HP-71B BASIC Reference

Programmer's guide · 249 keyword entries grouped by function · from the HP-71 Owner's Manual

Page 1 explains the HP-71 as a machine and language: keyboard and CALC mode, program-line syntax, variables and strings, operators, number formats and IEEE math exceptions, branching, subprograms, errors, flags, input, output formatting, files, data files, editing, clock and statistics. The following pages index every keyword from the Owner's Manual's Keyword Index by category. Hover any keyword for its syntax, description and Owner's Manual page. A final section covers the Emu71 emulator only.

Type: ● Statement ● Function (usable in expressions) ● Operator · OM p.nn = HP-71 Owner's Manual page

Sources: HP-71 Owner's Manual (HP 00071-90001 Rev. D, Feb 1985) — sections 1–14, appendices A–C, Keyword Index. Keywords in more than one category are listed in each. Emu71 section: Emu71 v1.19 manual and MK71MAC.TXT (C. Gießelink).

HP-71B Programmer's Reference

BASIC language, machine features, idioms — syntax, semantics

Keyboard & Modes

Shifts: f (gold, above keys) · g (blue, below keys). Release the shift before or hold it while pressing the key. Letters are uppercase; g+letter gives the other case; f LC toggles lowercase mode (flag –15).

KEY	ACTION
END LINE	Execute / enter the line (like RETURN)
ATTN	Clear display; halt a running program (SUSP). Also the ON key
f OFF / ON	Power off / on (auto-off after 10 min unless flag –3 set)
f CALC	Toggle CALC mode ↔ BASIC mode
RUN	Run current file (or resume) · in CALC: show next partial result
f CONT	Continue a suspended program
f SST	Single-step one statement (or one CALC step)
f USER	Toggle User keyboard · g 1USER = User kbd for next key only
g CMDS	Command Stack: last 5 commands; ▲▼ to browse, END LINE to run
f VIEW	Hold next key → shows its DEF KEY string
g ERRM	Show last error/warning message
f BACK	Backspace (delete left) · in CALC: undo last operation
f -CHAR / f -LINE	Delete char under cursor / delete to end of line
f I/R	Toggle Insert / Replace cursor
◀ ▶ · g ◀ g ▶	Move cursor · jump to start / end of the 96-char line
▲ ▼	Previous / next line (FETCH neighbours, CAT entries, Command Stack)
g CTRL+key	Control character (CTRL-M = CR, CTRL-J = LF, CTRL-g [= ESC)
ON+/	INIT: 1 (system reset, memory kept) · 2 (self-test) · 3 (memory reset: clears main RAM; independent RAM survives)

Display: 22-character window on a 96-character line; ← → annunciators show the line extends. Typing aids: f+top-row keys type keywords (e.g. f S = `DISP`).

ANNUNCIATORS

f g shift pending · **USER** User keyboard · **RAD** radians · **0 1 2 3 4** user flags 0–4 set · **BAT** low battery · **((·))** alarm · **PRGM** program running · **SUSP** suspended · **CALC** CALC mode · **AC** reserved.

CALC Mode

f CALC toggles. Evaluates numeric expressions with intermediate results as you type — same variables, functions, precedence and user-defined single-line functions as BASIC. Assignments allowed (`A=3*4`).

```
SIN(30          auto-supplies ")" — closing parens optional
()              implied result: value of the last evaluated
                expression
MAX(3,          flashing "," reminds of missing arguments
2+3*            partial results shown as precedence allows
```

KEY	IN CALC MODE
END LINE	Finish and evaluate
RUN / f SST	Show next intermediate step separately
f BACK	Backward execution: undo operands/operators one at a time
▲	Recall whole expression into Command Stack for editing; ◅ shows it was already evaluated

Not available in CALC: strings, DTH$/HTD, multi-line user functions, statements other than assignment, program lines. Don't insert/remove modules while CALC mode is on (memory reset).

Program Lines & Syntax

```
10 DISP "HI" @ BEEP @ WAIT 1          line number 1–9999;
                                      @ joins statements
20 ! comment   /   20 REM comment     remarks (! also
                                      allowed after a
                                      statement)
30 'AGAIN': X=X+1 @ IF X<5 THEN       label = up to 8
AGAIN                                 letters/digits +
                                      colon; quotes added
                                      by the 71
40 IF A>B THEN 100 ELSE DISP "NO"     THEN/ELSE take a
                                      line, label or
                                      statement(s)
LET A=1  ·  A=1  ·  A,B,C=0           LET optional;
                                      multiple assignment
```

- Keywords may be typed in lower case; the HP-71 uppercases keywords, file names and variable names. Strings keep case; quote with `"…"` or `'…'`.
- Line length 96 chars. Program lines are kept sorted; enter in any order. Multi-statement lines save 2 bytes each.
- **Variables**: a letter or letter+digit (`A` , `X1`); strings add `$` (`N$`); arrays `A(3)` , `M(2,3)` (max 2 dims). Numeric and string of the same name are distinct.
- **Files**: name = 1–8 letters/digits, first a letter; optional device `:MAIN` , `:PORT(n)` , `:CARD` . Reserved: `ALL CARD KEYS TO INTO PCRD` .
- Numbers: `1.5E-3` , `-7` ; 12 significant digits, exponent ±499. Hex via `HTD("FF")` , `DTH$(255)` .

