Honeywell

SERIES 60 (LEVEL 6)

GCOS/BES2 BASIC Interpreter

The Level 6 GCOS/BES2 BASIC Interpreter is a simple, easy-to-learn, all-purpose programming language.

The BASIC Interpreter provides an *interactive/conversational* environment in which a user can compose, edit, debug, and execute programs written in the Level 6 BASIC language. In addition, BES2 BASIC, with its program library and data file facilities, provides for program execution in a *nonconversational,* "production" mode of operation, which includes loading previously prepared programs from a disk library file and reading and writing sequential data files (also stored on disk) during the execution of the programs.

FEATURES

- Complete source program maintenance
- Data file processing language features
- Data file interchange with Level 6 FORTRAN programs
- Multiple statements per line
- Immediate statement execution for interactive programming and debugging
- Multidimensional arrays
- Unrestricted subscript expressions

ADVANTAGES

- Mature, standardized language
- Easy-to-learn, easy-to-use
- Fast debugging
- Useful for general problem solving and elementary production applications
- Smooth, simple interface with other BES2 facilities

MODES OF OPERATION

The BES2 BASIC Interpreter can be used in three generally distinct modes of operation:

- Interactive problem solving
- Production program building/checkout
- Production program execution

Interactive problem solving is normally a short-term activity in which the user composes a small, "one-shot" program to solve a current, often urgent problem. In accomplishing this, the interactive, conversational facilities of BES2 BASIC are heavily used: composing, editing, and storing program statement lines in memory, trial execution of sequences of these lines, and immediate execution of individual statements for experimental and debugging purposes.

Production program building/checkout is a longer-term activity involving the development of a larger, permanent program to be executed usually at regular intervals in the future. In this mode of operation, a disk library is used for external storage and retrieval of the source program being built, and sample data files can be created to test the proper execution of the program logic. The facilities described for interactive problem solving are, of course, also available in this mode.

Production program execution requires only minimal user/operator involvement and interaction. After loading the BES2 BASIC Interpreter and requesting the desired program from the disk/diskette library file (via the OLD command), the operator begins execution by typing "RUN." During execution, a program performs reads from and writes to data files on disk/diskette volumes, or accepts data input from the operator console device.

PROGRAM EDITING FACILITIES

The BES2 BASIC Interpreter contains a full range of elementary editing functions to create and change a stored source program. To add or insert a new line into an already existing stored program, the user types the appropriate new line number followed by one or more BASIC statements separated by backslashes (\). To replace an existing line, the user types its line number, followed by the new statement(s). To delete an existing line, the user types its line number, followed by a carriage return. To correct errors while typing a line,

the user types the standard BES2 line deletion and character deletion characters.

BASIC PROGRAMMING EXAMPLES

The following four examples utilize different techniques to solve for one of the two roots of the equation:

$$X^2 + 2X - 4 = 0$$

The root is:

$$\text{ROOT} = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

where A = 1, B = 2, C = –4

The BES2 BASIC Interpreter supplied the underscored responses:

1. Use of READ and DATA Statements for Variables

```
? NEW

READY
? 10 READ A,B,C
? 20 LET X=(–B+(B ^ 2–4*A*C) ^ 0.5)/2*A)
? 30 DATA 1,2,–4
? 40 PRINT "ROOT IS",X
? 50 END
? RUN
ROOT IS          1.23607

50 EXIT
READY
```

2. Use of Data Constants

```
? NEW

READY
? 10 LET X=(–2+(2 ^ 2–4*1*(–4)) ^ .5)/(2*1)
? 20 PRINT "ROOT IS",X
? 30 END
? RUN
ROOT IS          1.23607

30 EXIT
READY
```

3. Use of INPUT Statement

```
? NEW

READY
? 10 INPUT A,B,C
? 20 LET X=(–B+(B ^ 2–4*A*C) ^ .5)/(2*A)
? 40 PRINT "ROOT IS",X
? 50 END
? RUN
? 1,2,-4
ROOT IS          1.23607

50 EXIT
READY
```

4. Use of Arithmetic Calculator Loop

```
? NEW

READY
? 10 INPUT I\\PRINT I\\GO TO 10
? RUN
? (–2+(2 ^ 2–4*1*(–4)) ^ .5)/(2*1)
   1.23607
?
```

LANGUAGE SUMMARY

A BES2 BASIC program consists of a set of statements, normally terminated by an END statement. Each line (containing one or more statements) in a stored program in memory must be numbered. Unnumbered statements are executed immediately. Table 1 lists the major constituents of the BES2 BASIC language. The angular and square brackets in the table represent variables and optional elements, respectively.

