC data input by the user. The programme has to calculate the length of the string in order to arrange it into different registers. [Solution to this problem can be achieved using the ALENG function]

NB: Refer to the RGAX function in this manual for another solution to this problem.

DETAILED INSTRUCTIONS FOR ALENG

When the ALPHA register contains a string whose length has to be calculated, [ALENG] places the number of the characters in the X register and enables stack lift.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: x
X: x	X: Length of ALPHA string
L: l	L: l

APPLICATION PROGRAMS FOR ALENG

Example 2: The following routine replaces the lower case characters with upper case in the string present in the ALPHA register. It uses [ALENG] to determine the number of characters of the string (so long as the string does contain null character(s)).

```
01 LBL "CAP"
02 ALENG     Determines the no. of characters in the ALPHA string
03 LBL 00
04 ATOXL     Places in X the code of the first character
05 97        The codes of the lower characters are found between 97 and 122
06 X>Y?
07 GTO 01    If it is not lower case (< 97), go to LBL 01
08 CLX
09 122
10 X<Y?
11 GTO 01   If it is not a lower case (> 122)  go to LBL 01
12 CLX      One obtains capital letter code by subtracting
13 32       32  from the code of the corresponding lower case
14 -        letter (A=65,a=97 and 97-65=32)
15 R_
16 LBL 01
17 RDN
18 XTOAR    Places the capital letter to the right of the string
19 RDN
20 DSE X
21 GTO 00  Continues the loop to the end of the string
22 AON
23 END
```

ANUM Searching for a Number in the ALPHA Register

[**ANUM**] (**A**lpha to **NUM**ber): The [ANUM] function scans the contents of the ALPHA register from left to right in search of a number. The first number encountered is placed into the X register.

Example: If the ALPHA register contains the string PRICE: 1,234.50 obtained by reading the alphanumeric text of extended memory and extracting the numeric value for arithmetic manipulation, execution [XEQ] of ANUM places the number in the X register.

DETAILED INSTRUCTION FOR ANUM

1. The ANUM function searches for a numeric value contained in the string in the ALPHA register. If a number is found, the calculator places the number in the X register and sets flag 22. If the calculator does not find a number, the contents of the X register and the status of flag 22 are not changed.

2. The number in the ALPHA register can be in either format of digit separator representation. The separators "," and "." are interpreted in accordance with the status of flags 28 and 29. If the number in the ALPHA register is preceded by a minus sign, the calculator places a negative number into the X register at the time of execution of the function. Suppose that the ALPHA register contains the string in Example 1:

Flag 28	Flag 29	Display
Set	Set	1,234.5000
Set	Clear	1,0000
Clear	Set	1,2345
Clear	Clear	1,2340

THE STACK

Input	Output
T: t	T: z
Z: z	Z: y
Y: y	Y: x
X: x	X: number found in X
L: l	L: l

ANUMDEL Searching for a Number in ALPHA and Deletion

[**ANUMDEL**] (**A**lpha-to-**NUM**ber **DEL**ete) searches from left to right for a numerical value in the ALPHA register and places the first number it detects in the X register. It deletes this number from the ALPHA register as well as all the preceding characters.

Example 1: If the ALPHA register contains the string **PRICE:1,234.5 FRS**, to extract the numeric value for arithmetic operation, [XEQ] "ANUMDEL". This places the numeric portion of the string in the X register. The ALPHA string is deleted up to, and including the character "5".

DETAILED INSTRUCTIONS FOR ANUMDEL

1. The ANUMDEL function searches the contents of the ALPHA string for a numeric value. If the calculator finds a number, it places the number in the X register and deletes the string from the beginning to the end of the number.

2. If the string in the ALPHA register contains several numbers, separated by one or more non-numeric characters, [ANUMDEL] takes into account only the first number. [ANUMDEL] is identical to the [AMUM] function, except that [ANUM] does not alter the string in the ALPHA register. The HP-41 considers the execution of [ANUMDEL] as a numeric input, and sets flag 22 when it places the number into the X register. If the ALPHA register does not contain any numeric value, [ANUMDEL] deletes the ALPHA register, but does not alter the operational stack or the status of flag 22.

The characters "+" "-" "," "." and "E" (standing for exponent) are interpreted by [ANUMDEL] as numeric or non-numeric representations in accordance with the context in which they appear in the string. An isolated "+", for example, is not treated as a numeric character. A "+" or "-" immediately preceding, included in, or directly following a sequence of numbers will be interpreted as mere keys input from a keyboard (with [CHS] represented by "-"). For example, [ANUMDEL] returns the value -3425 if the function was executed when the ALPHA register contained the string "34-2+5".

3. In the ALPHA register numbers can represent values in any separator format. The digit separators "," or "." are digit interpreted in accordance with the status of flags 28 and 29. For example, if flags 28 and 29 are set, the comma is treated as a separator of a group of three figures. But the comma will be considered to be a non-numeric character if flag 28 is set and flag 29 is cleared.

Suppose that the ALPHA register contains the string as in Example No. 1: "PRIX:1,234.5 FRS and the calculator is placed in FIX 4 mode:

Flag 28	Flag 29	Number Displayed	New String
Set	Set	1,234.5000	FRS
Set	Clear	1,0000	,234.5FRS
Clear	Set	1,2345	FRS
Clear	Clear	1,2340	.5FRS

The Stack

Input	Output
T: t	T: z
Z: z	Z: y
Y: y	Y: x
X: x	X: Original value in ALPHA
L: l	L: l

APPLICATION PROGRAMS FOR ANUMDEL

Example 2: The HP7470A plotter when programmed returns an ASCII string describing the current position of the pen. This string contains 3 whole numbers separated by commas: X,Y and P. X is the ordinate of the position of the pen, Y is the abscissa, and P takes the value 0 or 1 depending on the position of the pen, whether it is raised or lowered. Supposing that the plotter places in the ALPHA register the value "123,456,1", a program can extract these values by executing [ANUMDEL] three consecutive times.

Press ON	Display	Comments/Remarks
[SF 28]		Ensures that the comma is not taken as a decimal separator
[CF 29]		Ensures that the comma is not considered as a separator of a group of three figures
[ANUMDEL]	123.0000	Ordinate
[ANUMDEL]	456.0000	Abscissa
[ANUMDEL]	1.0000	The pen is lowered

Example 3: The ALPHA register contains the string "34/-2/5"

[CF 28]	
[ANUMDEL]	34.0000
[ALPHA]	/-2/5
[ALPHA] [ANUMDEL]	-2,0000
[ALPHA]	/5
[ALPHA] [ANUMDEL]	5,0000

The above example shows that the symbols "/" and "*" are not interpreted like the symbols "+" "-" or "."

APPX Copying Integer Portion of Number in X into ALPHA

[**APPX**] [**AP**Pend **X**] appends the integer portion of the number in the X register to the right of the string in the ALPHA register.

Example: The result of a previous calculation present in the X register is 1,255.7 and the contents of the ALPHA register is "SURF:" [APPX] appends the integer portion of the number in the X register to the contents of the ALPHA register: the contents of the ALPHA register become "SURF:1,255"

DETAILED INSTRUCTIONS FOR APPX

1. [APPX] appends the integer portion of the number in the X register to the right of the string in the ALPHA register. [APPX] conforms to the status of flags 28 and 29. The number is operated upon in the FIX 0 mode, except that the decimal separator is not appended to the string and also the number is not rounded off. Like [ARCL], [APPX] does not generate any error message when its execution causes overflow in the ALPHA register.

If the X register contains an ALPHA string, the calculator dispalys the ALPHA DATA error message.

AROT Rotation of Contents of the ALPHA Register

[**AROT**] [**Al**pha **ROT**ation] effects rotation of the contents of the ALPHA register by the number of characters specified in the X register.

Example 1: The ALPHA register contains the string "AROT"; if one wishes to display in succession the strings "TARO" and then "ROTA" -

Press ON	Display
[ALPHA] AROT	AROT_
[ALPHA] 1 [CHS]	-1_
[XEQ] "AROT" [ALPHA]	TARO
[ALPHA] 2	2_
[XEQ] "AROT" [ALPHA]	ROTA

DETAILED INSTRUCTIONS FOR AROT

The [AROT] function effects a rotation of the contents of the ALPHA register by the number of characters specified in the X register (X modulo 24). The rotation takes place from the left if the content of the X register is positive and from the right if the content is negative. (See the Annexure at the end of this manual for more information on the effect of the [AROT] function on a string containing null characters).

The Stack

Execution of the [AROT] function does not modify the contents of the stack.

APPLICATION PROGRAMS FOR AROT

Example 2: The [AROT] function can be used with the [ANUM] and [POSA] functions for locating the occurrence of a character or of a string in the ALPHA register without change in the contents of the register.

