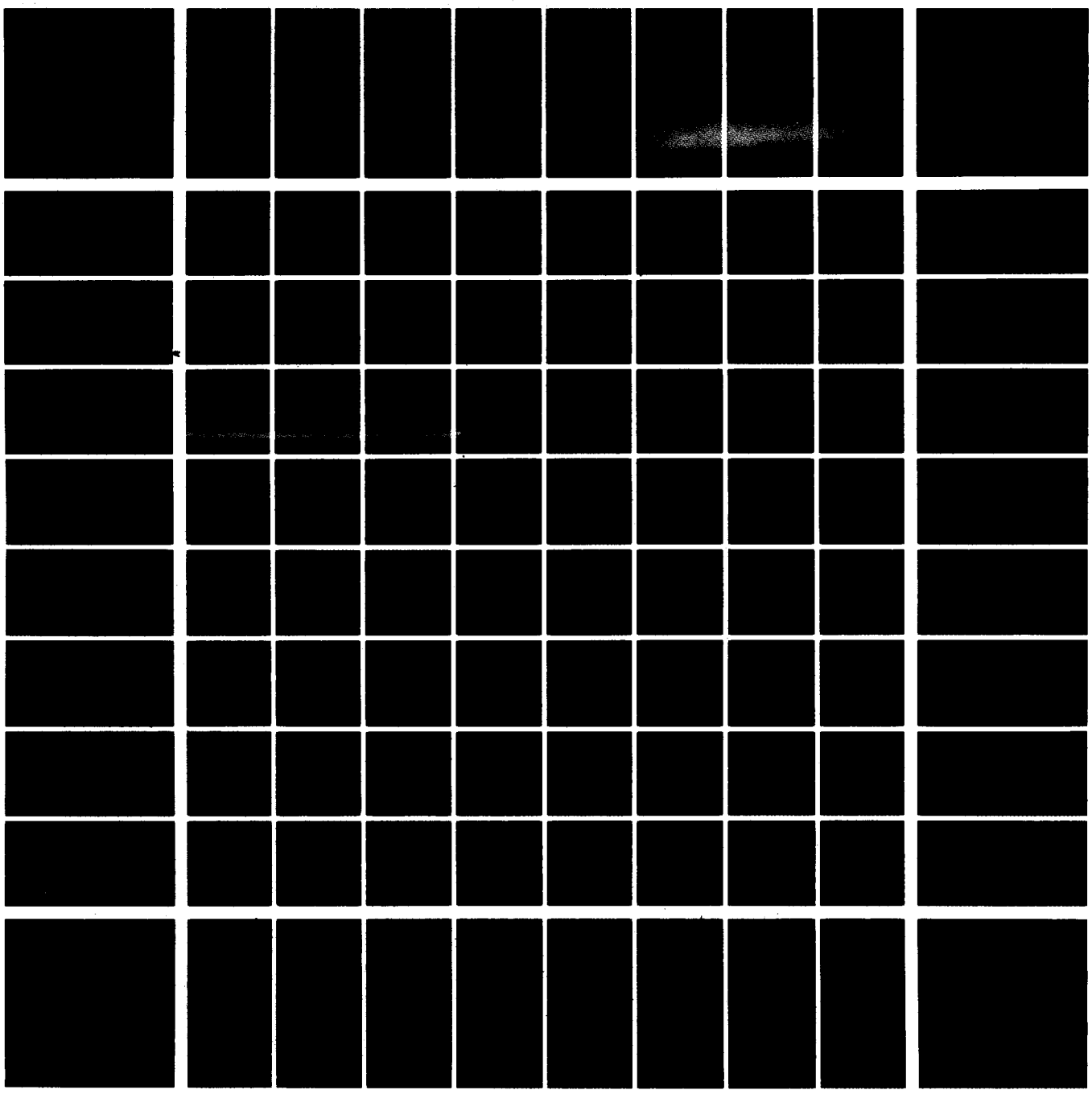
HEWLETT-PACKARD

HP 82180A

Extended Functions/Memory Module

OWNER'S MANUAL



Notice

Hewlett-Packard Company makes no express or implied warranty with regard to the keystroke procedures and program material offered or their merchantability or their fitness for any particular purpose. The keystroke procedures and program material are made available solely on an "as is" basis, and the entire risk as to their quality and performance is with the user. Should the keystroke procedures or program material prove defective, the user (and not Hewlett-Packard Company nor any other party) shall bear the entire cost of all necessary correction and all incidental or consequential damages. Hewlett-Packard Company shall not be liable for any incidental or consequential damages in connection with or arising out of the furnishing, use, or performance of the keystroke procedures or program material.

HEWLETT PACKARD

HP 82180A
Extended Functions/Memory Module

Owner's Manual

September 1985

82180-90001 Rev. D

Printed in Singapore

Contents

Introduction

The HP 82180A Extended Functions/Memory Module adds a number of useful functions to those already available on your HP-41 calculator and also provides you with extended memory. Extended memory can be augmented by adding one or two HP 82181A Extended Memory Modules, the use of which is also described in this manual.

Extended Functions

The functions provided by the module can be grouped in the four categories described under the following headings:

Programmable Functions

Some standard calculator functions, such as [ASN] and [SIZE] that are not programmable, have programmable equivalents in the module. Additional programmable functions that have no equivalent in the basic calculator have been provided to make it easier to write efficient programs.

Register and Flag Functions

Some of these let you manipulate blocks of registers easily. Others extend the utility of the calculator flags.

ALPHA Register Functions

These let you extract numeric data from the ALPHA register, search the ALPHA register for specific strings, and convert characters to numeric equivalents and vice versa.

Extended Memory Functions

These enable you to store and retrieve programs and data in the extended memory registers provided by the module and by extended memory modules. They also let you create and edit text composed of alphanumeric characters.

Extended Memory

By itself, the extended functions/memory module contains 127 extended memory registers.* In most respects, these are like the calculator registers with which you are already familiar. The important difference is that data stored in these registers is not immediately available to the calculator. Before such data can be used it must first be moved into calculator main memory. This is discussed in sections 1 and 3 of this manual.

One or two HP 82181A Extended Memory Modules may be used in conjunction with the HP 82180A Extended Functions/Memory Module. Each will add 238 registers. Thus it is possible to add 603 registers of extended memory to your HP-41C or HP-41CV.

---

* The memory registers provided by the HP 82180A Extended Functions/Memory Module and the HP 82181A Extended Memory Module are distinct from, and should not be confused with, registers $R_{(100)}$ through $R_{(318)}$ in main memory—which are called *extended storage registers* in the calculator owner's handbook.

The HP 82180A Extended Functions/Memory Module and the HP 82181A Extended Memory Module can be used with either the HP-41C or the HP-41CV calculator. The instructions in this manual apply to both calculators.

Always turn your calculator OFF before inserting or removing any modules. If you don't, the calculator may be damaged or the system's operation may be disrupted.

The HP 82180A Extended Functions/Memory Module can be identified by the legend X FUNCTIONS permanently marked on the module. The HP 82181A Extended Memory Module is marked X MEMORY.

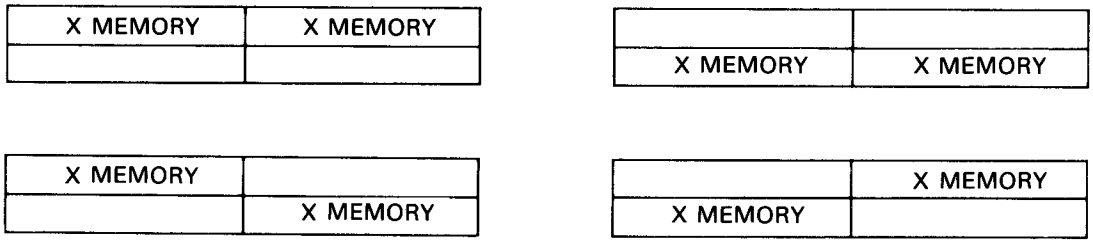
To insert an extended functions/memory module or an extended memory module, orient it so that the legend is right side up, hold the calculator with the keyboard facing up, and insert the module into a port. You'll feel it snap into place when it's properly seated.

To remove a module, use your fingernail to gently extend the extractor handle. Then grasp the handle and pull out.

Configurations

The extended functions/memory module can be installed in any calculator port. If you have only a single extended memory module in addition to the extended functions/memory module, it can be installed in any other port.

If you later add a second extended memory module (or if you install two extended memory modules at the same time), the extended memory modules must be arranged in one of the following configurations. Don't install them one above the other.



Port Configurations When Two HP 82181A Extended Memory Modules Are Used

If you remove one or more of the modules, some or all of the data in extended memory may be lost. To minimize this data loss there is an optimum sequence for removing modules. However, the sequence is conditional on the order in which the original configuration was accomplished, as follows:

- If extended memory modules were installed at different times, remove the modules in the order opposite to that in which they were installed.
- If extended memory modules were installed at the same time, remove the module in port 2 or 4 first, then, if necessary, the module in port 1 or 3.

The reasons for these removal procedures are explained in section 3.

Note: If you have the HP 82104A Card Reader plugged into the calculator and an HP 82181A Extended Memory Module plugged into port 2, and you execute the card reader function [VER], some information in the extended memory module may be changed. Therefore, you should avoid using the [VER] function if you are also using an extended memory module in port 2.

Using This Manual

When the extended functions/memory module is inserted in your calculator, its functions become available for your use. These functions are grouped and discussed in two general categories: extended functions (section 2) and extended memory (section 3). The extended functions section deals with programmable functions, functions to manipulate registers and flags, and functions for manipulating data in the ALPHA register. The extended memory section deals with transferring data and programs between the calculator's main memory and extended memory, and with editing commands that are used with ASCII files that can be created in extended memory.

For simplicity, functions provided by the extended functions/memory module are represented by single, colored keys—such as [ANUM]. When you want to execute a function, you can do it in two ways: by using [XEQ] [ALPHA] *name* [ALPHA], or by assigning the function to a key using [ASN] (or [PASN]), and pressing that key in User mode.