Variables, Arrays & Strings

```
DIM A(10), B(3,4), N$[80]                numeric arrays (default OPTION BASE 0), string max length
REAL X, Y(5)  ·  SHORT S(100)  ·  INTEGER I,J   declare precision (also dimensions arrays)
OPTION BASE 1                            lowest subscript 1 (default 0); must precede DIM
DESTROY A, B$  ·  DESTROY ALL            free variables (ALL: every variable, incl. arrays)
A$[3,5]  ·  A$[4]  ·  A$[3,3]="x"        substring chars 3–5 · from 4 to end · replace
A$ & B$  ·  LEN(A$)  ·  POS(A$,"x")      concatenate · length · position (0 if absent)
VAL("12.5")  ·  STR$(2/3)  ·  NUM("A")  ·  CHR$(65)   string↔number, char code↔char
UPRC$(A$)  ·  VER$  ·  A$="ab" & CHR$(13)   uppercase · ROM version string · control chars
```

TYPE	RANGE	STORAGE
REAL (default)	±1E-499 … ±9.99999999999E499, 12 digits	8 bytes
SHORT	5 digits, ±9.9999E499	4 bytes
INTEGER	–99999 … +99999	3 bytes
String	default max 32 chars; DIM up to memory	len+2

Undeclared numeric variables are REAL scalars created on first use (value 0); strings default to "" with max length 32. Arrays are dimensioned implicitly to (10) / (10,10) on first use. Comparing strings uses character codes.

Operators & Precedence

LEVEL	OPERATORS (HIGHEST FIRST)
1	( ) — innermost first
2	Functions: SIN, LOG, FACT, user FN…
3	^ (exponentiation)
4	unary –, NOT
5	* / % DIV
6	+ –
7	Relational: = # <> < <= > >= ?
8	AND
9	OR, EXOR

```
7 DIV 2 → 3  ·  7 \ 2 → 3          integer division (\ is the same operator)
MOD(7,2) → 1  ·  RMD(–7,2) → –1    modulo (sign of divisor) · remainder (sign of dividend)
50 % 200 → 100                     x % y = x percent of y
A=B  vs  IF A=B                    = assigns when the line can be read either way
X # Y  ·  X <> Y                   not equal (both forms)
X ? Y                              unordered: 1 if either operand is NaN
NOT A AND B → (NOT A) AND B        logical ops treat non-zero as true, return 1/0
"ABC" < "ABD" → 1                  strings compare by character code
```

Same-level operators evaluate left to right. EXOR: exactly one true. Relational operators return 1 (true) or 0 (false).

Number Display & Rounding

```
STD                                minimum digits for full 12-digit accuracy (default)
FIX 2  ·  SCI 4  ·  ENG 3          fixed decimals · scientific · engineering (0–11 digits)
OPTION ROUND NEAR|ZERO|POS|NEG     rounding direction for inexact results (default NEAR)
RES                                value of the last expression evaluated
EPS → 1E-499  ·  MAXREAL → 9.99999999999E499   smallest / largest positive real
MINREAL → 1E-499  ·  PI  ·  INF  ·  NAN   constants (INF/NAN with DEFAULT EXTEND)
RND  ·  RANDOMIZE [seed]           random 0≤r<1 · reseed (no arg: from clock)
IP(–3.7)→–3  FP(–3.7)→–.7  INT(–3.7)→–4   integer part · fraction · greatest integer ≤ x
FLOOR = INT  ·  CEIL(3.2)→4  ·  SGN  ·  ABS   rounding family
RED(x,y) → x–y*n, n=nearest int to x/y   reduce (like MOD but rounds)
```

Numbers ≥ 1E12 in magnitude are always shown in exponential form. FIX/SCI/ENG digits are readable in flags –17…–20; format in flags –13/–14; rounding in flags –11/–12. Display formats do not change stored values (12-digit internal precision).