Given that as a result of the operation of a device, the ALPHA register contains the sequence 68,2 69,88 (two numbers separated by a space). Let us suppose that we have to extract separately the two numbers to be used in a program. The following sequence of keystrokes illustrates the procedure:

Press ON	Display	
[CF] 28		
[XEQ] "ANUM"	68.2000	Places the first number in X
[STO] 20	68.2000	Stores for later use
32	32	Space code
[XEQ] "XTOAR"	32.0000	Adds a space to the right of the ALPHA register
[XEQ] "POSA"	4.000	Searches for the first space in the ALPHA register
[XEQ] "AROT"	4.000	Carries out a string rotation; rotation; ALPHA contains 69.88 68.2; in 69.88 68.2 in the absence of a space the ALPHA register will contain 69.8868.2
[XEQ] "ANUM"	69.8800	Places 69.88 in the X register

Transfer of Characters between the ALPHA and X Registers

ATOXL Transfer of the leftmost Character of ALPHA into X

[**ATOXL**] (**A**lpha-**TO**-**X** **L**eft) pulls out the first character of the ALPHA string and places its decimal code into the X register.

ATOXR Transfer of the rigthmost character of ALPHA into X

[**ATOXR**] (**A**lpha-**TO**-**X** **R**ight) extracts the last character of the string in the ALPHA register and places its decimal code in the X register.

ATOXX Transfer of a given ALPHA character into X

[**ATOXX**] (**A**lpha-**TO**-**X** by **X**) places in X the code of the character specified in the X register.

DETAILED INSTRUCTIONS FOR ATOXL, ATOXR and ATOXX

1. [**ATOXL**] deletes the leftmost character contained in the ALPHA register and places its decimal code in the X register. If the first character is followed by one or more null characters, these nulls become leading nulls and are erased from the string up to the first non-null character. If the ALPHA register is empty [ATOXL] places -1 into the X register.

2. [**ATOXR**] deletes the rightmost character of the string in the ALPHA register and places its decimal code in the X register. If the ALPHA register is empty, [ATOXR] places -1 into the X register.

3. [**ATOXX**] searches for the character whose position is specified by the number in the X register and returns its decimal character code into the X register. The string in the ALPHA register is not changed.

A positive value placed in the X register indicates a position in the ALPHA register counting from left to right from the first non-null character. This first character occupies the 0 position. This convention is identical to that used by the [POSA] function of the Extended Functions Memory Module.

On the other hand, a negative value placed in the X register indicates an absolute position in the ALPHA register. The positions are therefore counted from right to left starting from -1 for the character at the extreme right and going up to -24 for the extreme left position. The following table summarizes the way [ATOXX] interpretes the position of the characters.

Position of the Character	Character |
n > n >= length of string	Not available (DATA ERROR)|
0 <= n < length of string	Nth character according to the most exteme left character
n=0	The first character of the string from the left
-24 <= n < 0	The nth character from the from the right to the end of the register
n < -24	Not available (DATA ERROR)

If the X register contains an ALPHA string, the calculator displays the error message, ALPHA DATA.

Example: In this example, the whole of the ALPHA register is shown, nulls occupying the left part are shown as horizontal dashes for easy understanding, but caanot be displayed by the calculator.

Press	Display	
[ALPHA] DECAMETRE [ALPHA]	DECAMETRE	
0 [XEQ] ATOXX	68,000	"D" Code
4 [XEQ] ATOXX	77,000	"M" Code
6 [CHS] [XEQ] ATOXX	65,000	"A" Code
10 [CHS] [XEQ] ATOXX	0,000	Null character

BLDPT Building of Index

[**BLDPT**] (**B**ui**LD** **P**oin**T**er) builds an index of the form bbb.eeeii if X is positive or a matrix index if X is negative.

Example 1. The results of a calculation had placed in the Z register the number in the first register of a collection of values, into Y the last register and into X the number of registers separating each consecutive values; say, Z = 25, Y = 40 and X = 5.

To calculate the index, [XEQ] "BLDPT", [FIX] 5
X = 25.04005 will lead to registers R25, R30 and R40

Example 2. The result of a calculation has placed in the Z register the number of the first register of a matrix, in the Y register the number of rows and the in the X register the number of columns; i.e. Z = 25, Y = 4 and X = 5.

To calculate the matrix index, [CHS] [XEQ] "BLDPT", X = 25.04405

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

COMPLETE INSTRUCTIONS FOR BLDPT

1. To build an index of the form bb.eeeii:

- put bbb in the Z register
- put eee in the Y register
- put ii in the X register
- Execute [BLDPT].

2. To build a matrix index bbb.eeecc where bbb is a number representing the number of the first register in the matrix, eee is a number representing the number of the last register and cc is a number representing the number of columns of the matrix:

- place bbb in the Z register
- place the number of rows (|||) of the matrix in the Y register
- place the number of columns cc of the matrix in the X register and change its sign to negative
- Execute [BLDPT]

NOTE: If the X, Y or Z register contains an ALPHA string, the calculator displays the ALPHA DATA error message. The index is built by taking the absolute values of bbb and eee.

The Stack:

For x > 0		For x < 0	
Input	Output	Input	Output
T: t	T: t	T: t	T: t
Z: bbb	Z: t	Z: bbb	Z: t
Y: eee	Y: t	Y: |||	Y: t
X; ii	X: bbb.eee	X: cc	X: bbb.eeecc
L: l	L: eee	L: l	L: ||

BRKPT Analysis of index of a matrix

[**BRKPT**] (**BR**eak **P**oin**T**er) breaks up a bbb.eeeii index if X is positive or a matrix index, if X is negative.

Example 1. In a calculation it is required to know the elements of a bbb.eeeii index where bbb is the first register of a collection of values, where eee is the last register and where ii is the number of registers separating each value.
X = 25.04005 points to the registers R25, R30, R35 and R40, [XEQ] BRKPT, which restores Z=25, Y=40 and X=5.

Example 2. The index of a matrix is 25.04405, indicating that the matrix starts at R25 register, extends to R44 register and that the matrix is made up of 5 columns. The number of rows of the matrix is obtained by: [CHS] [XEQ] BRKPT which results in Z=25 (first register), Y=4 (number of rows), X = -5 (number of columns)

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

DETAILED INSTRUCTIONS FOR BRKPT

1. To break up an index of the bbb.eeeii form where bbb is a number between 0 and 999 representing the first element of a loop or of a vector, where eee is a number between 0 and 999 representing the last element and where ii is a positive number between 0 and 99 representing the "interval", it is advisable to ensure that the value placed in the X register is positive, by means of [XEQ] [ABS] for example, then executing BRKPT will place in Z the integer portion of the value found initially in X, and Y with the first three figures of the decimal part, and X with the 4th and 5th figures of the decimal portion. The index is retained in the L register.

2. To break up a bbb.eeecc matrix index where bbb is the first register of the matrix, where eee is the last register and cc is the number of columns, it is advisable to ensure that the value placed in the X register is negative by executing [ABS] followed by [CHS] for example, then executing BKRPT returns separately the first register (bbb) in Z, the number of rows (|||) = (eee + 1 - bbb)/cc in Y and the number of columns (cc) in X.

NOTE: If the X register contains an ALPHA string, the calculator will display the error message ALPHA DATA.

The Stack:

For x > 0		For x < 0	
Input	Output	Input	Output
T: t	T: x	T: t	T: x
Z: z	Z: bbb	Z: z	Z: bbb
Y: y	Y: eee	Y: y	Y: ll
X: bbb.eeeii	X: ii	X: bbb.eeecc	X: cc
L: l	L: bbb.eeeii	L: l	L: bbb.eeecc

where eee = (ll*|cc|-1+bbb)

CHFLAG Saving a Predefined Status of Flags

[**CHFLAG**] (**CH**arge **FLAG**s) saves the status of of the flags in the calculator prior to the execution of a program.

Example 1. It is required that starts by initializing the calculator for it to be in DEGrees mode, ENGineering notation format with 3 figures and the first 5 flags (0 to 4) set.

In RUN mode (PRGM indicator off) set the calculator to the above desired status, then switch to PRGM mode. Executing CHFLAG writes 2 lines into the program: the first line containing the instruction CHFLAG, the next line a string of seven characters. At the execution of the program the calculator is returned to its previously set configuration.