In this manual, the description of each function is preceded by a summary of information required by that function. This provides a quick, visual summary of how to execute the function. For example:

PASN X [*keycode*] ALPHA [*function name*]

This indicates that a keycode must be placed in the X-register and a function name placed in the ALPHA register before you execute PASN from the keyboard or in a program.

If at any time an error message is displayed by the calculator, refer to appendix B for an explanation of its cause.

Section 2

Extended Functions

Data Storage Register Operations

REGMOVE X [*sss.dddnnn*]

Executing REGMOVE (*register move*) copies a block of *nnn* registers, beginning at register *sss* (source), to a block of the same length, beginning at register *ddd* (destination). Any data that was already in the destination block is lost.

REGSWAP X [*sss.dddnnn*]

Executing REGSWAP (*register swap*) exchanges the contents of a block of *nnn* registers beginning at register *sss* with the contents of a block of the same length beginning at register *ddd*.

If *nnn* is zero for either REGMOVE or REGSWAP, one register will be copied or exchanged.

Flag Operations

It is often helpful to be able to restore the calculator flags to a preexisting configuration—for instance, to restore the display format after executing a program. The following two functions enable you to recall the condition of flags 0 through 43 and later to use this data to restore some or all of these flags to their previous condition.

RCLFLAG

Executing RCLFLAG (*recall flags*) recalls the status of flags 0 through 43 to the X-register as ALPHA data. You can then store the contents of the X-register for later use.

Note: When RCLFLAG is executed, the display will not be intelligible.

STOFLAG X [*flag status*]

STOFLAG X [*bb.ee*] Y [*flag status*]

If the flag status from a previously executed RCLFLAG function is placed in the X-register, executing STOFLAG (*restore flags*) restores calculator flags 0 through 43.

If you want to restore only some of the flags, place the flag status in the Y-register and a number in the form *bb.ee* (representing the beginning and ending flags of the block to be restored) in the X-register, and execute [STOFLAG].

Example. Suppose you want to write a program that gives answers in [FIX] 0 format, without a decimal point, but when program execution is finished you want the display format restored to whatever it was before you ran the program. The program lines below show how you could do this.

```
01 LBL"ABC
02 RCLFLAG
03 STO 20
04 FIX 0
05 CF 29
06    .
07    .
08    .
 .    .
 .    .
 .    .
20 RCL 20
21 STOFLAG
22 END
```

These two steps recall the status of flags 0 to 43 to the X-register and save that status in register 20. This block of flags includes flag 29, the digit grouping flag, and flags 36 through 41, the number of digits and display format flags.

These two steps set the display format for your program.

Your program.

These steps recall the original flag status to the X-register and restore the flags and display status.

[X<>F] **X** ***flag 0 through 7 status***

[X<>F] (*X exchange Flags*) uses the number in the X-register to set flags 0 through 7.

The number in the X-register is interpreted as a status code. Each flag (0 through 7) is represented by a number shown in the table below; the status code is the sum of those numbers for the flags that are set.

Executing [X<>F] sets the status of flags 0 through 7 according to the status code in the X-register, and places in the X-register a code representing the former status of those flags.

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

Example. Suppose you want to set flags 0, 3, 5, and 7, leaving flags 1, 2, 4, and 6 clear. Instead of pressing ■ [SF] 0 ■ [SF] 3 ... ■ [SF] 7, you could simply add up the numeric codes for the set flags according to the table above, then—with the sum in the X-register—press [X<>F].

Flag	Numeric Equivalent	
0	1	
3	8	
5	32	
7	128	
	169	is the number in the X-register.

If you enter 0 in the X-register and execute [X<>F], flags 0 through 7 are cleared and their previous status is placed in the X-register.

[X<>F] enables you to increase the number of *apparent* general purpose flags. To do so, calculate several flag status codes and store them in separate registers for later use. You can use the status of an individual flag as defined by a particular status code by simply recalling the status code from its storage register and executing [X<>F].

User Mode Operations

[PASN] **X** [*keycode*] **ALPHA** [*function or program name*]

Like [ASN], [PASN] (*programmable assign*) enables you to assign functions or programs to a key location. However, [PASN] can be executed from within a program. [PASN] requires you to enter the keycode for the key to which you wish to assign the function or program. This is the same keycode that the calculator itself displays when you use [ASN] to assign a function or program to a key. Keycodes are described more fully in the *HP-41C/41CV Owner's Handbook and Programming Guide*. Remember that keycodes for shifted keys are negative numbers.

As is the case with [ASN], you cannot use [PASN] to assign programs to any of the top four keys (keycodes 01 through 04) or to the shift key (keycode 31).

[PASN] cancels an assignment for the designated key if it is executed with the ALPHA register cleared.

[CLKEYS]

Executing [CLKEYS] (*clear keys*) clears all key assignments except those stored with programs in extended memory.

ALPHA String Operations

The extended functions/memory module enables you to shift data between the ALPHA and X-registers. In the ALPHA register, the data exists as alphabetic or numeric characters, while in the X-register, an alphabetic or numeric character is represented by a numeric character code.

Alphanumeric characters are represented within the calculator by character codes based on ASCII (American Standard Code for Information Interchange). In addition to the numerals and letters of the alphabet that correspond directly to ASCII, there are some nonstandard symbols represented by unique HP-41 codes. The following table lists symbols that can be displayed in ALPHA mode together with their character codes.

Displayable Characters and Their Equivalent Codes

Char.	Code	Char.	Code	Char.	Code	Char.	Code
←	0	,	44	@	64	T	84
↑	1	−	45	A	65	U	85
↓	4	.	46	B	66	V	86
→	5	/	47	C	67	W	87
▯	6	0	48	D	68	X	88
μ	12	1	49	E	69	Y	89
◿	13	2	50	F	70	Z	90
≠	29	3	51	G	71	[	91
space	32	4	52	H	72	\	92
!	33	5	53	I	73	]	93
"	34	6	54	J	74	↗	94
#	35	7	55	K	75	_	95
$	36	8	56	L	76	▔	96
%	37	9	57	M	77	a	97
&	38	:	58	N	78	b	98
'	39	;	59	O	79	c	99
(	40	<	60	P	80	d	100
)	41	=	61	Q	81	e	101
*	42	>	62	R	82	Σ	126
+	43	?	63	S	83	▌	127

Note: Some printer character sets match different characters with some of the above codes. Refer to your printer owner's manual.

Shifting Characters Between the ALPHA and X-Registers

ATOX

Executing ATOX (*ALPHA to X*) shifts the leftmost character out of the ALPHA register and places its character code in the X-register. If the ALPHA register is empty, the number zero is placed in the X-register.

XTOA X ***character code***

XTOA X ***ALPHA string***

Executing XTOA (*X to ALPHA*) with a character code from the above table in the X-register appends the character represented by the character code to the right-hand end of the string in the ALPHA register. (XTOA ignores the sign and fractional portion of the number in the X-register.) XTOA may be executed with any number from 0 to 255 in the X-register, but numbers that are not listed above as a character code are not valid codes, so the character appended in the ALPHA register will not be intelligible. (All segments of the display at that character position will be turned on.) If you execute XTOA with the number zero in the X-register, subsequent operations on the ALPHA register may not work properly until the register is cleared. (Refer to Appendix C for information on alpha operations with the null—code 0—character.) Executing XTOA with an alpha data string in the X-register appends the entire string to the ALPHA register.

Length of a String of Characters

ALENG

The ALENG (*ALPHA length*) function returns the number of characters in the ALPHA register to the X-register.

Searching ALPHA Strings

ANUM

The ANUM (ALPHA *number*) function scans the ALPHA register (from left to right) for an ALPHA-formatted number. If a number is found, its value is placed in the X-register and user flag 22 is set; if no number is found, the X-register and flag 22 are unchanged. If there is a space or another character (other than ., „ **E**, +, or −) between digits in the ALPHA register, the number placed in the X-register ends at that character. (Repetitions of . or **E** are ignored, and "," appears only where needed, regardless of whether it is used in the ALPHA register.)

> **Note:** ANUM ignores any + characters between digits in the Alpha register. However, any - character is interpreted as a CHS command. Thus, executing Anum with the string **7−5** produces **−75** in the X-register, and with **−7−5**, produces **75**.

The number placed in the X-register can consist of no more than 10 digits; additional digits in the ALPHA register will therefore be ignored. If the number in the ALPHA register is immediately preceded by a minus sign, it will be recalled as a negative number.

ALPHA numbers are formatted in the X-register according to the status of flags 28 (decimal point flag) and 29 (digit grouping flag) when ANUM is executed. Commas and periods in numbers may be interpreted differently depending on the status of these flags. For example, if the ALPHA register contains the string PRICE: $1,234.5, ANUM returns the following results depending on the status of flags 28 and 29:

Flag 28	Flag 29	Number Returned
set	set	1,234.5000
set	clear	1.0000
clear	set	1,2345
clear	clear	1,2340

Notice that when flag 29 is clear, a separator mark will be considered a character.

POSA X ***character code***

POSA X ***ALPHA string***

The POSA (*position in ALPHA*) function scans the ALPHA register for the ALPHA character or string specified in the X-register. There are two ways to specify the character or string. You can enter the character code for a single character, or you can enter an actual character or string of characters by using ASTO. If the specified character or string is found in the ALPHA register, the character position of the character (or the character position of the leftmost character in the string) is returned to the X-register. POSA replaces the value in the X-register with the result of executing POSA. (The stack does not lift.) The value in the X-register before you executed POSA is placed in the LAST X register. (Refer to appendix C for information on alpha operations with null characters.)

Character positions are counted from left to right, starting from position 0. If the specified string occurs more than once in the ALPHA register, only the position of the first occurence is returned. If the target string is not found in the ALPHA register, the number −1 is returned.