Math Exceptions & IEEE

EXCEPTION	FLAG	NAME	EXAMPLES
Invalid operation	–8	IVL	ACOS(2), LOG(-23), (-14)^(-1/3)
Division by zero	–7	DVZ	187/0, TAN(90)
Overflow	–6	OVF	FACT(254), 10*1E499
Underflow	–5	UNF	EXP(-1149), 1/3E499
Inexact result	–4	INX	1/3, 1+1E-50

```
DEFAULT ON                         (reset state) DVZ/OVF/UNF → warning + default value; IVL stops
DEFAULT OFF                        any exception except INX is an error (use ON ERROR)
DEFAULT EXTEND                     IEEE defaults: Inf, –Inf, NaN; math continues
TRAP(DVZ,2)  ·  TRAP(–7)           set trap for a flag: 0 error · 1 default value · 2 IEEE value; no 2nd arg → read
CFLAG MATH  ·  CFLAG –8            clear all / one math-exception flag
CLASS(X)                           category of X: 1 zero, 2 denormalised, 3 normalised… ±sign; NaN/Inf classes
X ? Y  ·  INF  ·  NAN              unordered test · IEEE constants (with DEFAULT EXTEND)
```

ERRN	WARNING (DEFAULT ON)	DEFAULT VALUE
1 UNF	underflow	0
2 OVF	overflow	±99999 INTEGER · ±9.9999E499 SHORT · ±9.99999999999E499 REAL
3 DVZ	EXPONENT(0)	–9.99999999999E499
4 DVZ	TAN of odd multiple of 90°	±9.99999999999E499
5 DVZ	0 ^ negative	±9.99999999999E499
6	0 ^ 0	1
12 DVZ	LN(0)	–9.99999999999E499

Branching, Loops & Conditionals

```
GOTO 100  ·  GOTO DONE             line or label
GOSUB 500 … RETURN  ·  POP         subroutine; POP discards one pending RETURN
ON N GOTO 10,20,LAB  ·  ON N GOSUB …   N rounded → 1st, 2nd… target
ON N RESTORE 100,200               sets DATA pointer by N
IF X<0 THEN X=–X @ Y=1 ELSE Y=0    statements after THEN/ELSE run only on that branch
IF A THEN 100                      THEN line-number = GOTO
FOR I=1 TO 10 STEP 2 … NEXT I      loop; NEXT var optional; body skipped if start beyond limit
WAIT 2.5  ·  PAUSE  ·  STOP  ·  END  ·  END ALL   delay · suspend (CONT resumes) · halt · end · end all subprograms too
```

```
ON TIMER #1,30 GOTO 900   ·   OFF TIMER #1      3 timers, 0.1 – 134 217 727 s; retriggers each interval
ON TIMER #2,15 GOSUB 100                        RETURN re-arms; timers can wake a switched-off HP-71
BYE  ·  OFF                                     turn off (a program with a timer resumes on wake-up)
```

GOTO/GOSUB cannot branch into a subprogram or user function. Timers are local to the program/subprogram that set them. Nested GOSUBs limited only by memory.

Subprograms & User Functions

```
CALL AREA(R, A)          ! in current file      call subprogram; parameters by reference
CALL AREA((R), A) IN GEOM                       (R) passes by value; IN file = other file
SUB AREA(X, Y) @ Y=PI*X*X @ END SUB             subprogram; own variables, flags & files shared
SUB SHOW(A(,), N$) … END SUB                    array (,) and string parameters
DEF FNC(F)=(F-32)*5/9                           single-line numeric function
DEF FNR$(S$)=S$[LEN(S$)]&S$[1,LEN(S$)-1]        string function
DEF FNF(N) @ IF N<2 THEN FNF=1 ELSE FNF=N*FNF(N-1) @ END DEF   multi-line: assign to name, END DEF returns
DISP FNC(212)  ·  X=FNF(5)                      use like built-ins (single-line ones also in CALC mode)
```

- SUB…END SUB blocks live in program files (after the main program or in a file of their own). CALL name searches the current file, then all files. RUN starts at the first line, so a file may be all subprograms.
- Subprogram variables are local; only parameters, files, flags, and DATA are shared. END SUB / END returns to the caller.
- Recursion allowed (memory limited). END ALL terminates the whole chain of subprograms.

Errors & Debugging

```
ON ERROR GOTO 900  ·  ON ERROR GOSUB 900        trap run-time errors; GOSUB returns to the failing statement
OFF ERROR                                       restore normal error halt
ERRN  ·  ERRL  ·  ERRM$                         last error number · line · message text
TRACE FLOW  ·  TRACE VARS  ·  TRACE VARS A,B$  ·  TRACE OFF   show branches / assignments while running
CONT  ·  CONT 120  ·  RUN 120  ·  RUN FILE,LAB  resume · from a line · run from line/label
SFLAG -1                                        suppress most warning messages
BEEP OFF  ·  SFLAG -2                           silence beeper (flag -2)
```

MESSAGE FORM	MEANING
`ERR:Invalid Expr`	syntax error from the keyboard
`ERR L30:String Ovfl`	run-time error at line 30 (program suspended)
`WRN L30:…`	warning; execution continues
`Line Too Long`, `Missing Param`, `Data Type`…	common messages — full list in the Reference Manual

g ERRM or `ERRM$` redisplays the last message. Errors set ERRN/ERRL; a suspended program shows SUSP — fix and CONT, or SST through it. Math-exception warnings 1–12 are listed in "Math Exceptions".