FULL INSTRUCTIONS FOR CHFLAG

1. In calculation mode set the calculator to the desired status prior to executing the program.

2. In PRoGraM mode, [XEQ] CHFLAG. This would write two lines of programme where the first line contains the instruction CHFLAG, the next line contains a string of 7 characters representing the set configuration. This string starts with an identifier which permits the calculator to keep track of the required configuration. If this string was altered or replaced by another which does not conform with the original configuration, the execution of CHFLAG in the course of running the program will bring about a stop of the program and the display of the message: CHFLAG ERR.

The Stack

The stack is not affected by the execution of CHFLAG.

NB: The ALPHA register is not modified by the execution of CHFLAG. The string of characters is a representation of the flags, it is not intended for the ALPHA register.

It is not necessary to precede CHFLAG with a test instruction (such as ISG or X = Y ?).

Example:

FS? 01	Checks if the flag is set
CHFLAG	Initializes the calculator
`......'	String representing the initial status

If the test is negative (FLAG 01 clear), this causes replacement of the contents of the ALPHA register with the string representing the initial status of the calculator. Only the flags 00 43 are affected by CHFLAG.

Flags 00 to 10 are the flags reserved for the user
Flag 11 Automatic execution of a program when set or after loading the program from mass storage.
Flags 12 to 20 Control of external devices
Flags 12 and 13, 15 and 16 are used by printers
Flag 12 Double width characters
Flag 13 Lower case letters
Flags 15   and 16   Printing modes of the HPIL printer

Clear	Clear	Manual mode
Clear	Set	Normal mode
Set	Clear	Trace mode
Set	Set	Trace and contents of the stack

Flag 17 Ignore CR-LF
Flag 18
Flag 19 [Use dependent on external device present]
Flag 20
Flag 21 Enable printing
Flag 22 Set by a numeric input
Flag 23 Set by alpha input
Flag 24 Ignore out of range error
Flag 25 Error (first occurrence) ignore
Flag 26 Tone (or beep) enable
Flag 27 User keyboard
Flag 28 Type of decimal separator
Flag 29 Presence or absence of separator of groups of three   figures
Flag 31 DMY mode of TIME module
Flag 32 MANIO mode of the HPIL module
Flag 34 ADROFF mode of the Extended I/O module
Flag 35 Auto starting inhibition (Autostart/Duplication module)
Flag 36 through Flag 39 Number of figures for FIX, SCI or ENG
Flag 40 and Flag 41 Display mode
Flag 42 and Flag 43 Angular mode

CLINC   Clearing the Increment Value of an Index

[**CLINC**] (CLear INCrement) truncates the value of an index in the X register from the 4th figure of the decimal portion.

Example 1. To have access to all the values of a matrix whose index 25.04405 is stored in R00 -

Press on	Display	
[RCL] 00	25.04405	Recalls the index value
[XEQ] CLINC	25.04400	
[XEQ] INT	25.00000	1st element of the index
[LASTX]	25.04400	
[XEQ] FRC	0.04400	
[EEX] 3 [*]	44.00000	Last element of the index

FULL INSTRUCTIONS FOR CLINC

[CLINC] replaces the value in X from the 4th figure (inclusive) of the decimal portion with 0s. The original value is retained in the L register.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: bbb.eeeii	X: bbb.eee
L: l	L: bbb.eeeii

NOTE: If the X register contains an ALPHA string, the calculator displays the error message, ALPHA DATA.

COLPT Calculation of a Matrix Index

[**COLPT**] (**COL**umn **P**oin**T**er), given a column number in the Y register and a matrix index in the X register, calculates a column index.

Example: In order to access the registers containing the index to the second column of the Matrix A below, and whose index is retained in the R00 register -

Press on	Display	
2	2_	Column number
[RCL] 00	25.044005	Recalling of the index
[XEQ] COLPT	26.04105	Index of the second column

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

FULL INSTRUCTIONS FOR COLPT

1. Determine the column number whose index is required.
2. Place in the X register the index of the matrix to which the column belongs.
3. [XEQ] COLPT which places in X the column index and restores the matrix index in the L register.

The Stack

Input:	Output:
T: t	T: t
Z: z	Z: t
Y: Column number	Y: z
X: bbb.eeeii	X:b'b'b'.e'e'i'i'
L: l	L: bbb.eeeii

NB: i'i' = ii

GETRBX Recalling of registers from Extended Memory

[**GETRGX**] (**GET R**e**G**isters by **X**) copies into registers specified in X, the contents of the registers of the current index (a control number containing the index) from the position of the index indicated by the increment specified in X.

Example: The control number constituting the index in R10 is 25.0440510, executing GETRGX copies the contents of the registers 10, 20, 30, ... from the index in extended memory into registers 25, 30, 35, ... of main memory.

FULL INSTRUCTIONS FOR GETRGX

1. It is advisable to ensure that the control number of the current index is in the required position by means of [SEEKPT] or [SEEKPTA].

2. The control number placed in the X register is a number in the form bbb.eeeiijj where bbb is the first register of main memory into which it is required to recopy the registers of the extended memory, where eee is the last register where it is desired to make the copy, where ii is the interval between two registers of the main memory and finally where jj is the interval between 2 registers copied successively from extended memory.

3. Executing [GETRGX] copies the designated registers from the current index into main memory.

The Stack:

The stack is not altered by the execution of [GETRGX].

APPLICATION PROGRAMMES FOR GETRGX

The figures below represent two matrices, the one on the left is put into main memory while the one on the right is put into extended memory. In each case, the register number and the contents of the register (represented by a letter) are indicated.

Start by putting the control index number in the first register to be recopied from extended memory by executing [SSEKPT].

To copy the registers of the second column of the matrix B extended memory into the third column of matrix A in main memory, it is sufficient to place in X the control number of the third column of matrix A (27.04205) completed by the interval of the register to be read in extended memory R03), i.e. X = 27.0420503.

27 = bbb, the 1st register of the vector in main memory
42 = eee, the last register of the vector in main memory
05 = ii, the interval between two registers of the vector in main memory
03 = jj, the interval between 2 registers read into extended memory.

Executing GETRGX recopies the registers corresponding to the index placed in X; the result is represented in the second figure.

Figure I : Starting Position

MATRIX A in Main Memory					
Col.	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25 A	R26 B	R27 C	R28 D	R29 E
Row 2	R30 F	R31 G	R32 H	R33 I	R34 J
Row 3	R35 K	R36 L	R37 M	R38 N	R39 O
Row 4	R40 Q	R41 R	R42 S	R43 T	R44 U

MATRIX B in Extended Memory			
	No. 1	No. 2	No. 3
Row 1	R11 a	R12 b	R13 c
Row 2	R14 d	R15 e	R16 f
Row 3	R17 g	R18 h	R19 I
Row 4	R20 j	R21 k	R22 l

Figure II : Result after Execution of GETRGX

MATRIX A in Main Memory					
Col.	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25 A	R26 B	R27 b	R28 D	R29 E
Row 2	R30 F	R31 G	R32 e	R33 I	R34 J
Row 3	R35 K	R36 L	R37 h	R38 N	R39 O
Row 4	R40 Q	R41 R	R42 k	R43 T	R44 U

MATRIX B in Extended Memory			
	No. 1	No. 2	No. 3
Row 1	R11 a	R12 b	R13 c
Row 2	R14 d	R15 e	R16 f
Row 3	R17 g	R18 h	R19 I
Row 4	R20 j	R21 k	R22 l

LC-AD Calculation of address of a matrix element

[**LC-AD**] (**L**ine-**C**olumn-**AD**dress) determines the register number of an element of a matrix from the row coordinatee, column and the matrix index.

Example: Determination of the register number of the element placed in row 2 and column 3 of matrix A whose index 25.04405 is in the register, R00.

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

Press on	Display	
2 [ENTER]	2.0000	Introduction of row number
3	3_	Introduction of column number
RCL 00	25.04405	Recalling of the matrix index
XEQ LC-AD	32.00000	Register number being sought

FULL INSTRUCTIONS FOR LC-AD

To determine the register number of a matrix element whose index, row number and column number are known: determine the row number, ENTER, column number, ENTER, matrix index. Executing LC-AD returns the register number in X and retains the index in L.

The Stack

Input:	Output:
T: T	T: T
Z: row number	Z: T
Y: Column number	Y: T
X: Matrix index	X: Register number
L: L	L: Matrix index

LINPT Calculation of the row index of a matrix

[**LINPT**] (**LIN**e **P**oin**T**er), given the row number in the Y register and the matrix index in the X register, calculates the row index.