Rotating the ALPHA Register to Continue a Search

[AROT] X [*number of characters*]

Executing [AROT] (*ALPHA rotate*) rotates the contents of the ALPHA register by the number of characters given in the X-register. The ALPHA register is rotated to the left if the number in the X-register is positive, or to the right if the number is negative. (Refer to Appendix C for the effects of [AROT] on null characters.)

[AROT] can be used with [ANUM] and [POSA] to extract a sequence of numbers from the ALPHA register.

Example. As the result of an operation by some peripheral device, the ALPHA register contains the sequence **68.2 69.88** (two numbers, separated by a space). You want to extract each of these numbers in turn and use them in a program.

The following sequence illustrates the process:

Keystrokes	Display	
[XEQ] [ALPHA] ANUM	XEQ ANUM_	
[ALPHA]	68.2000	Places the first number in the X-register.
[STO] 20	68.2000	Stores the value for later use.
32	32_	32 is the character code for a space.
[XEQ] [ALPHA] XTOA	XEQ XTOA_	
[ALPHA]	32.0000	Appends a space after 69.88 in the ALPHA register.
[XEQ] [ALPHA] POSA	XEQ POSA_	
[ALPHA]	4.0000	Scans the ALPHA register for the first occurrence of a space (character code 32 was in the X-register).
[XEQ] [ALPHA] AROT	XEQ AROT_	
[ALPHA]	4.0000	Rotates the ALPHA register to the left by 4 characters. Now it reads: 69.88 68.2. Note that if you had not used [XTOA] above, the string would read: 69.8868.2.
[XEQ] [ALPHA] ANUM	XEQ ANUM_	
[ALPHA]	69.8800	Places 69.88 in the X-register.

Miscellaneous Operations

Determining What Key has been Pressed

[GETKEY]

When a program executes [GETKEY] (*get key*), execution halts until a key is pressed or an interval of approximately 10 seconds elapses. If a key is pressed, its keycode is placed in the X-register. If no key is pressed, the number 0 is placed in the X-register at the end of the time interval.

[GETKEY] responds to the first key pressed, so there can be no shifted responses to [GETKEY]. If you press the gold shift key during a [GETKEY] wait, its keycode (31) is placed in the X-register.

[GETKEY] enables you to branch to a subroutine on the basis of an entry from the keyboard, even when the key pressed is not a digit key.

SIZE-Related Functions

[PSIZE] **X** ***number of data storage registers***

[PSIZE] works like the [SIZE] function provided in the calculator except that it can be executed from within a program. It makes it possible for a running program to reallocate the registers in main memory as required.

[SIZE?]

Executing [SIZE?] places the number of registers currently allocated to data storage into the X-register.

[SIZE?] can be used within a program to inhibit the execution of [PSIZE] when a memory reallocation is not required.

The following program lines illustrate how [SIZE?] and [PSIZE] might be used in a program.

01 .	Your program.
02 .	
.	
.	
07 SIZE?	The number of data storage registers presently allocated is placed in the X-register.
08 ***nn***	Key in the number of registers this program needs. The result of the previous step is now in the Y-register.
09 X>Y?	Is the number of storage registers required by the program greater than the number presently allocated?
10 PSIZE	Reallocate memory only if the answer to the above question is yes.

Clearing Programs

[PCLPS] **ALPHA** ***program name***

Executing [PCLPS] (*programmable clear programs*) clears one or more of the programs in main memory. All programs beginning with the one named in the ALPHA register (or the current program if the ALPHA register is clear) and continuing to the end of program memory are cleared. If a running program names itself (or clears the ALPHA register) and executes [PCLPS], that program and all following it will be cleared and program execution will terminate.

Section 3

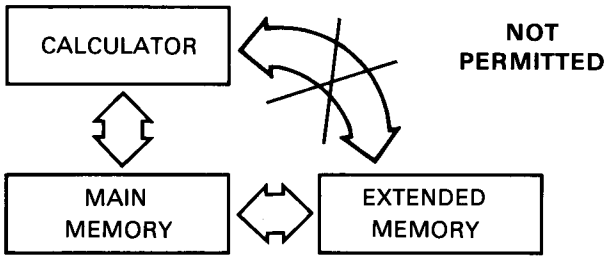
Extended Memory

Understanding Extended Memory

To use the extended functions/memory and extended memory modules effectively, you must understand the distinction between your calculator's main memory and the extended memory provided by these modules.

Your calculator by itself has a certain amount of main memory. If you have an HP-41C, you can add memory modules to increase the size of main memory up to that of the HP-41CV. Regardless of its size, main memory contains programs and data that are always instantly available to the calculator. You have only to press [XEQ] and enter a program name or [RCL] and enter a register number in order to execute a program or recall data.

Extended memory is somewhat different. In order for the calculator to use the programs and data in extended memory, they must first be transferred to main memory; they are not directly accessible. Extended memory gives you more storage space for programs and data, but the tradeoff you make for that extra capacity is the necessity of taking extra steps to transfer those programs and data between main and extended memory.



The registers in extended memory* are organized in structures called "files." A program you create in calculator main memory can be transferred to extended memory as one type of file.

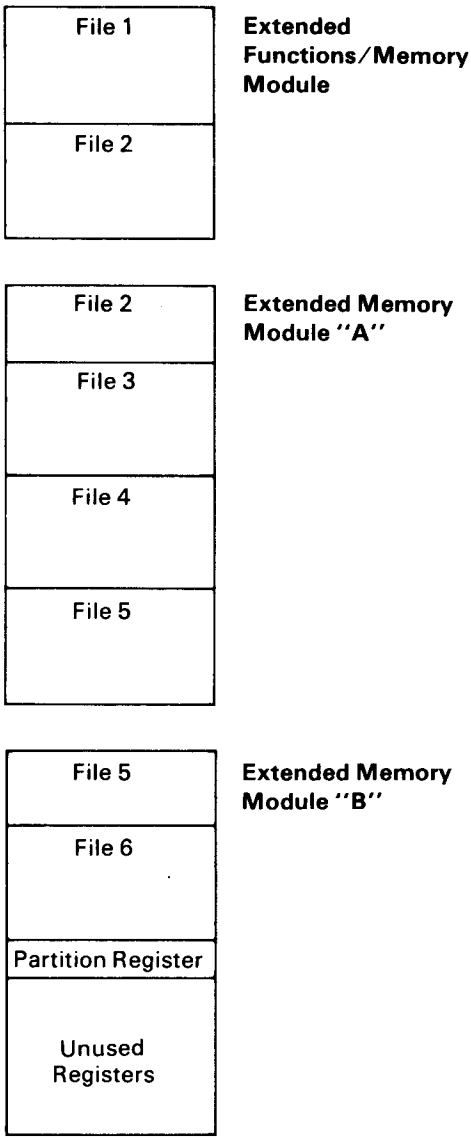
There are three kinds of files that can be stored in, and recalled from, extended memory:

- Data Files
- ASCII Files
- Program Files

* Refer to footnote, page 5.

Files consist of two registers (called the *header*) that contain information about the file, and one or more registers that contain data. Following the last file there is also one register used as a partition between used and unused extended memory. When extended memory is empty, three registers are reserved for the first file to be loaded (that is, two header registers and the partition register).

Files are stored in extended memory in the order in which they are created. Sometimes a file may be partly in one module and partly in another. In the illustration on the right, file 5 starts in extended memory module A, but some registers are in extended memory module B. This is why it is important to use the proper sequence in removing extended memory modules. If you have to remove a module, you want to lose as few files as possible. In the situation illustrated, if you remove extended memory module B, you lose only files 5 and 6. However, if you remove extended memory module A, the only file you will leave in extended memory will be file 1!



The guidelines in section 1 for removing extended memory modules are based on the following: when all the extended memory registers in the extended functions/memory module are filled and more data is to be stored, the extended functions/memory module senses whether one or two extended memory modules are installed. If only one is installed, the additional data is stored in its registers, regardless of its port location. If another extended memory module is installed later, it will be used when the first module overflows.

However, if there are two extended memory modules installed when the extended functions/memory module needs more registers, the data will first be stored in the extended memory module in calculator port 1 or 3. The extended memory module in port 2 or 4 will only be used when the first module is full.

If you perform a master clear (as described in the HP-41 Owner's Manual), all information in extended memory, as well as that in main memory will be lost.

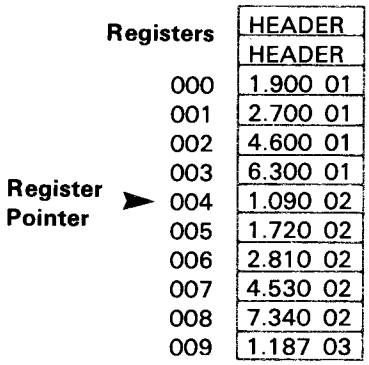
File Names and File Pointers

As previously mentioned, the first two registers in a file are called the header. They store certain information about the file that the calculator makes use of. The first register contains the file name. This can be any combination of alphabetic and numeric characters, including spaces, up to seven characters long. If you try to give a file a longer name, excess characters are truncated. If you create a file with a name of fewer than seven letters, the calculator adds spaces to bring the character count up to seven.

The second register in the header contains information on the length and type of the file and one or two "pointers" that are used to gain access to specific items in the file.

Data Files

Data files enable you to retain important data while using all the registers in main memory. Most of the functions that are used with data files make use of the file's "register pointer," an integer that is used to refer to a specific register within the file. In the example data file on the right, the pointer is positioned to the fifth register.