Flags

```
SFLAG 3  ·  CFLAG 3  ·  FLAG(3)                 set · clear · test (1/0)
IF FLAG(-10) THEN …                             system flags are negative
CFLAG ALL  ·  CFLAG MATH                        clear all user flags 0–63 · clear -4…-8
SFLAG -25  ·  RESET                             loud beeper · reset system flags to defaults
```

FLAG	MEANING (SET = …)
0 – 63	User flags; 0–4 have annunciators
–1	Suppress warning messages
–2 / –25	Beeper off / loud
–3	Continuous on (no 10-min auto-off)
–4…–8	INX UNF OVF DVZ IVL math exceptions
–9	User keyboard active
–10	Radians (RAD annunciator)
–11,–12	Rounding: 00 NEAR · 01 ZERO · 10 POS · 11 NEG
–13,–14	Format: 00 STD · 10 FIX · 01 SCI · 11 ENG
–15	Lowercase keyboard
–16	OPTION BASE 1
–17…–20	Digits shown (binary, –17 = bit 0)
–26	BASIC prompt suppressed
–46	EXACT executed (read-only)
–57 –60…–64	AC · alarm · BAT · PRGM · SUSP · CALC annunciators (read-only)

–1…–32 may be set/cleared/tested; –33…–64 test only. All flags are global (visible in subprograms).

Input & the Keyboard

```
INPUT A, B$                                     prompts "?"; items separated by commas; expressions allowed
INPUT "NAME, AGE: "; N$, A                      custom prompt (quoted string; can't be overwritten)
INPUT "N? ","10"; N                             default string appears in the display for editing
LINPUT "TEXT? "; L$                             whole line into one string (commas, quotes kept)
K$=KEY$                                         oldest key from the 15-key buffer ("" if none); same names as DEF KEY
KEYDOWN  ·  KEYDOWN("Q")  ·  KEYDOWN("#43")     any key down? · specific key (unshifted only)
PUT "RUN"&CHR$(13)  ·  PUT "#43"                push keystrokes into the key buffer
DISP$                                           readable characters currently in the display (max 96)
```

USER KEYS

```
DEF KEY "fS","DISP TIME$";                      typing aid (";") — string only displayed
DEF KEY "gC","RUN CLOCK":                       execute-only (":") — runs directly, display ignored
DEF KEY "A","STD @ FIX 2"                       display-then-execute (no suffix)
DEF KEY "#43"  ·  DEF KEY "K"                   key by number (1–56, f: 57–112, g: 113–168) · cancel
FETCH KEY "gC"  ·  KEYDEF$("gC")                edit definition · read assigned string
USER  ·  USER ON  ·  USER OFF                   toggle / set the User keyboard (flag –9)
LIST KEYS  ·  PURGE KEYS  ·  MERGE KEYFILE      definitions live in the "keys" file
COPY KEYS TO K1  ·  COPY K1 TO KEYS             save / restore key sets
```

Output & Formatting

```
DISP A; B$, C                          ";" no space (numbers get sign/space), "," next 21-col field
DISP "X=";X;  ·  PRINT TAB(10);A       trailing ; suppresses end-of-line · TAB
PRINT …                                to printer (or display when no printer is assigned)
DISP USING "DDD.DD,4X,5A"; N, S$       format string inline
PRINT USING 100; A, B  ·  100 IMAGE 2(8X,6D)/
F$="SDDD.DD" @ DISP USING F$; V        string variable format
DELAY 0.5,0.1  ·  DELAY 8              line rate (s), scroll rate; ≥ 8 = infinite (wait)
WIDTH 22  ·  PWIDTH 80  ·  ENDLINE CHR$(13)&CHR$(10)   line lengths 0–96 · end-of-line string (≤3 chars)
WINDOW 5,18  ·  WINDOW 1               protect display columns outside 5–18 · unprotect
CONTRAST 9  ·  BEEP 440,.5  ·  BEEP    0–15 · Hz, seconds (default 500 Hz, .25 s)
GDISP D$  ·  D$=GDISP$                 132-column dot pattern out / in (1 char per column)
CHARSET CHR$(64)&…  ·  CHARSET$        define chars 128… (6 bytes per char) · read set
```

IMAGE FIELD SPECIFIERS

SPEC	MEANING
D	digit (leading zeros as blanks)
Z	digit with leading zeros
*	digit, leading asterisks
S	sign + or –
M	– or blank
.	decimal point
R	comma as radix
C	digit separator ,
E	exponent field
K	compact (like STD, no padding)
A	character
X	blank
"text"	literal
/	end of line
#	suppress end of line
B	byte: char with the given code
n(…)	repeat group n times
,	field separator

Symbol list per the Reference Manual's IMAGE entry (the Owner's Manual defers to it). A format string is reused when the list has more items than fields. Numeric field ↔ numeric item, else error. Escape sequences (ESC = CHR$(27)): `ESC<` cursor off · `ESC>` on · `ESC E` clear · `ESC N/R` insert/replace · `ESC %cn` cursor to column CODE(c) · `ESC C/D` right/left · `ESC K` clear to end · `ESC O` delete char. Text written while the cursor is off is a protected field.