Example: In order to have access to the registers of registers of the second row of matrix A, whose index is in the R00 register:

Press on	Display	
2	2_	Row number
RCL 00	25.04405	Recalling of the index
XEQ LINPT	30.03400	Index of the second row

MATRIX A

Column	No. 1	No. 2	No. 3	No. 4	No. 5
Row 1	R25	R26	R27	R28	R29
Row 2	R30	R31	R32	R33	R34
Row 3	R35	R36	R37	R38	R39
Row 4	R40	R41	R42	R43	R44

DETAILED INSTRUCTIONS FOR LINPT

1. Determine the row number whose index is being searched for.

2. Place in the X register the index of the matrix where the row belongs.

3. Executing LINPT places in X the row index and saves the matrix index in the L register.

The Stack

Input:	Output:
T: t	T: t
Z: z	Z: t
Y: row number	Y: z
X: bbb.eeeii	X:b'b'b'.e'e'i'i'
L: l	L: bbb.eeeii

NOP No operation

[**NOP**] (**N**o **OP**eration) is meant to follow an programme instruction comprising a test with conditional branching when the branching is not to be effected.

Example: In the course of a loop, it required to increase the contents of the Y register and that of the X register.

One may write the following lines into a program:

```
ISG Y      Increments the Y register
NOP        Renders inoperative the resulting skip
ISG X      Increments the X register
GTO 03       and loops if greater
```

POSA Searching for the position of a character in ALPHA

[**POSA**] (**POS**ition in **A**lpha) scans through the ALPHA register from left to right in search for the character or string specified in the X register.

Example 1: The string "ABCDEFGHIJ" is in the ALPHA register, what is the position of the character "D"?

Press on	Display	
68	68_	Character code for D
[XEQ] POSA	3.0000	Position of character D in the ALPHA register

Example 2:

Press on	Display
CLA	
[ALPHA] DEF [ALPHA]	
[ASTO] . X	DEF
[ALPHA] ABCDEFGHIJ [ALPHA]	DEF
[XEQ] POSA	3 Position of character D in the ALPHA register

DETAILED INSTRUCTIONS FOR POSA

1. [POSA] scans the ALPHA register from left to right in search of a character or string in the X register. The string can be specified in 2 ways: by entering the code of the single character or by placing the character or string into the X register with the help of the [ASTO] [.] [X] sequence. If the calculator finds the string in the ALPHA register it places the position of the first character into the X register.

2. The positions are counted from left to right starting from position 0. If the string or the character appears several times in the ALPHA register the calculator gives only the position of the first occurrence. If the string or the character does not exist in the ALPHA register, the calculator returns the value -1.

3. The string or the character code is saved in the LASTx register.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: code or string	X: position in ALPHA register
L: l	L: code or string

FUNCTIONS FOR THE ALLOCATION OF MEMORY

PSIZE Allocation of memory registers in programmes

[**PSIZE**] (**P**rogrammable **SIZE**) allocates for data, from within a program, the number of registers specified in the X register

SIZE? Determination of allocated memory

[**SIZE?**] places in the X register the number of memory registers allocated to data at the time of its execution.

The [SIZE?] and [PSIZE] functions can be employed in the same program to reallocate a collection of registers without destroying data.

Example:

```
01 ...
02 ...
   ...)
   ...)  (Program)
07 SIZE?      The calculator places in X the number of registers allocated to data
08 125        The new program requires 125 registers of data. The previous result is in
              the Y register.
09 X>Y ?      Is the number of registers required greater the actual number allocated?
10 PSIZE      If yes, reallocate the memory registers.
```

READEM Reading of files into extended memory from mass storage

[**READEM**] (**READ E**xtended **M**emory) copies from mass storage (HP82161A cassette reader for example) the contents of the extended memory previously stored there by way of the WRTEM function.

Example 1. To load the "MAT3" file stored on the cassette:

Press On	Display	
[XEQ] EMDIR	DIR EMPTY	Confirms that extended memory is empty.

In the case where 2 XMEMORY modules are present, the number of available registers is therefore 600.

[ALPHA] MAT3 [ALPHA]	600.0000	ALPHA contains the generic names of the files to be read.
[XEQ] READEM	600.0000	The files are loaded into extended memory.

```
[XEQ] EMDIR   MATRP       P012   All these files have been read by
              A           D100   READEM
              TEXT        A040
              ...
```

DETAILED INSTRUCTIONS FOR READEM

1. After inputting the generic name of the file to be read, executing READMEM copies the specified file from the cassette into extended memory.

2. If the HPIL module is not connected, NOHPIL error message is displayed by the calculator.

3. If the file is not on the cassette, the error message, FL NOT FOUND is displayed.

4. If the extended memory space is not sufficient, the calculator displays NO ROOM. In this case, add one or two XMEMORY modules.

5. If the HPIL module is connected but the cassette reader is missing from the loop, the NO DRIVE message is displayed and the execution of the function is terminated.

6. If the file is not the type created by WRTEM, FL TYPE ERR is displayed and the execution of the function is terminated.

NB: [READEM] erases all the files previously stored in extended memory and reads in the file or files from the external source.

The Stack:

The stack is not altered by [READEM]

REVERSE FUNCTION: WRTEM

RG — Key prefix for RG function

[**RG**] is a function intended to facilitate the entry via the keyboard names of functions starting with RG. This function is used essentially to assign a key. For example, RG assigned to the [LN] key.

ASN "RG" 15 (Press on [Shift Key][ASN][ALPHA][R][G][ALPHA][LN]. Place the calculator in USER mode. Thereafter to execute or program or function whose name begins with RG (RGVIEW for example), press on:

[RG][Key LN][ALPHA]VIEW[ALPHA]

This sequence is equivalent to:

[XEQ][ALPHA][R][G][V][I][E][W][ALPHA]

You thereby save 2 key depressions each time you use a function beginning with the 2 letters RG.

DETAILED INSTRUCTIONS FOR RG

1. Assign RG to a key and place the calculator in USER mode.

2. To execute or program a function whose name starts with RG, press successively on:

[RG] (previously assigned to a key)
[ALPHA]
... Remaining characters of the name of the function
... (for example SUM for the RGSUM function)
...
[ALPHA]

OPERATIONS BETWEEN REGISTERS

RG+- Sum of difference, term by term, of 2 vectors

[**RG+-**] (**R**e**G**isters **+** or **-**) adds or subtracts, term by term, the elements of 2 vectors whose indices are specified in the Y and X registers. The sign of the value in X determines the type of operation to be carried out.

RG* Multiplication, term by term, of 2 vectors

[**RG***] (**R**e**G**ister *****) multiplies, term by term, the elements of two vectors whose indices are specified in the Y and X registers.

RG/ Division, term by term, of 2 vectors

[**RG/**] (**R**e**G**isters **/**) divides, term by term, the elements of 2 vectors whose indices are specified in the Y and X registers.

Example: In the matrix below -

-- Replace the first column with the sum of the terms of the 3rd column and those of the first column;

-- then calculate the squares of the 4th column;

-- finally, divide each of these squares by the first 4 values of the first row.

The matrix index is retained in the R00 register.

Matrix before execution:
NB: Each space contains its register number and its initial contents.

MATRIX B

Col	1	2	3	4	5
Row 1	R25 142	R26 20	R27 857	R28 40	R29 1
Row 2	R30 285	R31 12	R32 714	R33 14	R34 2
Row 3	R35 428	R36 22	R37 571	R38 24	R39 3
Row 4	R40 714	R41 32	R42 285	R43 34	R44 4

```
Press on          Display

[CF]28[FIX]5
[1][RCL]00         25.04405
[COLPT]            25.04005     Index of the first column
[3][RCL]00         25.04405
[COLPT]            27.04205     Index of the third column
[XEQ] RG+-          25.04005     Index of the vector where the results are arranged.
```

At this stage, one can verify that the registers R25, R30, R35 and R40 which constitute the first column all contain 999.

```
[4][RCL]00         25.04405
[COLPT][ENTER]      28.04305       X and Y contain the index
                       of the 4th column
[RG] *            28.04305
```

At this point, the elements of the 4th column are:

R28 = 1600 R33 = 196 R38 = 576 R43 = 1.156

```
[1][RCL]00         25.04405
[LINPT]           25.02900      Index of the first row
[XEQ] RG/           28.04305
```

Finally, the 4th column contains the results of the division and the matrix is shown thus:

MATRIX C

Col	1	2	3	4	5
Row 1	R25 999	R26 20	R27 857	R28 1.60	R29 1
Row 2	R30 999	R31 12	R32 714	R33 9.80	R34 2
Row 3	R35 999	R36 22	R37 571	R38 0.67	R39 3
Row 4	R40 999	R41 32	R42 285	R43 722	R44 4

DETAILED INSTRUCTIONS FOR RG+- RG* RG/

1. The [RG+-], [RG*] and [RG/] functions require two indices: the index for the operands in Y and the index for the operators in X.