A Data File

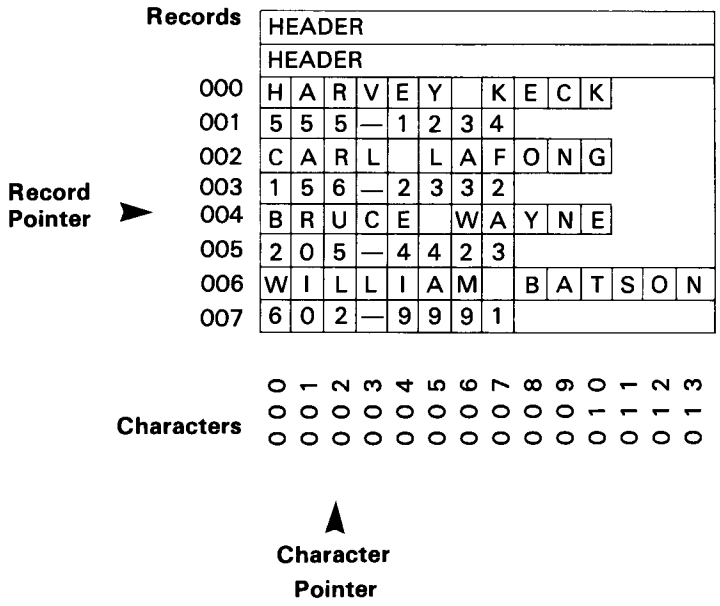
Many of the function descriptions in this section include examples. To duplicate their results, you can create and save a data file like the one on this page by pressing the keys shown below. The [CRFLD] and [SAVEX] functions used will be explained later in this section (pages 24 and 28).

Keystrokes	Display	
[ALPHA] SAMPL D	SAMPL-D	
[ALPHA] 20	20	
[XEQ] [ALPHA] CRFLD	XEQ CRFLD	
[ALPHA]	20.0000	
■ [ASN] [ALPHA]	ASN	
SAVEX [ALPHA] [Σ+]	ASN SAVEX 11	Assigns [SAVEX] to the [Σ+] key.
	20.0000	X-register displayed.
[USER]	20.0000	Switch to user mode.
19 [Σ+]	19.0000	
27 [Σ+]	27.0000	
46 [Σ+]	46.0000	
63 [Σ+]	63.0000	
109 [Σ+]	109.0000	
172 [Σ+]	172.0000	
281 [Σ+]	281.0000	
453 [Σ+]	453.0000	
734 [Σ+]	734.0000	
1187 [Σ+]	1,187.0000	
■ [ASN] [ALPHA]	ASN	
[ALPHA] [Σ+]	1,187.0000	Clears [SAVEX] assignment from the [Σ+] key.

The above data is now stored in the first 10 registers of SAMPL-D; the remaining registers are set to zero.

ASCII Files

ASCII files enable you to create texts of alphanumeric characters and search and edit your texts. To avoid the limitations that would be imposed by dealing with these files in terms of registers, ASCII files are organized into "records" and "characters," as shown in the following illustration. Each record in an ASCII file may contain from 1 to 254 characters. As the illustration shows, an ASCII file has two pointers—a record pointer and character pointer.



In the illustration, the record pointer is set to the fifth record and the character pointer is set to the third character. Taken together, the combination points to the **U** in **BRUCE WAYNE**. You can duplicate the ASCII file in the illustration by pressing the keys shown below. The [CRFLAS] and [APPREC] functions used will be explained later in this section (pages 24 and 29).

Keystrokes	Display
[ALPHA] SMPL - AS	SMPL - AS_
[ALPHA] 20	20_
[XEQ] [ALPHA] CRFLAS	XEQ CRFLAS_
[ALPHA]	20.0000
■ [ASN] [ALPHA]	ASN _
APPREC [ALPHA] [Σ+]	ASN APPREC 11
	20.0000
[ALPHA] HARVEY KECK	HARVEY KECK_
[ALPHA] [Σ+]	20.0000
[ALPHA] 555-1234	555-1234_
[ALPHA] [Σ+]	20.0000
[ALPHA] CARL LAFONG	CARL LAFONG_
[ALPHA] [Σ+]	20.0000
[ALPHA] 156-2332	156-2332_
[ALPHA] [Σ+]	20.0000
[ALPHA] BRUCE WAYNE	BRUCE WAYNE_
[ALPHA] [Σ+]	20.0000
[ALPHA] 205-4423	205-4423_
[ALPHA] [Σ+]	20.0000
[ALPHA] WILLIAM BATSON	WILLIAM BATSON_
[ALPHA] [Σ+]	20.0000
[ALPHA] 602-9991	602-9991_
[ALPHA] [Σ+]	20.0000
■ [ASN] [ALPHA]	ASN _
[ALPHA] [Σ+]	20.0000

Program Files

A program file is a program that is stored in extended memory. You can give yourself more room in main memory if you keep most of your programs in extended memory until you need one of them.

The following keystroke sequence creates a program that reads and displays the contents of ASCII file SMPL-AS, then transfers the program to extended memory as a program file. The [SEEKPTA], [GETREC], and [SAVEP] functions will be explained later in this section (pages 25, 26, and 32).

Keystrokes	Display
[PRGM] [GTO] [·] [·]	00 REG *nn*
[LBL] [ALPHA]	
SAMPL - P [ALPHA]	01 LBL"SAMPL - P
[ALPHA] SMPL - AS [ALPHA]	02"SMPL - AS
0	03 0_
[XEQ] [ALPHA] SEEKPTA [ALPHA]	04 SEEKPTA
[LBL] 01	05 LBL 01
[XEQ] [ALPHA] GETREC [ALPHA]	06 GETREC
[ALPHA] [AVIEW] [ALPHA]	07 AVIEW
[GTO] 01	08 GTO 01
[PRGM]	20.0000
[GTO] [·] [·]	20.0000
[ALPHA] SAMPL - P [ALPHA]	20.0000
[XEQ] [ALPHA] SAVEP	XEQ SAVEP_
[ALPHA]	20.0000

Working Files

Some extended memory functions require you to enter a file name in the ALPHA register before execution. After one of these functions has been executed, the calculator is set to the named file.* That is, the named file has become your *working* file in the same way that executing [GTO] with a program name makes the named main memory program your "working" program. Certain functions described on the following pages operate only on working files.

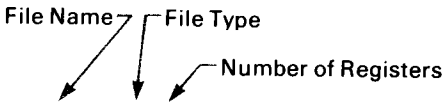
File Management

[EMDIR]

Executing [EMDIR] (*extended memory directory*) displays a list of the files in extended memory. The list can also be printed out. For each file, the file name appears on the left and the file type (indicated by D, A, or P) and the number of registers occupied appear on the right. After all the files have been listed, the number of extended memory registers still available for storing files is returned to the X-register.

Example. If you created the files described earlier in this section, executing [EMDIR] will produce the following listing:

*Except [PURFL] (*purge file*), which purges the named file from memory.



SAMPL-D D020
SMPL-AS A020
SAMPL-P P005
549.0000 Number of registers available (assuming two extended memory modules installed).

If there are no files in extended memory, the message DIR EMPTY is displayed and the number of registers available for storing files is returned to the X-register. For example, if the extended function/memory module is plugged in without any extended memory modules, executing [EMDIR] with extended memory empty places **124.0000** in the X-register. (While 127 registers are actually available, three are reserved for the first file to be loaded–refer to page 20.)

While the directory listing is being displayed (or printed), you can halt the listing by pressing any key except [R/S] or [ON]; listing continues when you release the key. You can terminate the listing by pressing [R/S] or [ON]. If you terminate the listing while a file name is displayed, that file becomes the working file if it is a data or ASCII file.

[CRFLD] **X** *number of registers* **ALPHA** *data file name*
[CRFLAS] **X** *number of registers* **ALPHA** *ASCII file name*

The [CRFLD] (*create file-data*) and [CRFLAS] (*create file-ASCII*) functions create, respectively, data files and ASCII files. When you execute [CRFLD], you will need to specify the same number of registers in the X-register as you have data items to store. It isn't necessary to add two registers for the header; the calculator takes care of that automatically.

When you create an ASCII file, if you know exactly how many characters and records there will be, you can calculate the number of registers that are required using the following steps: (1) Add the total number of characters in the file to the number of records, (2) Add 1 to the result, and (3) Divide the result by 7 and round up to a whole number.

Usually, you won't know exactly how many records or characters will be in an ASCII file. If you can make an estimate of the number of characters, a good rule of thumb is to add 20 percent to your estimate and divide the result by 7 to obtain an approximation of the number of registers needed.

When you create a file using [CRFLD] or [CRFLAS], that file becomes your working file.

[PURFL] **ALPHA** *file name*

Executing [PURFL] (*purge file*) removes the named file from extended memory.

[CLFL] **ALPHA** *file name*

Executing [CLFL] (*clear file*) retains the named file, but clears all the data in it. (The named file may not be a program file.) In data files, [CLFL] enters zeroes in all registers; in ASCII files, it sets the number of

records to zero. The named file becomes the working file.

FLSIZE	ALPHA *file name*

Executing FLSIZE (*file size*) returns the number of registers in the named file to the X-register. The named file becomes the working file. (If the ALPHA register is empty when FLSIZE is executed, the size of the working file will be returned.)

Program File Operations

SAVEP	ALPHA *program name, file name*
SAVEP	ALPHA *program name*
SAVEP	ALPHA *, file name*

Executing SAVEP (*save program*) copies the named program from main memory into extended memory under the specified file name. If only the program name is entered in the ALPHA register, the program is saved under that name. If only a comma and a file name are given, the current program in main memory is saved under the file name.

If a program file already exists with the specified file name, executing SAVEP purges the old file and creates a new program file with the specified file name.

To insure that a program will occupy the fewest possible extended memory registers, press ■ GTO · · before executing SAVEP.

GETP	ALPHA *program file name*

Executing GETP (*get program*) replaces the last program in main memory with the program stored in the named file.* If GETP is executed from the keyboard, the calculator is set to the first line of the new program. If GETP is executed from a running program, the results depend on whether or not the running program is the last program in main memory. If the running program is *not* the last program, it continues to run. If it *is* the last program, it is replaced by the program in the named file and execution continues from the first statement in that program.

GETSUB	ALPHA *program file name*

Executing GETSUB (*get subroutine*) copies the program stored in the named file to main memory following the last existing program. The calculator is *not* set to the transferred program.