Files & Memory

FILE TYPE	WHAT
BASIC	program (EDIT/RUN)
TEXT	LIF1 text records (TRANSFORM, HP-IL transfer)
DATA	random/sequential records (CREATE DATA)
SDATA	8-byte numbers, HP-41 compatible
KEY	key definitions ("keys" = current)
BIN / LEX	binary program / language extension (keywords)

```
EDIT PROG  ·  EDIT  ·  NAME NEWNAME    make current (create if new) · edit workfile · name it
CAT  ·  CAT ALL  ·  CAT PROG  ·  CAT :PORT(1)  ·  CAT$(2)   catalog: current file · all · one · device · nth entry
COPY A TO B  ·  COPY A TO :PORT(0)  ·  COPY :CARD  ·  COPY A TO CARD   duplicate (dest. must not exist)
RENAME A TO B  ·  PURGE A  ·  PURGE KEYS   rename · delete
MERGE SUBS  ·  MERGE SUBS,100,200      append lines of another BASIC file (line range) into current
SECURE F  ·  UNSECURE F  ·  PRIVATE F  no modify/purge · undo · irreversible: run only, no list/copy/edit
PROTECT / UNPROTECT                    card write-protection (HP 82400A)
TRANSFORM PROG INTO TEXT PTXT  ·  TRANSFORM PTXT INTO BASIC   program ↔ text file
MEM  ·  MEM(0)  ·  SHOW PORT  ·  SHOW PORT 0   free bytes (main RAM / port) · port sizes & type (1 RAM 2 ROM)
FREE PORT(0.02)  ·  CLAIM PORT(0)      make 4K unit independent RAM (survives INIT: 3) · reclaim (clears it)
ADDR$("PROG")  ·  PEEK$("2F400",8)  ·  POKE "2F400","0F"   file address · read/write nibbles as hex strings
```

Devices: `:MAIN`, `:PORT` (all ports), `:PORT(n)` n = 0–5 (5 = card reader port), `:CARD`, `:PCRD` (private card). Search order without a device: main RAM, then ports 0,1,2… Port 0 = internal 16 K in four 4 K units (0, .01, .02, .03). CAT columns: NAME S TYPE LEN DATE TIME PORT (S: P private, S secure).

Data: DATA/READ & Data Files

```
100 DATA 1, 2.5, "text", ABC           constants in program lines (unquoted strings OK)
READ A, B, S$  ·  RESTORE  ·  RESTORE 200   sequential read · reset pointer (to a line)
CREATE DATA DF, 50, 32                 DATA file: 50 records of 32 bytes (defaults: 1 record, 256 bytes)
CREATE TEXT TF  ·  CREATE SDATA SF, 100   text file · SDATA (100 8-byte numbers)
ASSIGN #1 TO DF  ·  ASSIGN #1 TO DF:PORT(1)  ·  ASSIGN #1 TO *   open channel 1–255 · close
PRINT #1; A, B$, M(,)                  sequential write; whole array with (,)
PRINT #1,5; X  ·  READ #1,5; X         random access: record 5
READ #1; A, B$  ·  RESTORE #1  ·  RESTORE #1,3   sequential read · rewind · to record 3
IF … THEN … ON ERROR GOTO EOF          end of file gives an error → trap it
CALL SUB1(#1)  ·  SUB SUB1(#C)         pass an open channel to a subprogram
```

DATA files store numbers (8 bytes) and strings (length+text) in records; an item that doesn't fit a record spills to the next in sequential mode but errors in random mode. TEXT files are read/written a line (record) at a time. Closing: ASSIGN # TO * or program END.

Editing & Running Programs

```
EDIT PROG                              create/select current file — the PRGM lines you type go here
10 DISP "HELLO"                        type line number + statements, END LINE
AUTO  ·  AUTO 100,5                    auto line numbers (start, step; default 10,10); ATTN ends
FETCH 30  ·  FETCH LAB  ·  ▲ ▼         bring a line to the display for editing; step through
LIST  ·  LIST 100,200  ·  PLIST        list (display, paced by DELAY) · to printer
DELETE 30  ·  DELETE 30,90  ·  DELETE ALL   remove lines
```

RENUMBER · RENUMBER 100,10 · RENUMBER 1000,10,500,600	renumber (new start, step, old first, old last)
RUN · RUN PROG · RUN PROG,LAB · RUN 200	run current / named file, from a line or label
CALL PROG · CHAIN PROG:CARD	run as subprogram · purge current, load & run next
CONT · f-SST · ATTN	continue · single-step · suspend
COPY TO PROG · DELETE ALL	save the workfile under a name · clear workfile
SECURE PROG · PRIVATE PROG	protect before distributing

workfile is the unnamed default file (NAME gives it a name). Only one current file; RUN filename makes it current. BIN files run with RUN/CALL but can't be listed; LEX files add keywords and just need to be in memory. TRACE FLOW/VARS for debugging.