2. The results are loaded into the registers specified by the index placed in the Y register.

3. After performing the calculations, the X register contains the index where the results have been arranged and the L register contains the index of the operators.

The Stack

Input	Output
T: t	T: t
Z: z	Z: t
Y: Index no. 1	Y: z
X: Index no. 2	X: Index no. 1
L: l	L: Index no. 2

SCALAR OPERATION IN REGISTERS

RG+Y Addition of a constant to a group of registers

[**RG+Y**] (**R**e**G**isters **+ Y**) adds the value contained in the Y register to the contents of the registers specified in X.

RG*Y Multiplication of a group of registers

[**RG*Y**] (**R**e**G**isters *** Y**) multiplies the contents of the registers specified in X by the value in Y.

RG/Y Division of a group of registers

[**RG/Y**] (**R**e**G**isters **/ Y**) divides the contents of the registers specified in X by the value in Y.

Example: In Matrix B below:

- Subtract the constant 5 from the contents of of the first column;
- Calculate the double of the elements of the third row;
- Divide each term of the 5th column by 6.

The matrix index is retained in the R00 register.

MATRIX B

Col	1	2	3	4	5
Row 1	R25 1	R26 2	R27 3	R28 4	R29 5
Row 2	R30 6	R31 7	R32 8	R33 9	R34 10
Row 3	R35 11	R36 12	R37 13	R38 14	R39 15
Row 4	R40 16	R41 17	R42 18	R43 19	R44 20

Press on	Display	
5 [CHS] [ENTER_]	-5.0000	Input of the constant
1 [RCL] 00	25.04405	
[COLPT]	25.04405	Index of the first column
[RG] + Y	25.04405	Index of the vector where the results are arranged.

At this point, the 3rd row contains the double of the preceding values R35 = 12 R36 = 24 R37 = 26 R38 = 28 R39 = 30

```
6 [ENTER_]           6.00000     Input of the constant
5 [RCL] 00          25.04405
[COLPT]             29.04405     Index of the 5th column
[RG] / Y            29.04405
```

Finally, the 5th column contains the results of the division, and the resulting matrix is shown below:

MATRIX B

Col	1	2	3	4	5
Row 1	R25 -4	R26 2	R27 3	R28 4	R29 0.83
Row 2	R30 1	R31 7	R32 8	R33 9	R34 1.66
Row 3	R35 12	R36 24	R37 26	R38 28	R39 5
Row 4	R40 11	R41 17	R42 18	R43 19	R44 3.33

DETAILED INSTRUCTIONS FOR RG+Y, RG*Y, RG/Y

1. The functions [RG+Y], [RG*Y] and [RG/Y] require an index in X and a value in Y.

2. The results from calculations are placed in the registers specified by the index in the X register, that is to say, they replace the values on which the calculations were carried out.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: scalar value	Y: scalar value
X: Index	X: Index
L: l	L: l

The stack is not altered by [RG+Y], [RG*Y] and [RG/Y].

RGAX Registers to ALPHA or ALPHA to registers

[**RGAX**] (**R**e**G**isters-**A**LPHA by **X**) comprise 2 functions:

1. If X < 0 copies the ALPHA register into the registers specified by the index in X;

2. If X > 0, the register specified by the index in X are copied in conformity with the string in the ALPHA register.

Example - The string ABCDEFGHIJKLMONPQRSTUVWXYZ is in the ALPHA register. To retain the string in pairs of regiaters starting from R10, proceed as follows:

```
        Press on        Display

10.00002 [CHS]      -10.00002_    Index. The negative value                  loading into
the registers.
```

[RG] AX	-17.00002	The index indicates the last register occupied in memory following the loading process
[RCL] 10	ABCDEF	First six characters
[RCL] 12	GHIJKL	Next 6 characters
[RCL] 14	MNOPQR	Next 6 characters
[RCL] 16	STUVWX	Last 6 characters

If it is desired, the following process will replace the ALPHA register with the contents of registers R12 and R16.

12.00004	12.00004	Index for recall of the string
[XEQ] CLA	12.00004	Clears the contents of the ALPHA register
[RG] AX	17.00004	Index to the next register
[ALPHA]	CHIJKLSTUVWX	Note that the loading terminates with the last character of the string.

DETAILED INSTRUCTIONS FOR RGAX

1. The [RGAX] function cab be used to fill the whole of the ALPHA register with the registers indicated in X by the index. In this case, the index has to be a negative value. At the time of loading, the calculator sets an end of string marker intended for re-reading the last register used. This marker is not visible. However, an alteration of the contents of the last register results in a loss of this reference.

2. The [RGAX] function can equally be used for recalling of a string present in a range of registers. In this case, the index has to be positive. The loading of the string is done in conformity with the characters already present in the ALPHA register. If the new string comprises more that 24 characters, only the last 24 characters will remain in the ALPHA register. The preceding characters to the left are lost. The loading is carried out until the end of string marker is found by the calculator (see the preceding paragraph), or, if the calculator does not find the end of string reference, until it finds a numeric value in the register. In this case, the numeric value is loaded in the current format in the same way that it is done with [ARCL].

3. Whatever form of use, [RGAX] places the initial index in the L register and an a+-bbb.eeeii index into the X register where bbb is the number of the last register used plus 1, and the fractional part is that of the initial index. On the other hand, the first three figures after the comma can be any character. since [RGAX] does not take these into account.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: Initial index	X: New index
L: l	L: Initial index

RGCOPY Copying or exchange of contents of registers

[**RGCOPY**] (**R**e**G**isters **COPY**) functions in two modes:

If X >= 0, [RGCOPY] copies the contents of the registers designated by the index placed in X, into those specified by the index in Y.

If X < 0, [RGCOPY] exchanges the registers specified in X with those designated in Y.

Example: In the matrix B below, copy the contents of the registers of the first column into the registers of the 3rd column, then exchange the contents of the 2nd column with those of the first row.

MATRIX B

Col	1	2	3	4	5
Row 1	R25 1	R26 2	R27 3	R28 4	R29 5
Row 2	R30 6	R31 7	R32 8	R33 9	R34 10
Row 3	R35 11	R36 12	R37 13	R38 14	R39 15
Row 4	R40 16	R41 17	R42 18	R43 19	R44 20

It is to be assumed that the index of this matrix is contained in the R00 register.

Press on	Display	Remarks
3 [RCL]00[COLPT]	27.04205	Index of the destination registers
1 [LASTx][COLPT]	25.04005	Index of the original registers
[XEQ] RGCOPY	27.04205	Index of the new contents
[RGVIEW]		Lists the contents of the 3rd row R27 = 1, ...,R42 = 16.
1[RCL]00[LINPT]	25.02905	1st index
2[LASTx][COLPT][CHS]	-26.04105	2nd index
[RG] COPY	25.02905	The stack drops

The matrix is now as expected:

MATRIX B

Col	1	2	3	4	5
Row 1	R25 2	R26 7	R27 12	R28 17	R29 5
Row 2	R30 6	R31 1	R32 6	R33 9	R34 10
Row 3	R35 11	R36 1	R37 11	R38 14	R39 15
Row 4	R40 16	R41 4	R42 16	R43 19	R44 20

DETAILED INSTRUCTIONS FOR RGCOPY

1. The sign of the index placed in X determines if the registers are copied (X >= 0) or exchanged (X < 0).

2. The copying is carried out from the registers designated by the index placed in X into those designated by the index in the Y register. After execution, the stack drops.

3. The exchange takes place between the registers designated in X and Y. After execution, the stack drops. If the range of registers do not overlap, the exchange is carried out by starting with the lower number of registers. If there is an overlapping, the calculator determines the point at which it is to proceed with the exchange in order not to lose data.

The Stack

Input	Output
T: t	T: t
Z: z	Z: t
Y: destination index	Y: z
X: index of origin	X: destination index
L: l	L: index of origin

RGINIT Initialization of a range of registers

[**RGINIT**] (**R**e**G**isters **INIT**ialize) has 2 functioning modes:

If X >= 0 [RGINIT] places the contents of the whole range of registers into the registers specified by the index in X.

Example: In matrix B below whose index is retained in the R00 register, columns 3 and 5 will be cleared to zeros and then filled with the numbers 1 to 5 in the first row.