Any key assignments recorded with the program named will become active if GETP or GETSUB is executed in User mode.

GETSUB inserts an **END** in program memory before copying a subroutine. This ensures that the last program in memory is not overwritten. If the last program in memory already has an **END**, GETSUB creates an extra program containing only the line **01 END**.

---

* If you press ■ GTO · · before executing GETP, the calculator creates a blank program space at the end of program memory. If you then execute GETP, the copied program will replace the last, blank program—leaving the stored programs intact.

Example. Clear SAMPL - P from main memory and then recall it from extended memory.

Keystrokes	Display	
XEQ ALPHA CLP ALPHA	CLP _	Clears SAMPL - P.
ALPHA SAMPL - P	CLP SAMPL - P_	
ALPHA	572.0000	(Display shown assumes results remain from preceding example.)

Keystrokes	Display	
ALPHA SAMPL - P	SAMPL - P_	Specifies program file SAMPL - P.
ALPHA	572.0000	
XEQ ALPHA GETSUB ALPHA	572.0000	Copies SAMPL - P at end of program memory.
GTO ALPHA SAMPL - P	GTO SAMPL - P_	Positions calculator to SAMPL - P.
ALPHA	549.0000	
PRGM	01 LBL"SAMPL - P	First line of retrieved program.
PRGM	572.0000	

Operations Common to Data and ASCII Files

SEEKPTA	X	*rrr*	ALPHA	*data file name*
SEEKPTA	X	*rrr.ccc*	ALPHA	*ASCII file name*

Executing SEEKPTA (*seek pointer by ALPHA*) makes the named file the working file and repositions the pointer or pointers on the basis of the value in the X-register. In a data file, the integer part of the number in the X-register indicates the register that the pointer is positioned to. In an ASCII file, the integer portion of the number in the X-register positions the record pointer; the first three digits in the fractional portion position the character pointer.

Example. Make the ASCII file SMPL - AS the working file and reposition the record and character pointers.

Keystrokes	Display	
ALPHA SMPL - AS	SMPL - AS _	Specifies the file name.
ALPHA 6.013	6.013	Specifies the record (006) and the character (013)
XEQ ALPHA SEEKPTA	SEEKPTA_	
ALPHA	6.0130	Makes SMPL - AS the working file and positions the pointers to the "N" in "WILLIAM BATSON".

SEEKPT	X	*rrr*
SEEKPT	X	*rrr.ccc*

Executing SEEKPT (*seek pointers*) has the same effect on the working file that executing SEEKPTA has on the named file. Executing SEEKPTA with the ALPHA register empty is the same as executing SEEKPT

RCLPTA ALPHA *file name*

Executing RCLPTA (*recall pointer by ALPHA*) makes the named file the working file and returns the value or values of its pointer or pointers to the X-register. The value of the register pointer in a data file is returned as an integer. The values of the record and character pointers in an ASCII file are returned in the form *rrr.ccc*, where *rrr* is the value of the record pointer and *ccc* is the value of the character pointer.

RCLPT

Executing RCLPT (*recall pointers*) returns the value or values of the pointer or pointers in the working file to the X-register. Executing RCLPTA with the ALPHA register empty is the same as executing RCLPT.

Example. Recall the pointers from the ASCII file of the preceding example.

Keystrokes	**Display**	
0	0_	Clears display.
XEQ ALPHA RCLPT	XEQ RCLPT_	
ALPHA	6.0130	Recalls the pointer values set in the preceding example from the working file.

If you name a program file and execute RCLPTA, the number of bytes in the program is placed in the X-register. If the working file is a program file and you execute RCLPT, the number of bytes in the program is placed in the X-register.

Data File Operations

SAVER ALPHA *data file name*

Executing SAVER (*save registers*) copies all of the data storage registers in main memory to the named data file (or to the working file, if the ALPHA register is empty). The first register in main memory is copied to register 000 in the data file, the second is copied to register 001, and so forth. At the end of the process, the register pointer in the data file indicates either the next available register or the end of the file.

SAVERX X *bbb.eee*

Executing SAVERX (*save registers by X*) copies a block of main memory registers to the working data file. The block of main memory registers to be saved is designated by a value in the X-register in the form *bbb.eee*, where *bbb* is the register number of the first register and *eee* is the number of the last register. The registers are copied into the data file starting at the current pointer position. SAVERX moves the pointer to the register following the last register copied or to the end of the file. SAVERX will not be executed if there is not enough room in the working file for the block of registers to be copied.

[SAVEX]

Executing [SAVEX] (*save X-register*) copies the contents of the X-register into the working file at the register indicated by the pointer. (Any data in the indicated register is replaced by the data from the X-register.) After the data is copied, the pointer is moved to the next register.

The procedure on page 21 for creating file SAMPL - D and transferring data to it shows how [SAVEX] is used.

[GETR] **ALPHA** ***data file name***

Executing [GETR] (*get registers*) recalls the contents of the named file to main memory. The contents of register 000 in the named file are placed in main storage register 000, the contents of register 001 in the corresponding register in main memory, and so forth. Execution of [GETR] stops at either the last data register in main memory or at the end of the data file in extended memory. Executing [GETR] with the ALPHA register empty copies the working file to main memory if the file is a data file.

[GETRX] **X** ***bbb.eee***

Executing [GETRX] (*get registers by X*) copies data from the working file to a block of registers in main memory starting at register *bbb* and ending at register *eee*. Registers are copied from the working file starting at the current pointer position. If the desired register block is larger than the portion of the data file below the pointer, then *no* filling occurs and the error message **END OF FL** appears.

Executing [GETX] (*get to X-register*) copies the contents of the register indicated by the pointer in the working file to the X-register and moves the pointer to the next register.

[GETX]

Example: Recall the contents of a register in file SAMPL - D.

Keystrokes	Display	
8	8_	Pointer will be set to register 008.
[ALPHA] SAMPL - D	SAMPL - D_	File name.
[ALPHA]	8.0000	
[XEQ] [ALPHA] SEEKPTA	XEQ SEEKPTA	Makes SAMPL - D the working file and positions the pointer to register 008.
[ALPHA]	8.0000	
[XEQ] [ALPHA] GETX	XEQ GETX_	
[ALPHA]	734.0000	The contents of the register.

ASCII File Operations

Operations Involving Entire Records

APPREC ALPHA *text*

Executing APPREC (*append record*) appends the contents of the ALPHA register to the working file as the last record of the file. The record and character pointers are updated to point to the last character of the last record in the file. No action is taken if the ALPHA register is empty.

DELREC

Executing DELREC (*delete record*) deletes the record indicated by the record pointer in the working file. DELREC sets the character pointer to zero, but it does not change the record pointer.

Example: Delete an entry from SMPL - AS.

Keystrokes	Display	
2	2_	Pointer will be set to character 000 of record 002.
ALPHA SMPL - AS	SMPL - AS_	File name.
ALPHA	2.0000	
XEQ ALPHA SEEKPTA	XEQ SEEKPTA_	Makes SMPL - AS the working file and positions the pointers.
ALPHA	2.0000	
XEQ ALPHA DELREC	XEQ DELREC	
ALPHA	2.0000	"CARL LAFONG" is deleted. The record that was 003 is now 002.
XEQ ALPHA DELREC	XEQ DELREC	
ALPHA	2.0000	"156-2332" is deleted. Records move up. "BRUCE WAYNE" is now in record 002.

INSREC ALPHA *text*

Executing INSREC (*insert record*) inserts a record in front of the record indicated by the record pointer. INSREC sets the character pointer to the last character in the inserted record, but it does not change the record pointer.

Example: Insert an entry ahead of "BRUCE WAYNE" in SMPL-AS. (It is assumed that the preceding example has made SMPL-AS the working file and positioned the pointers to record 002 and character 000.)

Keystrokes	Display	
ALPHA 702-1133	702-1133_	Phone number.
ALPHA	2.0000	
XEQ ALPHA INSREC	XEQ INSREC_	Phone number is inserted in record 002; data in other records moves down. Record pointer is at the end of the phone number (which is now record 002).

Keystrokes	Display	
[ALPHA]	2.0000	
[ALPHA] BILL BAILEY	BILL BAILEY_	Name.
[ALPHA]	2.0000	
[XEQ] [ALPHA] INSREC	XEQ INSREC_	Name is inserted before phone number; phone number and data in other records move down.
[ALPHA]	2.0000	
[XEQ] [ALPHA] SAMPL - P	XEQ SAMPL - P_	Runs the program to list the file.
[ALPHA]	HARVEY KECK	
	555-1234	
	BILL BAILEY	
	702-1133	
	BRUCE WAYNE	
	205-4423	
	WILLIAM BATSON	
	602-9991	
	END OF FL	

Operations Involving Characters Within Records

[APPCHR] **ALPHA** [*text*]

Executing [APPCHR] (*append characters*) appends the contents of the ALPHA register at the end of the record indicated by the record pointer in the working file. [APPCHR] also sets the character pointer to the end of the record. No action is taken if the ALPHA register is empty.

Example: Modify a record in SMPL - AS by appending " JR." to record 000, "HARVEY KECK". Assume SMPL - AS is still the working file.

Keystrokes	Display	
0	0_	Specifies record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Move pointers.
[ALPHA]	0.0000	
[ALPHA] JR.	JR.	Text to be appended.
[ALPHA]	0.0000	
[XEQ] [ALPHA] APPCHR	XEQ APPCHR_	Appends characters to record.
[ALPHA]	0.0000	
0	0_	Character 000 of record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	
[ALPHA]	0.0000	Move pointers.
[XEQ] [ALPHA] GETREC	XEQ GETREC_	Recalls record 000.
[ALPHA]	0.0000	
[ALPHA]	HARVEY KECK JR	
[ALPHA]	0.0000	

[DELCHR] **X** [*number of characters*]

Executing [DELCHR] (*delete characters*) deletes characters up to the number specified in the X-register, starting from the current pointer position in the working file. [DELCHR] will not delete characters beyond the end of the current record, and it does not change the record pointer.