Clock, Calendar & Timers

SETDATE "85/03/07" · SETDATE 85066 · SETDATE "1985/03/07"	YY/MM/DD or YYDDD (day of year)
DATE → 85066 · DATE$ → "85/03/07"	numeric YYDDD · string
SETTIME "08:15:00" · SETTIME 8*3600+15*60	24-hour string or seconds since midnight
TIME → 29700 · TIME$ → "08:15:00"	seconds since midnight · string
ADJABS -3600 · ADJABS "-01:00:00"	shift clock, no speed correction (time zones)
ADJUST 43 · ADJUST "00:01:00"	shift and accumulate a speed-correction (±100 h)
EXACT	mark "clock is exact now"; applies accumulated correction (flag -46)
AF(0.5) · AF	set / read speed adjustment factor directly
RESET CLOCK	clear speed correction, keep time
ON TIMER #1,60 GOSUB TICK	see Branching — 3 timers, wake from OFF

Set the clock with SETTIME after a memory reset, then EXACT; later use SETTIME/ADJUST to correct drift and EXACT again to teach the crystal correction. Calendar range 0000–9999 (Gregorian from 1582/1752).

Statistics

STAT S(3) · STAT S	create/select current statistical array for 3 variables (max 15) · reselect
CLSTAT	clear (zero) the current statistical array
ADD 6.4, 3.85, 3.6 · DROP 6.4, 3.85, 3.6	add / remove a data point (one value per variable)
TOTAL(2) · TOTAL(0) · MEAN(1) · SDEV(3)	sum of var 2 · number of points · mean · sample std. dev.
CORR(1,2)	sample correlation of variables 1 and 2
LR 2,1 · LR 2,1,A,B	linear regression of var 2 on var 1; intercept→A, slope→B
PREDV(X)	predicted dependent value at X for the current LR model

The array holds only summary statistics (sums, sums of squares, cross products), always base 0 and REAL. Default variable number is 1. Fit other curves by transforming data (e.g. ADD X, LN(Y) for exponential fits).

Idioms

10 DESTROY ALL @ OPTION BASE 1 @ STD	clean start of a program
20 DELAY 0,0	fastest display, no waiting
30 ON ERROR GOSUB 900	central error handler
40 INPUT "N? ",STR$(N); N	re-prompt with previous value as default
50 K$=KEY$ @ IF K$="" THEN 50	wait for a key
60 IF UPRC$(K$)="Q" THEN END	quit on Q (ATTN itself suspends the program)
70 DISP "Working…"; @ … @ DISP	progress on one line
80 T=TIME @ … @ DISP TIME-T;"s"	timing (seconds)
90 A$=DTH$(HTD(A$)+1)	hex arithmetic
100 IF FLAG(-10) THEN RADIANS ELSE DEGREES	preserve angle mode
110 X=RND*(H-L)+L	random real in [L,H)
120 SFLAG -1 @ … @ CFLAG -1	quiet section
DEF KEY "gR","RUN":	one-key run from the User keyboard
STARTUP "RUN MENU"	command executed at every power-on
LOCK "SECRET"	lock the keyboard until the password is typed

Keyword Reference

249 entries · 13 categories · hover for syntax & details

Program Entry & Editing 14

FILES

EDIT	Select / create program file
NAME	Name the workfile
TRANSFORM	BASIC ↔ TEXT
PRIVATE	Make file run-only
SECURE	Protect from change
UNSECURE	Remove SECURE

LINES

AUTO	Automatic line numbers
DELETE	Delete lines
FETCH	Bring line to display
LIST	List program
PLIST	List to printer
RENUMBER	Renumber lines
REM (!)	Remark
@	Statement separator

Program Execution & Control 34

RUN

RUN	Run a program
CONT	Continue
CALL	Call subprogram
CHAIN	Chain program
BYE	Power off
OFF	Power off

BRANCH

GOTO	Branch
GOSUB	Subroutine call
RETURN	Return
POP	Drop return address
ON…GOTO	Computed GOTO
ON…GOSUB	Computed GOSUB
ON…RESTORE	Computed RESTORE
IF…THEN…ELSE	Conditional
FOR…NEXT	Loop

STOP & WAIT

END	End program
STOP	Halt
PAUSE	Suspend
WAIT	Delay

SUBPROGRAMS & FUNCTIONS

SUB	Begin subprogram
END SUB	End subprogram
DEF FN	Define user function
END DEF	End multi-line function
FN	Call user function

TIMERS

ON TIMER #	Timer branch
OFF TIMER	Cancel timer

ERRORS

ON ERROR GOTO	Error trap
ON ERROR GOSUB	Error trap (subrouti…
OFF ERROR	Cancel trap
ERRN	Error number
ERRL	Error line
ERRM$	Error message
DEFAULT	Exception handling
TRACE	Trace execution