MATRIX B

Col	1	2	3	4	5
Row 1	R25 a	R26 b	R27 c	R28 d	R29 e
Row 2	R30 f	R31 g	R32 h	R33 i	R34 j
Row 3	R35 k	R36 l	R37 m	R38 n	R39 o
Row 4	R40 p	R41 q	R42 r	R43 s	R44 t

Press on	Display	Remarks
3[RCL][COLPT]	27.04205	Index of the 3rd column
[XEQ] RGINIT	27.04205	Clears the 3rd column to zero
5[LASTx][COLPT]	27.04205	Index of the 5th column
[XEQ] RGINIT	27.04205	Clears the 5th column to zero
1[LASTx][LINPT][CHS]	-25.02900	The negative sign indicates
[XEQ] RGINIT	-25.04400	an initialization with whole numbers of 1 to N

MATRIX B

Col	1	2	3	4	5
Row 1	R25 1	R26 2	R27 3	R28 4	R29 5
Row 2	R30 f	R31 g	R32 0	R33 i	R34 0
Row 3	R35 k	R36 l	R37 0	R38 n	R39 0
Row 4	R40 p	R41 q	R42 0	R43 s	R44 0

DETAILED INSTRUCTIONS FOR RGINIT

1. When the index placed in the X register is positive, the specified registers are cleared to zero.

2. When the index placed in the X register is negative, the designated registers are loaded successively with the numbers 1 to N.

The Stack:

The execution of [RGINIT] function does not alter the stack.

<u>RHNb Number of registers</u>

[RGNb] (**R**e**G**isters, **N**um**b**er of) calculates the number of registers specified by the index placed in X.

Example: To know the number of elements of a matrix whose index is retained in the R00 register, and then the number of registers contained in a row:

Press on	Display	Remarks
[RCL]00[CLINC]	25.04400	Index of the registers
[XEQ] RGNb	20.00000	The matrix is made up of 20 registers
1[RCL]00[LINPT]	25.02900	Row index
[XEQ] RGNb	5.00000	A row spans 5 registers

DETAILED INSTRUCTIONS FOR RGNb

[RGNb] places in X the number of elements designated by an index of the form bbb.eeeii placed in the X register. The index is retained in the L register.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: index	X: number of elements
L: l	L: index

RGSUM Sum of registers

[**RGSUM**] (**R**e**G**isters, **SUM** of) places in X the sum of the contents of the registers specified by the index placed in X.

Example: In matrix F below whose index is in R00, one is looking for the sum of the elements of the first column and the sum of the absolute value of the 4th column.

MATRIX F

Col	1	2	3	4	5
Row 1	R25 -14	R26 15	R27 25	R28 2	R29 8
Row 2	R30 7	R31 13	R32 19	R33 20	R34 1
Row 3	R35 0	R36 6	R37 12	R38 18	R39 24
Row 4	R40 23	R41 4	R42 5	R43 11	R44 17
Row 5	R45 16	R46 22	R47 3	R48 9	R49 10

Press on	Display	Remarks
1[RCL]00[COLPT]	25.04505	Index of the first column
[RGSUM]	32.00000	Sum of the elements
4[RCL]00[COLPT]	28.04805	Index of the 4th column
[CHS]	-28.04805	Index for absolute value
[RGSUM]	60.00000	Sum of the absolute values

DETAILED INSTRUCTIONS FOR RGSUM

[RGSUM] returns in X the sum of the elements specified by the index placed in the X register. If the index is negative, the calculator effects a summation of the absolute values.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: index	X: sum of the elements
L: l	L: index

APPLICATION PROGRAMS FOR RGSUM

Example 2: In matrix F above, one wishes to put into the 3rd column, the percentages corresponding to the values of the 2nd column in relation to their sum:

Press on	Display	Remarks
3[RCL]00[COLPT]	27.04705	Destination index

```
2[LASTx][COLPT]       26.04605       Index of origin
[RGCOPY]              27.04705       Copies the 2nd column in                    place of the
3rd.
[RGSUM]               60.00000       Sum of elements
[LASTx][X<>Y]         60.00000       Preserves the index
100 [/]               0.60000        Set for % calculation
[X<>Y][RG/Y]          27.04705       The calculations are                        completed
```

The matrix now appears as below:

MATRIX F

Col	1	2	3	4	5
Row 1	R25 -14	R26 15	R27 25	R28 2	R29 8
Row 2	R30 7	R31 13	R32 21.6	R33 -20	R34 1
Row 3	R35 0	R36 6	R37 10	R38 18	R39 24
Row 4	R40 23	R41 4	R42 6.6	R43 11	R44 17
Row 5	R45 16	R46 22	R47 36.6	R48 9	R49 10

The 3rd column in effect contains the percentages corresponding to the values of the elements of the 2nd column in relation to their sum.

RGVIEW Scrolling or listing of registers

[**RGVIEW**] (**R**e**G**isters **VIEW**) is intended for different modes of presentation and/or scrolling of registers.

Example: To carry out different presentations of the matrix below: in some situations, the contents of the registers may be altered.

MATRIX I

Col	1	2	3	4	5
Row 1	R25 1	R26 2	R27 3	R28 4	R29 5
Row 2	R30 6	R31 7	R32 8	R33 9	R34 10
Row 3	R35 11	R36 12	R37 13	R38 14	R39 15
Row 4	R40 16	R41 17	R42 18	R43 19	R44 20

|

Press on	Display	Remarks
[CF]28[FIX]6[<-]	0.000000	
[RCL]00[RGVIEW]	25 = 1.000000	
	30 = 6.000000	Call up the 1st column
[R/S]	35 = 11.00000	Interruption of the listing
[SST]	40 = 16.00000	Single step forward scrolling
[BST]	35 = 11.00000	Backward single step scrolling

```
[<-]                    25.044005       Stops the scrolling
[CLINC]                 25.044000       Range of registers index
[RGVIEW]                25 = 1.000000
                        26 = 2.000000    Automatic scrolling of
                        27 = 3.000000     consecutive registers
                        28 = 4.000000
[ON]                                     Puts off the calculator
[ON][CHS]               -25.044000       Index for stopping at the the first element
[RGVIEW]                25 = 1.000000
15                      25 = 15_        Value input
[CHS]                   25 = -15_
[EEX]                   25 = -15_       The scrolling is similar
2[CHS]                  25 = -15  .2_     to the usual method of display
[R/S]                   26 = 2.000000    Data confirmed
[BST]                   25 = -0.15000    Verification
[SST][ALPHA]            26 = 2.000000    Displays ALPHA mode
ABCDEF                  26 = ABCDEF_     ALPHA display accepts
G                       26 = BCDEFG_      six characters
[<-]                    26 = BCDEF_      Correction possible
[R/S][BST]              26 = BCDEF       Data validated and verified
[SST] A                 27 = A_         The ALPHA mode is retained
[ALPHA]                 27 = 3.000000    Reverts to numeric mode
[EEX] 2                 27 = 1  2_
[SST][BST]              27 = 3.000000    Data not validated by [R/S]
[<-]                     -25.0440000      Returns to calculation mode
2[EEX]6[CHS]            2       -6
[RCL]00[+]              25.044052       Matrix index
[ALPHA] RIEN A          RIEN A_         Gives a name to the matrix
[ALPHA][RGVIEW]        A1,1 = -0.150000 Only the last character of the ALPHA register serves as
                                          name for the matrix
                        A1,2 = BCDEF      The progression takes place automatically
                        A1,3 = 3.000000    across the matrix
[R/S]                   A1,4 = 4.000000   [R/S] immobilized
[SST][SST]              A2,1 = 6.000000   The coordinates of the [BST]
                        A1,5 = 5.000000     elements are displayed
                          on the left
19,5                    A1,5 = 19.5_      The matrix can be scrolled rapidly and clearly.
[R/S]   .               A2,1 = ._
[R/S]                   A2,2 = 7.000000
[BST]                   A2,1 = 0.000000
[<-]6[EEX]6[CHS]        6     -6_
[RCL] 00 [CHS]          -25.044056        Index of the linear matrix
[RGVIEW]                A1 =-0.1500000    1st element of the first column
[SST]                   A2 = 0.000000     2nd element (R30)
[<-]3[RCL]00[COLPT]     27.042050    3rd column index
6[EEX]6[CHS][+]         27.042056
[CHS][RGVIEW]           A1 = 3.000000     1st element [R27]
[ALPHA] LUNDI           A1 = LUNDI_
[R/S] MARDI             A2 = MARDI_       Scrolling of the column
[R/S] MERCR.            A3 = MERCR._       element by element
[R/S] JEUDI [R/S] ALPHA          -27.042056    End of scrolling and return to calculation mode.
4[EEX]6[CHS][ENTER]             0.000004      Building of a new matrix
3[RCL]00[COLPT][+]              27.042054      index
[CHS][RGVIEW]                   LUNDI =        In {RGVIEW} mode -29
LUNDI = 29_                                       accepts data while
[R/S] 12                        MARDI = 12_                retaining in the display
```

```
[R/S][BST]                 12.000000 =    the previous contents
[<-] 1 [EEX] 6 [CHS]       1        -6
[RCL]00[+][RGVIEW]         25 = -0.150000   In automatic listing
                           35 = 11.000000    mode, null values are
                           40 = 16.000000    skipped
                           25.044051
```

DETAILED INSTRUCTIONS FOR RGVIEW

1. [RGVIEW] is a general function of display, printing and of scrolling of data for registers in main memory.