Example. Modify a record in SMPL - AS by changing "HARVEY KECK JR." to "H KECK JR."

Keystrokes	Display	
[.]001	.001	Location of "A" in "HARVEY".
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Moves pointers.
[ALPHA]	0.0010	
5	5_	Five characters.
[XEQ] [ALPHA] DELCHR	XEQ DELCHR	Deletes the characters.
[ALPHA]	5.0000	
0	0_	Character 000 of record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Resets pointer to beginning of record.
[ALPHA]	0.0000	
[XEQ] [ALPHA] GETREC	XEQ GETREC_	Recalls the current record.
[ALPHA]	0.0000	
[ALPHA]	H KECK JR.	The record has been modified.
[ALPHA]	0.0000	

[INSCHR] **ALPHA** [*text*]

Executing [INSCHR] (*insert characters*) inserts the contents of the ALPHA register ahead of the current character in the working file. [INSCHR] sets the character pointer to the last character inserted.

Example. Modify a record in SMPL-AS by placing a period after "H" in "H KECK JR."

Keystrokes	Display	
[.]001	.001_	Location of "H" in record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Moves pointers.
[ALPHA]	0.0010	
[ALPHA] [■] [.]	._	Character to be inserted (a period).
[ALPHA]	0.0010	
[XEQ] [ALPHA] INSCHR	XEQ INSCHR_	Inserts the characters.
[ALPHA]	0.0010	
0	0_	Character 000 of record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Resets pointer to beginning of record.
[ALPHA]	0.0000	
[XEQ] [ALPHA] GETREC	XEQ GETREC_	Recalls the current record.
[ALPHA]	0.0000	
[ALPHA]	H. KECK JR.	The period has been inserted.
[ALPHA]	0.0000	

Searching an ASCII File

[POSFL] **ALPHA** [*text*]

The [POSFL] (*position in file*) function scans the working file, starting from the current pointer position, for a string of text that matches the contents of the ALPHA register. If a match is found, the pointers are

repositioned to the first character of the matching text and the pointer value is returned to the X-register. If no match is found, the pointer is not moved and the number −1 replaces the value in the X-register. The previous value in the X-register is preserved in the LAST X register.

Example. Search SMPL - AS for the text string "BATSON".

Keystrokes	Display	
0	0_	Character 000 of record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Moves pointers to start of file.
[ALPHA]	0.0000	
[ALPHA] BATSON	BATSON_	Target text.
[ALPHA]	0.0000	
[XEQ] [ALPHA] POSFL	XEQ POSFL_	
[ALPHA]	6.0080	The target string starts at character 008 of record 006.

Transferring ASCII Files to Main Memory

[GETREC]

Executing [GETREC] (*get record*) clears the ALPHA register and recalls up to 24 characters from the working file into the ALPHA register. Characters are copied starting from the current pointer position.

After the transfer, the pointer is set to the character following the last one copied. If there are fewer than 24 characters between the pointer and the end of the record, transfer stops at the end of the record. When [GETREC] is executed, flag 17 is set if the end of the record is not reached. If the end of the record *is* reached, flag 17 is cleared. This is useful if you have a printer with a line length greater than 24 characters connected to the calculator via HP-IL (the Hewlett-Packard Interface Loop). If you follow each [GETREC] with the HP-IL command [OUTA], the contents of the ALPHA register are output to the printer without a terminating carriage return and linefeed if flag 17 is set. This lets you print the contents of a record longer than 24 characters on a single line.

[ARCLREC]

Executing [ARCLREC] (*alpha recall record*) appends a record or part of a record to the ALPHA register until the ALPHA register is full. When [ARCLREC] is executed, flag 17 is set if the end of the record is not reached. If the end of the record *is* reached, flag 17 is cleared. [ARCLREC] sets the character pointer to the character following the last one copied.

Example. Append a record from SMPL - AS to data already in the ALPHA register by placing a name and phone number on one line.

Keystrokes	Display	
0	0_	Character 000 of record 000.
[XEQ] [ALPHA] SEEKPT	XEQ SEEKPT_	Moves the pointers.
[ALPHA]	0.0000	
[XEQ] [ALPHA] GETREC	XEQ GETREC_	Clears ALPHA, places the contents of the current record in the ALPHA register, and advances the record pointer.
[ALPHA]	0.0000	

Keystrokes	Display	
[ALPHA] ■ [APPEND] [SPACE]	H. KECK JR. _	
[ALPHA]	0.0000	
[XEQ] [ALPHA] ARCLREC	XEQ ARCLREC _	Appends the contents of the current record to ALPHA.
[ALPHA]	0.0000	
[ALPHA]	H. KECK JR.	
	555-1234	
[ALPHA]	0.0000	

Transferring Data Between Extended Memory and Mass Storage Devices

You can transfer ASCII files that have been created in extended memory to mass storage devices, such as a tape cassette in the HP 82161A Digital Cassette Drive, using the HP 82160A HP-IL Module. The cassette offers a permanent storage medium from which files can be recalled when necessary.

[SAVEAS]	**ALPHA**	***extended memory file name, mass storage file name***
[SAVEAS]	**ALPHA**	***extended memory file name***

Executing [SAVEAS] (*save ASCII*) copies the named ASCII file in extended memory to the specified destination file in mass storage. If you omit the comma and the destination file name and simply enter the source file name in the ALPHA register, the file in extended memory will be copied to a file with the same name in mass storage, if one exists. [SAVEAS] does not create a file in mass storage; that must be done using the [CREATE] function in the HP-IL module.

[GETAS]	**ALPHA**	***mass storage file name, extended memory file name***
[GETAS]	**ALPHA**	***mass storage file name***

Executing [GETAS] (*get ASCII*) copies the named ASCII file in mass storage to the specified destination file in extended memory. If you omit the comma and the destination file name and simply enter the source file name in the ALPHA register, the file in mass storage will be copied to a file with the same name in extended memory, if one exists. Before you can execute [GETAS], you must first have created the file in extended memory using [CRFLAS].

Execution of either [SAVEAS] or [GETAS] stops when the end of either the source or destination file is reached. If the destination file is too small, an error will be generated, but some data will be copied.

Programming and the Extended Functions/Memory Module

All functions provided by the HP 82180A Extended Functions/Memory Module can be entered whenever the module is plugged into the calculator. While the extended functions/memory module is connected, program lines with extended functions are displayed and printed as standard functions.

If the module is disconnected later, these program lines are displayed and printed as **XROM** followed by two identification numbers. The first number, 25, indicates that the function are provided in the extended functions/memory module. The second number identifies the particular function. The XROM numbers for the functions in the extended functions/memory module are listed below.

Function	XROM Number	Function	XROM Number	Function	XROM Number
ALENG	XROM 25,01	GETKEY	XROM 25,17	RCLPT	XROM 25,33
ANUM	XROM 25,02	GETP	XROM 25,18	RCLPTA	XROM 25,34
APPCHR	XROM 25,03	GETR	XROM 25,19	REGMOVE	XROM 25,35
APPREC	XROM 25,04	GETREC	XROM 25,20	REGSWAP	XROM 25,36
ARCLREC	XROM 25,05	GETRX	XROM 25,21	SAVEAS	XROM 25,37
AROT	XROM 25,06	GETSUB	XROM 25,22	SAVEP	XROM 25,38
ATOX	XROM 25,07	GETX	XROM 25,23	SAVER	XROM 25,39
CLFL	XROM 25,08	INSCHR	XROM 25,24	SAVERX	XROM 25,40
CLKEYS	XROM 25,09	INSREC	XROM 25,25	SAVEX	XROM 25,41
CRFLAS	XROM 25,10	PASN	XROM 25,26	SEEKPT	XROM 25,42
CRFLD	XROM 25,11	PCLPS	XROM 25,27	SEEKPTA	XROM 25,43
DELCHR	XROM 25,12	POSA	XROM 25,28	SIZE?	XROM 25,44
DELREC	XROM 25,13	POSFL	XROM 25,29	STOFLAG	XROM 25,45
EMDIR	XROM 25,14	PSIZE	XROM 25,30	X<>F	XROM 25,46
FLSIZE	XROM 25,15	PURFL	XROM 25,31	XTOA	XROM 25,47
GETAS	XROM 25,16	RCLFLAG	XROM 25,32		

If a module function is entered, using XEQ, into a program line while the module is not connected, the function is recorded, displayed, and printed as **XEQ**^{T} followed by the function name. Program execution will be slowed by lines of this form because the calculator will search for a matching ALPHA label or function name—first in program memory, then in each module plugged in.

Care, Warranty, and Service Information

Module Care

> **CAUTION**
>
> Always turn off the calculator before connecting or disconnecting any module or peripheral. Failure to do so could result in damage to the calculator or disruption of the system's operation.

- Keep the contact area of the module free of obstructions. Should the contacts become dirty, carefully brush or blow the dirt out of the contact area. Do not use any liquid to clean the contacts.
- Store the module in a clean, dry place.
- Always turn off the calculator before installing or removing any module or peripherals.
- Observe the following temperature specifications:

  Operating: 0° to 45° C (32° to 113° F).

  Storage: −40° to 75° C (−40° to 167° F).

Limited One-Year Warranty

What We Will Do

The HP 82180A Extended Functions/Memory Module is warranted by Hewlett-Packard against defects in materials and workmanship affecting electronic and mechanical performance, but not software content, for one year from the date of original purchase. If you sell your unit or give it as a gift, the warranty is transferred to the new owner and remains in effect for the original one-year period. During the warranty period, we will repair or, at our option, replace at no charge a product that proves to be defective, provided you return the product, shipping prepaid, to a Hewlett-Packard service center.

What Is Not Covered

This warranty does not apply if the product has been damaged by accident or misuse or as the result of service or modification by other than an authorized Hewlett-Packard service center.