Storage & Memory 15

VARIABLES

DIM	Dimension
REAL	Declare REAL
SHORT	Declare SHORT
INTEGER	Declare INTEGER
OPTION BASE	Array base
DESTROY	Free variables
LET	Assign
STAT	Statistical array

PORTS

MEM	Free memory
SHOW PORT	Port info
FREE PORT	Make independent RAM
CLAIM PORT	Reclaim RAM
ADDR$	File address
PEEK$	Read memory
POKE	Write memory

Operators 17

ARITHMETIC

+	Add
−	Subtract / negate
*	Multiply
/	Divide
^	Power
DIV (\)	Integer divide
%	Percent

RELATIONAL

=	Equal
# / <>	Not equal
< <=	Less
> >=	Greater
?	Unordered

LOGICAL

AND	Both true
OR	Either true
EXOR	One true
NOT	Negate
&	Concatenate

General Math 42

PARTS & ROUNDING

ABS	Absolute value
SGN	Sign
IP	Integer part
FP	Fractional part
INT	Floor
FLOOR	Floor
CEIL	Ceiling
MAX	Maximum
MIN	Minimum
MOD	Modulo
RMD	Remainder
RED	Reduce

POWERS & ROOTS

SQR	Square root
SQRT	Square root
FACT	Factorial
EXP	e^x
EXPM1	e^x – 1
LN	Natural log
LOG	Natural log
LOGP1	ln(1+x)
LGT	Common log
LOG10	Common log
EXPONENT	Exponent of x

RANDOM & CONSTANTS

RND	Random
RANDOMIZE	Seed
PI	π
RES	Last result
EPS	Smallest positive
MINREAL	Smallest positive
MAXREAL	Largest real
INF	Infinity
NAN	Not a number

CONVERSIONS

DTH$	Decimal → hex
HTD	Hex → decimal
CLASS	Number class

EXCEPTIONS

IVL	Invalid flag
DVZ	Divide-by-zero flag
OVF	Overflow flag
UNF	Underflow flag
INX	Inexact flag
TRAP	Exception trap
OPTION ROUND	Rounding mode

Trigonometry 12

FUNCTIONS

SIN	Sine
COS	Cosine
TAN	Tangent
ASIN / ASN	Arcsine
ACOS / ACS	Arccosine
ATAN / ATN	Arctangent
ANGLE	Angle of (x,y)
DEG	Rad → deg
RAD	Deg → rad

MODE

DEGREES	Degree mode
RADIANS	Radian mode
OPTION ANGLE	Angle mode

Statistics 10

DATA

STAT	Statistical array
CLSTAT	Clear stats
ADD	Add data point
DROP	Remove data point

RESULTS

TOTAL	Sum
MEAN	Mean
SDEV	Std. deviation
CORR	Correlation
LR	Linear regression
PREDV	Predicted value

Strings 10

FUNCTIONS

LEN	Length
POS	Position
VAL	String → number
STR$	Number → string
NUM	Character code
CHR$	Code → character
UPRC$	Uppercase
VER$	Version
&	Concatenate
s$[a,b]	Substring

Input & Output 35

DISPLAY & PRINT

DISP	Display
PRINT	Print
DISP USING	Formatted display
PRINT USING	Formatted print
IMAGE	Format line
TAB	Tab
WIDTH	Display line width
PWIDTH	Printer line width
ENDLINE	End-of-line string
DELAY	Display timing
CONTRAST	LCD contrast
WINDOW	Protect columns
DISP$	Read display
GDISP	Dot graphics
GDISP$	Read dots
BEEP	Tone
LC	Lowercase

KEYBOARD INPUT

INPUT	Input values
LINPUT	Input a line
KEY$	Key from buffer
KEYDOWN	Key pressed?
PUT	Push keys
USER	User keyboard

DATA STATEMENTS & FILES

DATA	Data constants
READ	Read DATA
RESTORE	Reset DATA pointer
CREATE	Create data file
ASSIGN #	Open / close channel
PRINT #	Write to file
READ #	Read from file
RESTORE #	Reposition file

FORMATS

STD	Standard format
FIX	Fixed format
SCI	Scientific
ENG	Engineering

File Management 17

CATALOG & COPY

CAT	Catalog
CAT$	Catalog entry
COPY	Copy file
RENAME	Rename
PURGE	Delete file
MERGE	Merge lines

PROTECTION

SECURE	Prevent change
UNSECURE	Allow change
PRIVATE	Run-only forever
PROTECT	Card write-protect
UNPROTECT	Card write-enable

MEMORY

MEM	Free bytes
SHOW PORT	Port sizes
FREE PORT	Independent RAM
CLAIM PORT	Reclaim
ADDR$	File address
TRANSFORM	Change type

Time & Date 11

READ

DATE	Date number
DATE$	Date string
TIME	Time number
TIME$	Time string

SET & ADJUST

SETDATE	Set date
SETTIME	Set time
ADJABS	Shift clock
ADJUST	Shift + correct
EXACT	Mark exact
AF	Adjustment factor
RESET CLOCK	Clear correction