2. The index (control value) in the X register determines the mode of access to the registers. The index is of the form bbb.eeeiij.

If X >= 0: continuous scrolling takes place until interruption by means of the [R/S] key or exhaustion of the registers specified by the index.

If X < 0, scrolling stops at the first register associated with the corresponding value of the index. The next register is obtained by means of the [SST] key. The [R/S] key restarts the the listing in the same way as X >= 0.

When j is an odd number, the registers containing zero values are not scrolled. If j = 0 or 1 the scrolling is normal; that is the display includes the register number before its contents.

If j = 2 or 3 [RGVIEW] displays the elements of a matrix preceded by the name of the matrix, the row number and the column number.

If j = 4 or 5 [RGVIEW] displays the contents of the register followed by the "=" sign. This sign is retained with the contents when the scrolling is interrupted.

Example: Display LUNDI =
Scrolling stopped LUNDI = 10_

If j = 6 or 7 [RGVIEW] displays the name of the linear matrix, the interval value and its contents.

In ALPHA mode only the last 6 characters scrolled are accepted.

A printer in NORMAL or TRACE mode prints the range of registers generated by [RGVIEW].

3. [RGVIEW] behaves like CATalogue (permits [BST] and [SST]).

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: index	X: index
L: l	L: used pointer

SORT Numeric and/or ALPHA sorting

[**SORT**] (**SORT**er) sorts the contents of the registers specified in X.

Example: In matrix A below:

MATRIX A

Col	1	2	3	4	5
Row 1	R25 14	R26 B	R27 21	R28 2	R29 8
Row 2	R30 7	R31 13	R32 19	R33 20	R34 1
Row 3	R35 0	R36 A	R37 -12	R38 18	R39 24
Row 4	R40 23	R41 99	R42 50	R43 11	R44 17

Press on	Display	Remarks
2[RCL]00[COLPT]	26.04105	Building of the index of the 2nd column
[XEQ] SORT	SORTING	Sorting in progress
	26.04105	Sorting terminated
3[LASTx](COLPT)[CHS]	-27.04205	3rd column index; the negative sign indicates descending sorting order.
[XEQ] SORT	SORTING	Sorting in progress
	-27.04205	Sorting terminates

MATRIX A

Col	1	2	3	4	5
Row 1	R25 14	R26 13	R27 50	R28 2	R29 8
Row 2	R30 7	R31 99	R32 21	R33 20	R34 1
Row 3	R35 0	R36 A	R37 19	R38 18	R39 24
Row 4	R40 23	R41 B	R42 -12	R43 11	R44 17

DETAILED INSTRUCTIONS FOR SORT

1. [SORT] sorts without discriminating between numeric values or alpha strings. Strings are sorted according to the ASCII code and are considered greater than numeric values.

2. The index placed in X designates the register for sorting.

3. If X >= 0, the contents are sorted in ascending order.

4. If X < 0, the contents are sorted in descending order.

5. During sorting, the message "SORTING" is displayed. If a message is already in the display, it is maintained and "SORTING" is not displayed.

The Stack:

The execution of the [SORT] function does not later the stack.

STO>L Storing of data according to the index in L

[**STO > L**] (**STO**re by **L**) arranges the value present in the X register at the address indicated by the integer portion of the index placed in the L register, and increments the index placed in L; the stack changes are not taken into account.

Example 1: To fill all the values of the first row of a matrix of 4 rows, 5 columns and starting with register 25;

Press on	Display	Remarks
1	1_	
[RCL] OO	25.04405	Recalling of the index
[LINPT]	25.02900	Calculates the index of the first row
[STO] [.] [L]	25.02900	Places the index in L
50	50_	1st element of the row
[XEQ] STO > L	50.00000	Arranges the value in R25
[VIEW] [.] [L]	26.02900	The index has been incremented
60	60_	2nd element of the row
[XEQ] STO > L	60.00000	Arranges the 2nd element
70	70_	
[XEQ] STO > L	70.00000	
80	80_	
[XEQ] STO > L	80.00000	
90	90_	
[XEQ] STO > L	90.00000	
[LASTx]	30.02090	

DETAILED INSTRUCTIONS FOR STO > L

[STO > L] uses the contents of the L register as index for arranging the values put stored successively in the X register. At the execution of [STO > L] the contents of X is transferred into the data register specified in L. The stack changes are not taken into account, so that several values can be arranged successively altering the contents of the Y, Z and T registers. Also, the index stored in L is incremented automatically. This conserves the memory space occupied by the program.

The Stack

Input	Output
T: t	T: t
Z: z	Z: z
Y: y	Y: y
X: Value to be arranged	X: Arranged value
L: bbb	L: bbb+1

Remark: The fractional part of the L register is ignored.

NOTE: If the L register contains an ALPHA string, the calculator signals the error message ALPHA DATA.

APPLICATION PROGRAM FOR STO > L

Example 2. The [STO > L] function is intended for storing of values into registers in the course of programming. Thus to store the first column of the matrix below, whose index is in the R00 register, the sequence is as follows:

1 RCL 00 COLPT STO > L 50 STO > L 60 STO > L 70 STO > L 80 STO > L

MATRIX I

Col	1	2	3	4	5
Row 1	R25 50	R26	R27	R28	R29
Row 2	R30 60	R31	R32	R33	R34
Row 3	R35 70	R36	R37	R38	R39
Row 4	R40 80	R41	R42	R43	R44

SUB$ Extraction or justification of a sub-string

[**SUB$**] (**SUB**string) extracts a sub-string from the content of the ALPHA register, or formats a string by addition of spaces to the right or left.

Example: To extract 7 characters beginning from the letter C of the string "ABCDEFGHIJKLMNOPQRSTUVW" present in the ALPHA register:

Press on	Display	Remarks
2.08	2.08	2 is the position of the last character before C, 8 is the position of the 7th character to be isolated
[XEQ] SUB$	2.0800	Extracts the sub-string
[ALPHA]	CDEFGHI	Sub-string

To justify to the right by a space of 10 characters:

[ALPHA]		
10	10_	Size of the space used
[CHS]	-10_	Negative sign indicates justification to the right
[XEQ] SUB$	-10.0000	
[ALPHA]	CDEFGHI	The string is preceded by 3 spaces for correct justification

To put 5 spaces to the right of this string:

[ALPHA]	-10.0000	
15	15_	Indicates the new space used.
[XEQ] SUB$	15.0000	Justification to the left since the content of the X register is positive
[ALPHA]	CDEFGHI	
[APPEND]	DEFGHI _	The ALPHA register shifts to thevleft and the cursor waits for a new character after 5 spaces

FULL INSTRUCTIONS FOR SUB$

[SUB$] modifies the contents of the ALPHA register in conformity with value in the X register.

- If X contains a whole number (say x), the calculator extracts |x| characters to the right of the string of the initial string. if the initial string comprises less than |x| character s the calculator adds more spaces to complete the string to |x| characters; the spaces are to the left if x is negative, to the right if positive.

- If X contains a number with fractional part (bb.ee), the calculator extracts the substring composed of bb to ee characters from the initial string (the character to the extreme left is numbered 0). If ee is greater than the ASCII value of the last character, the substring extracted is made up of the substring from bb completed by the number of spaces required to obtain ee-bb+1 characters. The spaces added are to the right if X is sign of X does not intervene if ee is less than or equal to the number of the last character of the initial string.

- If bb is greater than the number of the last character of the initial string, SUB$ puts into the ALPHA register, a string made up of ee-bb+1 spaces.

The Stack:

The stack is not altered by the execution of [SUB$].

NOTE: If the ALPHA register contains 24 characters the calculator puts characters with zero code at the beginning of the string which will appear as small arrows preceding the string.

TF55 Reversal of the printer existence flag

[**TF55**] (**T**oggle **F**lag **55**) reverses the status of flag 55, which normally indicates if a printer is connected to the HP41. This flag cannot be manipulated by the user without the PANAME module. The TF55 function operates as follows:

1. Sets flag 55 when a printer is not connected to the HP41; this facilitates the use of certain programs (available for example in application modules) that are executed mandatorily with flag 21 (printer enable) set, to be used as subroutines. Such programs are interrupted they encounter the VIEW and AVIEW if flag 55 is not set. TF55 because it sets flag 55 avoids these interruptions.