No other express warranty is given. The repair or replacement of a product is your exclusive remedy. **ANY OTHER IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS IS LIMITED TO THE ONE-YEAR DURATION OF THIS WRITTEN WARRANTY.** Some states, provinces, or countries do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to you. **IN NO EVENT SHALL HEWLETT-PACKARD COMPANY BE LIABLE FOR CONSEQUENTIAL DAMAGES.** Some states, provinces, or countries do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This warranty gives you specific legal rights, and you may also have other rights which vary from state to state, province to province, or country to country.

Warranty for Consumer Transactions in the United Kingdom

This warranty shall not apply to consumer transactions and shall not affect the statutory rights of a consumer. In relation to such transactions, the rights and obligations of Seller and Buyer shall be determined by statute.

Obligation to Make Changes

Products are sold on the basis of specifications applicable at the time of manufacture. Hewlett-Packard shall have no obligation to modify or update products once sold.

Warranty Information

If you have questions about this warranty, please contact Hewlett-Packard at one of the following locations.

In the United States. Call (503) 757-2002 or write to:

Hewlett-Packard Co.
Calculator Service Center
1030 N.E. Circle Blvd.
Corvallis, OR 97330

In Europe. Call (022) 83 81 11 or write to:

Hewlett-Packard S.A.
150, route du Nant-d'Avril
P.O. Box CH-1217 Meyrin 2
Geneva
Switzerland

Note: Do *not* send computers to this address for repair.

In other countries. Call (415) 857-1501 in the U.S.A. or write to:

Hewlett-Packard Intercontinental
3495 Deer Creek Road
Palo Alto, California 94304
U.S.A.

Note: Do *not* send computers to this address for repair.

Service

Hewlett-Packard maintains service centers in most major countries throughout the world. You may have your unit repaired at a Hewlett-Packard service center any time it needs service, whether the unit is under warranty or not. There is a charge for repairs after the one-year warranty period.

Hewlett-Packard products are normally repaired and reshipped within five (5) working days of receipt at any service center. This is an average time and could vary depending upon the time of year and the work load at the service center. The total time you are without your unit will depend largely on the shipping time.

Obtaining Repair Service in the United States

For repair service in the United States, ship your computer to:

Hewlett-Packard Co.
Calculator Service Center
1030 N.E. Circle Blvd.
Corvallis, Oregon 97330

or mail it to:

Hewlett-Packard Co.
Calculator Service Center
P.O. Box 999
Corvallis, Oregon 97339

The telephone number for the Calculator Service Center is (503) 757-2002.

Obtaining Repair Service in Europe

Service centers are maintained at the following locations. For countries not listed, contact the dealer where you purchased your unit.

AUSTRIA
HEWLETT-PACKARD Ges.m.b.H.
Kleinrechner-Service
Wagramerstrasse-Lieblgasse 1
A-1220 Wien (Vienna)
Telephone: (0222) 23 65 11

BELGIUM
HEWLETT-PACKARD BELGIUM SA/NV
Woluwedal 100
B-1200 Brussels
Telephone: (02) 762 32 00

DENMARK
HEWLETT-PACKARD A/S
Datavej 52
DK-3460 Birkerod (Copenhagen)
Telephone: (02) 81 66 40

EASTERN EUROPE
Refer to the address listed under Austria.

FINLAND
HEWLETT-PACKARD OY
Revontulentie 7
SF-02100 Espoo 10 (Helsinki)
Telephone: (90) 455 02 11

FRANCE
HEWLETT-PACKARD FRANCE
Division Informatique Personnelle
S.A.V. Calculateurs de Poche
F-91947 Les Ulis Cedex
Telephone: (6) 907 78 25

GERMANY
HEWLETT-PACKARD GmbH
Kleinrechner-Service
Vertriebszentrale
Berner Strasse 117
Postfach 560 140
D-6000 Frankfurt 56
Telephone: (611) 50041

ITALY
HEWLETT-PACKARD ITALIANA S.P.A.
Casella postale 3645 (Milano)
Via G. Di Vittorio, 9
I-20063 Cernusco Sul Naviglio (Milan)
Telephone: (2) 90 36 91

NETHERLANDS
HEWLETT-PACKARD NEDERLAND B.V.
Van Heuven Goedhartlaan 121
NL-1181 KK Amstelveen (Amsterdam)
P.O. Box 667
Telephone: (020) 472021

NORWAY
HEWLETT-PACKARD NORGE A/S
P.O. Box 34
Oesterndalen 18
N-1345 Oesteraas (Oslo)
Telephone: (2) 17 11 80

SPAIN
HEWLETT-PACKARD ESPANOLA S.A.
Calle Jerez 3
E-Madrid 16
Telephone: (1) 458 2600

SWEDEN
HEWLETT-PACKARD SVERIGE AB
Skalholtsgatan 9, Kista
Box 19
S-163 93 Spanga (Stockholm)
Telephone: (08) 750 2000

SWITZERLAND
HEWLETT-PACKARD (SCHWEIZ) AG
Kleinrechner-Service
Allmend 2
CH-8967 Widen
Telephone: (057) 31 21 11

UNITED KINGDOM
HEWLETT-PACKARD Ltd
King Street Lane
GB-Winnersh, Wokingham
Berkshire RG11 5AR
Telephone: (0734) 784 774

International Service Information

Not all Hewlett-Packard service centers offer service for all models of HP products. However, if you bought your product from an authorized Hewlett-Packard dealer, you can be sure that service is available in the country where you bought it.

If you happen to be outside of the country where you bought your unit, you can contact the local Hewlett-Packard service center to see if service is available for it. If service is unavailable, please ship the unit to the address listed above under Obtaining Repair Service in the United States. A list of service centers for other countries can be obtained by writing to that address.

All shipping, reimportation arrangements, and customs costs are your responsibility.

Further Information

Service contracts are available. For information about service contracts, please contact the Calculator Service Center in Corvallis, Oregon.
Computer product circuitry and design are proprietary to Hewlett-Packard. Service manuals are not available to customers.
For more information about service, please contact your nearest Hewlett-Packard service center.

When You Need Help

Technical Assistance. For technical assistance with your computer,

call:
(503) 757-2004
8 a.m. to 3 p.m.
Pacific time

or write to:
Hewlett-Packard Co.
Portable Computer Division
Calculator Technical Support
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Product Information. For information about Hewlett-Packard products and prices, contact your local Hewlett-Packard dealer. For the name of the dealer nearest you, or to order free literature about Hewlett-Packard products,

call toll-free:
(800) FOR-HPPC
(800) 367-4772

or write to:
Hewlett-Packard Co.
Personal Computer Group
PCG Telemarketing
10520 Ridgeview Court
Cupertino, CA 95014

Error Messages

This is a list of messages and errors relating to the functions provided by the extended functions/memory module. When any of these errors are generated, the function attempted is not performed, except as noted.

Display	Functions	Meaning
CHKSUM ERR	GETP GETSUB	Part of the program file has been lost.
DATA ERROR	AROT POSA XTOA X<>F	Number in the X-register is greater than 255.
	PSIZE SEEKPT SEEKPTA	Number in the X-register is greater than 999.
	CRFLAS CRFLD	Number in the X-register is 0. (Attempt has been made to create a file 0 registers in length.)
	STOFLAG	The data in the X-register (or the Y-register, if the X-register contains a range of flags in the form *bb.ee*) is not data that was obtained by executing RCLFLAG.
DIR EMPTY	EMDIR	There are no files in extended memory.
DUP FL	CRFLAS CRFLD	A file of the same name already exists in extended memory. The file of the same name becomes the working file. ASCII, data and program files cannot share a common name. Hint: Use suffixes to distinguish related files (SAMPL - D; SAMPL - P).
END OF FL	APPCHR APPREC ARCLREC DELCHR DELREC GETREC GETRX GETX INSCHR INSREC SAVER SAVERX SAVEX SEEKPT SEEKPTA	An attempt has been made to position the pointer, to read, write, or delete past the end of the file. For SAVER and SEEKPTA the named file becomes the working file, but the file and pointers are not changed. For APPCHR, APPREC, INSCHR, and INSREC there is not enough room to append the characters or record.
	GETAS SAVEAS	File transfer was not completed because the end of the destination file was encountered before the end of the source file. Part of the file is transferred.

Display	Functions	Meaning
END OF REC	SEEKPT SEEKPTA	An attempt has been made to position the character pointer past the end of the current record.
FL NOT FOUND	CLFL FLSIZE GETAS GETP GETR GETSUB PURFL RCLPTA SAVEAS SAVER SEEKPTA	The filename in the ALPHA register does not exist in extended memory or, if the ALPHA register is empty, then there is no working file in extended memory.
	APPCHR APPREC ARCLREC DELCHR DELREC GETREC GETRX GETX INSCHR INSREC POSFL RCLPT SAVERX SAVEX SEEKPT	There is no working file in extended memory. Hint: This would happen if there were no files in extended memory or if you just purged a file.
FL TYPE ERR	APPCHR APPREC ARCLREC CLFL DELCHR DELREC GETAS GETP GETR GETREC GETRX GETSUB GETX INSCHR INSREC POSFL SAVEAS SAVER SAVERX SAVEX SEEKPT	Either the working file or the file specified in the ALPHA register is of the wrong type for the function attempted. For example, you can't use CLFL on a program file; you'll get this message.