Customization & Keyboard 20

USER KEYS

DEF KEY	Define key
KEY	Key name forms
FETCH KEY	Edit definition
KEYDEF$	Read definition
KEY$	Buffered key
KEYDOWN	Key pressed
PUT	Push keys
USER	User keyboard

DISPLAY & SYSTEM

CHARSET	Alternate chars
CHARSET$	Read char set
CONTRAST	Contrast
DELAY	Output timing
WINDOW	Protected fields
LC	Lowercase
LOCK	Lock keyboard
STARTUP	Power-on command
DTH$ / HTD	Hex conversion
FIX / IMAGE	see Output
PEEK$ / POKE	Memory access
ADDR$	File address

System Settings & Flags 12

FLAGS

SFLAG	Set flag
CFLAG	Clear flag
FLAG	Test flag
RESET	Reset settings

MODES

DEFAULT	Exceptions
TRAP	Exception trap
DEGREES / RADIANS	Angle mode
OPTION ANGLE	Angle mode
OPTION BASE	Array base
OPTION ROUND	Rounding
DELAY	Display timing
IVL DVZ OVF UNF INX	Exception fl…

Emu71 Emulator Notes host-side only — not part of the HP-71B

Emu71 — running the HP-71B on the Mac (Wine)

Install: `~/ide/HP71B/emu71/` — `./emu71.sh` starts `run/EMU71.exe` under Wine. First start: pick a KML skin (e.g. *Christoph's Real HP71B for 1024x768*) → OK. Emu71 emulates the real ROM (HP71B.ROM = 1BBBB, checksum-verified), so everything on these pages applies unchanged; the emulator adds only its Windows menu.

EMU71 MENU	USE
File → New / Open / Save (.e71)	Emulator state = the calculator's whole memory. Save before quitting (or enable Settings → "Automatically Save Files On Exit").
Edit → Port Configuration…	Plug modules: RAM (4/32 K…), ROMs from `run/` (MATHROM.BIN, Hpilrom.bin, FORTHROM.BIN/HRDFORTH.BIN), HP-IL interface. Then `SHOW PORT` / `CAT :PORT(n)` on the 71.
Edit → Copy Stack / Paste Stack	Copy the display text to the clipboard · paste clipboard text into the command line (BASIC mode). Handy for single lines.
Edit → Reset Calculator · Backup	CPU reset (then INIT: 1/2/3 as on the real machine) · snapshot / restore state.
File → Settings → Authentic Calculator Speed	Uncheck for maximum speed (needed for fast macro replay).
Tools → Macro → Record / Play / Stop / Settings	Keyboard macro recorder — the way to "type" a whole program file into the 71.
View → Change KML Script	Other skins incl. module-overlay backgrounds (kml-overlays).

EDITING PROGRAMS OFFLINE AND LOADING THEM

Emu71 has no "load text file" command; instead a text listing is converted to a keystroke macro that Emu71 replays into the calculator (MK71MAC package, `emu71/mk71mac/`).

```
1. Write the program as plain ASCII in your editor: prog.bas   one program line per text line, LF line ends; ≤ 96 chars/line
   EDIT MYPROG                                                 first line: make/select the target file (or NAME it)
   10 DISP "HELLO" @ BEEP                                      normal HP-71 lines; keywords may be lower case
   RUN                                                         optionally finish with a command
2. ~/ide/HP71B/emu71/bas2mac.sh prog.bas                       → prog.mac (runs ASC71MAC.EXE via Wine; UTF-8 handled)
3. Emu71: Tools → Macro → Play…  choose prog.mac               keys are typed into the running 71 (BASIC prompt, no program running)
4. CAT MYPROG · LIST · RUN                                     check; File → Save to keep it in the .e71 state
```

- Every LF in the text presses END LINE; CR is ignored. Character set: printable ASCII 32–126 except `\ | ~` and DEL. Use `DIV` instead of `\`.
- Replay speed: Tools → Macro → Settings → Manual + slider; if keystrokes get lost, slow it down or untick "Authentic Calculator Speed".
- Type errors on the 71 beep and leave the bad line un-entered — the rest continues. Check with `LIST`; the 71 keeps only what parsed.
- Reverse direction: `LIST` and Edit → Copy Stack copies one display line at a time; for whole listings use PLIST to a virtual HP-IL printer (Emu71 virtual HP-IL over TCP/IP + *ILPer*/*pyILPER*) or TRANSFORM to TEXT and copy to a virtual LIF disk.
- Larger workflow: virtual HP-IL (HP-IL ROM in a port + pyILPER on the Mac) gives a mass-storage drive — `COPY MYPROG TO :TAPE`, `COPY :TAPE`… exactly as with a real HP 82161A.

Sources: Emu71 v1.19 manual (`emu71/emu71-bin/Emu71.htm`), MK71MAC.TXT. Nothing in this box applies to a physical HP-71B.