2. Cancels flag 55 when a printer is connected to the HP41. This accelerates the speed of execution of programs as long as the printer is not required; the printer is re-enabled by a new TF55.

DETAILED INSTRUCTIONS FOR TF55

1. To set flag 55 when it is not set, execute TF55.

2. To cancel flag 55 when it is set, execute TF55.

VKEYS Viewing of assigned keys

[**VKEYS**] (**V**iew **KEYS**) displays successively the assigned keys (redefinitions accessible in USER mode) from the R/S key, from top to bottom and from right to left. As an illustration, if the function "PROMPT" is assigned to the key "ENG" (yellow key followed by [3], code of the key -74), the calculator will display: -74 PROMPT.

The assignment listing can be

- temporarily interrupted by the continuous pressing of a key other than R/S or ON;
- definitively interrupted by pressing the R/S or ON keys which besides puts off the calculator.

NOTE: VKEYS is not programmable.

WRTEM Saving of files from extended memory on to mass storage

[**WRTEM**] (**WR**ite **E**xtended **M**emory) recopies the contents of extended memory on to a supported medium (HP82161A cassettes or HP9114 diskettes).

Example: To save the group of files "MAT 3" from extended memory on to cassette:

Press on	Display	Remarks
[XEQ] EMDIR	MATRP P012	
	A D100	These files have previously
	TEXT A040	been ready by READEM
	...	
[ALPHA] MAT 3 [ALPHA]	600.0000	ALPHA contains the generic name of the files to be read
[XEQ] WRTEM	600.0000	The files have been stored on cassette

DETAILED INSTRUCTIONS FOR WRTEM

1. After you have placed the generic name of the files to be read, executing WRTEM copies the specified file from extended memory on to the cassette.

2. If the HPIL module is not connected, the NO HPIL message is displayed.

3. If the file exists on the cassette, the file is replaced.

The Stack:

The stack is not altered by WRTEM.

REVERSE FUNCTION : READEM

X<>F Exchanging X with flags 0 to 7

The **X<>F** function exchanges the contents of the X register and of a fictitious register F which contains a representation of the status of flags 0 to 7. This representation is a whole number between 0 and 255, corresponding to the sum of the values of the flags set.

Flag	Value
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

For example, if flags 0, 1 and 3 are set, and flags 2, 4, 5, 6 and 7 cleared, the "F" register contains:

```
  1 (value of flag 0)
+ 2 (value of flag 1)
+ 8 (value of flag 3)
= 11
```

DETAILED INSTRUCTIONS FOR X<>F

To define a new status of flags 00 to 07, and simultaneously obtain the representation of their actual state:

1. Calculate (cf. above) the representation R' of the new status desired, and put this value in the X register;

2. Execute X<>F.

Consequently, the X register contains R, representation of the state of flags 00 to 07 before the execution of X<>F, and the new status of the flags corresponds to the R' representation.

Example of application of X<>F:

The following XFLAGS programme makes possible the use of (up to 80) flags of general usage. These "extended flags" (EF) numbered 00 to 79 are used as follows:

- to set the EF number N, place N in the X register and execute XSF.

- to cancel the EF number N, place N in the X register and execute XFS?. Consequently, flag 8 of the HP41 takes on the same status (set or cancelled) as the EF number N.

The XSF, XCF and XFS? programs use the stack, registers R00 to R09; XFS? use flag 8 as well.

Listing of the XFLAGS programs:

LBL "XFLAGS"
.009
RGINIT
RDN
RTN
LBL "XSF"
XEQ 00
SF IND Y
GTO 01
LBL "XCF"
XEQ 00
CF IND Y
GTO 01

LBL "XFS"
XEQ 00
CF 08
FS? IND Y
SF 08
LBL 01
X<>F
STO IND Z
R_
RTN
LBL 00
STO Y
8/
MOD
RCL IND Y
X<>F
.END.

Note: Executing "XFLAGS" cancels all X-Flagss 00 to 79.

X…NN? Comparison between X and a register

The X # NN?, X <= NN?, X < NN?, X = NN?, X >= NN? and X > NN? functions are similar to the functions of standard comparison (e.g. X = Y?) of the HP41, but they do not compare the contents of the X and Y registers, but the content of the X register with the content of the register specified in Y. These functions also compare ALPHANUMERIC strings.

DETAILED INSTRUCTIONS FOR X ... NN?

T effect a comparison between the contents of the X register and that of a register R, placed in Y, proceed as follows:

If the R register is:	Place in Y
- a data register Rnnn	- the number nnn
- the Z register	- the string "Z" ("Z" ASTO . Y)
- the T register	- the string "Y"
- the L	- the string "L"

then carry out the comparison. In calculation mode, the HP41 displays YES or NO according to the result of the comparison.

In the process of execution of a program, the program line following the test is executed if the result of the comparison is YES; it is ignore if the reverse is the case, like all other test operations of the HP41.

These functions compare numbers and alphanumeric strings in the following way:

1. A number is always strictly less that a string

2. Strings are arranged by the value of the codes of their characters (e.g. "AB0" < "ABA" because the code for "0" is 48 and that of "A" is 65).

3. A short string identical to the beginning of a longer string is considered as less (e.g. "ABC" < "ABCD").

Y/N YES or NO to a question

[Y/N] simplifies programs which, at the time of their execution, poses a YES or NO answer requirement to the user.

Example: The following program sequence makes the calculator to display the question: END Y/N? and directs the execution of the program to label 00 if the user replies YES to the question (by pressing the O key) or to label 01 if the user replies NO to the question (by pressing the N key):

```
"END"       10000    Rest of the program
 Y/N
 GTO 00
 GTO 01
```

DETAILED INSTRUCTIONS FOR Y/N

The Y/N function is used only in the middle of a program.

1a. To ask a question of the form:

```
     Message          Y/N?
```

place the "message" (7 characters maximum) into the ALPHA register and execute Y/N;

1b. To ask a question in another form (e.g. END   Y/N) place the message into the ALPHA register, execute AVIEW then Y/N.

2. In all cases, at the time of the Y/N execution, the calculator stops and waits for the pressing of a key:

- if the ON key is touched, the calculator is switched off;
- if the R/S key is pressed, the execution of the program is suspended and the program pointer is moved to the line immediately following Y/N;
- if the Y (Yes) or O (Oui) key is pressed, program execution proceeds to the line immediately following Y/N;
- if the N (No or Non) key is pressed, the line immediately following Y/N is ignored, and execution of the program proceeds to the second line following Y/N (like in the case of a test with a false result - see for example the use instruction for the X=Y? function in the HP41 user manual).
- Any other key is ignored.

ON Appendix

This appendix describes the supplementary "functions" available in the HP41 calculator when it is switched with the [ON] key. The "functions" are similar to the function of reinitialisation of the calculator, which is achieved by simultaneously pressing the [ON] and [<-] keys when the calculator is off.

Notation:

ON/+ symbolizes the the "function" obtained by holding on to the + key depressed at the same time as the ON key when the calculator is off, the + key being re;eased after the ON key.

ON/. Changes the "American" mode of display of numbers (e.g. 1.2345) to the "European" format (1,2345) and vice versa. This "function" exists on the Hewlett-Packard series 10 calculators (HP10C, HP11C, HP12C, HP15C and HP16C). Note that this "function" reverses the status of flag 28.

ON/K Cancels all key assignments obtained with the ASN function and activates USER mode.

ON/A carries out the assignments of the "A set" listed in the table below. If one of the keys used by these assignments already makes up an assignment, this is not altered.

ATOXL	ALENG	ATOXX	ANUMDEL	ATOXR
XTOAL	AROT	YTOAX	ANUM	XTOAR

ON/M is like ON/A, but activates the set of assignments M listed below:

STO>L	BRKPT	COLPT	AD-LC	RGVIEW
RG	BLDPT	LINPT	LC-AD	CLINC

ON/T like ON/A, but activates the set of assignments T listed below:

AXIS	BOX	SETORG	RMOVE	*CSIZE
*HOME	RESET	*LABEL	*MOVE	*LDIR
*PREGX	REVLFX	BACKSPX	RDRAW	*LTYPE
		OUT	*DRAW	COLOR

ON/V like ON/A, but activates the assignment set V listed below:

	SCRLUP	CLEAR	XYTAB	CTYPE
HOME	SCRLDN	CLEARO	CSRL	CSRR
	SCRLX	CSRVX	CSRUP	CSROFF
		CSRNX	CSRDN	CSRON

Scan Copyright ©
The Museum of HP Calculators
www.hpmuseum.org

Original content used with permission.

Thank you for supporting the Museum of HP Calculators by purchasing this Scan!

Please to not make copies of this scan or make it available on file sharing services.