Display	Functions	Meaning
	SEEKPTA	
KEYCODE ERR	PASN	The number in the X-register does not correspond to an assignable key. Hint: Did you try to assign a function to the shift key (31)?
NAME ERR	CLFL CRFLAS CRFLD GETAS GETP GETSUB PURFL SAVEAS SAVEP	ALPHA register is empty.
	PCLPS SAVEP	The named program does not exist in main memory.
NO DRIVE	GETAS SAVEAS	No HP-IL module is plugged into the calculator, or no mass storage device is on the interface loop.
NO ROOM	CRFLAS CRFLD	There is not enough space left in extended memory for a file of the size specified by the number in the X-register.
	SAVEP	There is not enough space in extended memory to store the program.
	GETP GETSUB	There is not enough space in main memory to hold the program.
	PSIZE	(When executed as a program instruction.) There is not enough space in main memory.
NONEXISTENT	All Functions	The extended functions/memory module is not plugged in or is malfunctioning.
	GETRX REGMOVE REGSWAP SAVERX	One or more registers specified by the number in the X-register does not exist in main memory.
	STOFLAG	One or more of the flags specified by the number in the X-register is out of the range 0-43.
PACKING TRY AGAIN	GETP GETSUB	(When executed from the keyboard.) There is not enough program space in main memory.
	PSIZE	(When executed from the keyboard.) There is not enough space in main memory.
	PASN	There is not enough space in main memory.
REC TOO LONG	APPCHR INSCHR	If the function were completed, the resulting record would be more than 254 characters long.
ROM	SAVEP PCLPS	The program named is in ROM.

Appendix C

Null Characters

Null Characters and the ALPHA Register

The null character in your calculator is the ¯ (overbar) and corresponds to character code 0.* Normally the calculator does not display null characters. However, under certain conditions you can use the Extended Functions/Memory module to place null characters in ALPHA data strings. (This allows you to include nulls in data strings to be transferred to HP-IL devices.)

When you display the ALPHA register, any null characters to the right of the first non-null character will be displayed.

Treatment of Null Characters

Because the calculator attaches special significance to null characters in the ALPHA register, there are several functions that do not operate normally if a null exists in a data string used by ALPHA functions:

- Nulls in ALPHA displays appear as an ¯ (overbar) character and are printed as a * (the character corresponding to character codes 0 and 10) by the HP 82143A and HP 82162A Printers.
- If you execute [APPEND] (refer to the label on the back of the calculator) when the last character in the ALPHA register is a null, the ALPHA *display* appears blank. However, the contents of the ALPHA *register* are not affected. Thus, the characters that are entered after executing [APPEND] are appended properly to the existing ALPHA data string. To restore the ALPHA display, execute [AVIEW] or switch ALPHA mode off, then on.
- If you store an ALPHA string containing nulls in a data register, the nulls do not appear if you use [VIEW] to display that register. If you print the contents of that register, only the characters to the left of the first null are printed. The remaining characters are ignored by the printer. (All characters in the string, including the nulls, remain properly stored in the data register and reappear in the ALPHA register if the contents of the data register are recalled using [ARCL].)
- If a string containing one or more nulls is rotated so that a null becomes the leftmost character, that null and all nulls that immediately follow it are lost.
- If the last character in the ALPHA register is a null, and if the calculator is in ALPHA append mode (the _ prompt appears to the right of the null character), pressing [←] clears the entire ALPHA register.
- If an ALPHA string in the X-register contains a null when you execute [POSA], the calculator searches the ALPHA register only for that portion of the string that is to the left of the first null in the string.
- Any null in a file name that is entered in the ALPHA register is ignored.

* The null character and character code should not be confused with the **NULL** message that is displayed when a calculator function key is held down for more than about one-half second.

Function Index

Function	Description	ALPHA Register	X-Register
[ALENG] **(page 15)**	Returns length of the string in the ALPHA register to the X-register.		
[ANUM] **(page 15)**	Searches the ALPHA register for an ALPHA formatted number. Returns its value to the X-register.		
[APPCHR] **(page 30)**	Appends contents of the ALPHA register as a new record at the end of the current record in extended memory working file.	Source string.	
[APPREC] **(page 29)**	Appends contents of the ALPHA register as a new record at the end of the current working file in extended memory.	Source string.	
[ARCLREC] **(page 32)**	Appends record or portion of record from working file in extended memory to the ALPHA register. Copies from current pointer position until either the ALPHA register is full or record is exhausted.		
[AROT] **(page 16)**	Rotates contents of ALPHA register.		Number of characters by which ALPHA is to be rotated. (Positive numbers rotate to the left; negative to the right.)
[ATOX] **(page 14)**	Deletes left-most character in ALPHA register, converts it to numeric character code, and places code in X-register.		
[CLFL] **(page 24)**	Clears a data or ASCII file.	File name.	
[CLKEYS] **(page 13)**	Clears all key assignments.		
[CRFLAS] **(page 24)**	Creates an ASCII file in extended memory.	File name.	Number of registers.
[CRFLD] **(page 24)**	Creates a data file in extended memory.	File name.	Number of registers.
[DELCHR] **(page 30)**	Deletes characters in working file, starting at current pointer position.		Number of characters to be deleted.
[DELREC] **(page 29)**	Deletes the record indicated by the record pointer in the working file.		
[EMDIR] **(page 23)**	Lists the directory of extended memory files.		
[FLSIZE] **(page 25)**	Returns the number of registers in named file to X-register.	File name.	
[GETAS] **(page 33)**	Copies named ASCII file from mass storage to extended memory.	Mass storage file name, extended memory file name.	
[GETKEY] **(page 16)**	Halts program execution until a key is pressed or approximately 10 seconds		

Function	Description	ALPHA Register	X-Register
	elapses. Puts keycode in X-register if key is pressed, puts 0 in X-register if no key is pressed.		
GETP **(page 25)**	Replaces last program in main memory with contents of named program file.	Program file name.	
GETR **(page 28)**	Copies entire data file to main memory registers, beginning with register 00.	Data file name.	
GETREC **(page 32)**	Clears ALPHA register and copies record or portion of record from working file to ALPHA register. Characters are copied from current pointer position until 24 characters have been copied or the end of the record is reached.		
GETRX **(page 28)**	Copies all or a portion of the registers in the working data file in extended memory to the designated registers in main memory. (Copy starts from the current pointer position in the working file.)		*bbb.eee* (beginning and ending registers in main memory into which data is to be transferred).
GETSUB **(page 25)**	Copies named program from extended memory to end of program storage in main memory.	Program file name.	
GETX **(page 28)**	Copies current register in working file to X-register.		
INSCHR **(page 31)**	Inserts characters in ALPHA register into working ASCII file in front of the current character.	Characters to be inserted.	
INSREC **(page 29)**	Inserts characters in ALPHA register as a new record in front of the current record in the working ASCII file.	Characters to be inserted.	
PASN **(page 13)**	Programmable ASN function.	Function or Program name.	Keycode.
PCLPS **(page 17)**	Deletes named program and all following it from main memory.	Program name.	
POSA **(page 15)**	Scans the ALPHA register for the character(s) in the X-register and returns the position of the first character to the X-register (−1 if no match).		ALPHA substring or character code.
POSFL **(page 31)**	Searches the working ASCII file for a sub-string matching the string in the ALPHA register. Returns the record and character pointer locations to the X-register of the first character of the substring if a match is found. Returns a −1 if no match is found.	Target string.	
PSIZE **(page 17)**	Programmable SIZE function.		Number of data storage registers to be allocated.
PURFL **(page 24)**	Purges (deletes) the named file from extended memory.	File name.	
RCLFLAG **(page 11)**	Recalls data to the X-register representing the status of flags 00-43.		
RCLPT **(page 27)**	Recalls a number representing the pointer positions in the working file to the X-register (*rrr* for data files, *rrr.ccc* for ASCII files), or		

Function	Description	ALPHA Register	X-Register
	recalls the number of bytes in a program file.		
[RCLPTA] **(page 27)**	Recalls a number representing the pointer positions in the named data or ASCII file (or representing the number of bytes in the program in a program file) to the X-register (*rrr* for data files and program files, *rrr.ccc* for ASCII files). Makes the named file the working file.	File name.	
[REGMOVE] **(page 11)**	Copies *nnn* main memory registers beginning with register *sss* to a new location beginning at register *ddd*.		*sss.dddnnn*
[REGSWAP] **(page 11)**	Swaps *nnn* registers beginning with register *sss* with *nnn* registers beginning with register *ddd*.		*sss.dddnnn*
[SAVEAS] **(page 33)**	Copies ASCII file from extended memory to mass storage.	Extended memory file name, mass storage file name.	
[SAVEP] **(page 25)**	Copies the named program from main memory to a program file in extended memory.	Program name, file name.	
[SAVER] **(page 27)**	Copies all main memory registers to the named data file.	File name.	
[SAVERX] **(page 27)**	Copies a block of main memory registers (indicated by the number in the X-register) to the working data file.		*bbb.eee* (beginning and ending registers of block to be saved).
[SAVEX] **(page 28)**	Copies the contents of the X-register to the working data file at the current pointer position.		
[SEEKPT] **(page 26)**	Positions the pointers in the working file to the locations indicated by the number in the X-register.		*rrr* (data files) or *rrr.ccc* (ASCII files).
[SEEKPTA] **(page 26)**	Positions the pointers in the named file to the locations indicated by the number in the X-register.	File name.	*rrr* (data files) or *rrr.ccc* (ASCII files).
[SIZE?] **(page 17)**	Returns the number of data storage registers in main memory to the X-register.		
[STOFLAG] **(page 11)**	Restores the status of flags 0-43 (or a block within that group). Uses the data obtained by executing [RCLFLAG].		flag status or *bb.ee* (beginning and ending flags in block to be restored; Y-register contains flag status).
[X<>F] **(page 12)**	Exchanges the contents of the X-register with the status of flags 0-7.		Code number (0-255)
[XTOA] **(page 14)**	Either converts the number in the X-register to its equivalent character and appends the character to the ALPHA register, or appends the alpha data string in the X-register to the ALPHA register.		Character number (0–255) or alpha string.

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

European Headquarters
150, Route du Nant-d'Avril
P.O. Box, 1217 Meyrin 2
Geneva, Switzerland

HP-United Kingdom
(Pinewood)
GB-Nine Mile Ride, Wokingham
Berkshire RG11 3LL

82180-90001 Rev. D English

Printed in Singapore 9/85