LET and GOTO are the assignment and control transfer statements, respectively. GOSUB is a subroutine call, and the RETURN statement defines the return point from the subroutine which was called. IF ... THEN allows a single relational operator between expressions, and control passes to the statement number following the THEN if the relation is true. NEXT is used to terminate the range of the FOR statement and must use exactly the same variable as in the FOR statement. If the third expression in a FOR statement is omitted, it is assumed to be 1. The expression in the ON statement is evaluated and truncated to an integer. For expression =1, control is transferred to the first statement number in the list; for expression =2, control is transferred to the second statement number in the list, etc.

READ assigns to the listed variables the values obtained from a DATA statement. The latter is used to specify all the values needed for the variables. For output, the user can specify variable names or literals; the literals are enclosed within quotation marks. Thus, if X is 625, the statement

```
PRINT "THE SQUARE ROOT OF" X,
"IS" SQR (X)
```

causes the following to be printed:

```
THE SQUARE ROOT OF 625 IS 25.
```

For normal printing purposes, the output line is divided into five zones of 13 spaces each. The user can change the width of these zones, however, through the use of commas and semicolons. A PRINT statement without anything following signals a new line.

The program terminates with the END statement; STOP returns BASIC to the command mode.

DIM is used to specify numeric variable subscripts whose values are not equal to 10, and to specify the size of string variables when the default size is not desired. DATA specifies numeric and string values to be accessed by READ statements.

RESTORE returns the current pointer to the first value of the first DATA statement in the program. An INPUT statement types a question mark and the program waits for the user to type in the data items requested in the INPUT list.

Functions are defined by the DEF statement; the function name consists of the letters FN followed by a single letter. Any expression which fits on one line can be used to define a function. The expression may even include other functions.

RANDOMIZE gets a new "seed" from the real-time clock for use by the RND random number function. OPTION STRINGSIZE specifies a default size for all character strings, when the default desired is not the standard 18.

REM statements are nonexecutable and are used to enter comments and explanations in the program listing.

FILES specifies all data files which are to be processed in the program, and implicitly assigns a numeric designator to each, for use in the other data file statements.

SCRATCH# logically clears the specified data file, and opens it for output. WRITE# puts the specified data values out onto the data file. READ# brings in input items from the data file and assigns them to the variables listed. RESTORE# repositions the data pointer to the first item in the data file and opens it for input. IF END# specifies the processing path to be taken when an end-of-file condition is reached on the input data file.

Statements, control commands, built-in functions, and diagnostics are summarized in Tables 1–4. These and other language and programming considerations are described in detail in the *BES2 BASIC Reference Manual,* Order No. AU44.

SYSTEM REQUIREMENTS

Minimum equipment required:

- Level 6 central processor with 16K words of memory
- Dual diskette unit
- KSR teleprinter or compatible console

Optional equipment:

- 16K additional words of memory
- Additional dual diskette unit
- Cartridge disk units

Related software:

- Runs under the BES2 operating environment, with 16K words of memory

---

Specifications may change as design improvements are introduced.

Table 1. BASIC Language Summary

[LET] <variable> [,<variable>] ... = <expression>

IF <expression> <relation> <expression> { THEN / GOTO } <line number>

GOTO <line number>

GOSUB <line number>

RETURN

FOR <unsubscripted variable> = <expression> TO <expression> STEP <expression>

NEXT <unsubscripted variable>

ON <expression> GOTO <line number> , <line number>...

STOP

END

DATA <numeric/string constant> , <expression>...

READ <numeric/string variable> [,<numeric/string variable>] ...

RESTORE

INPUT <numeric/string variable> [,<numeric/string variable>] ...

PRINT <numeric/string expression> , <numeric/string constant>...

DIM <variable> (<integer> [,<integer>] ...)...

DEF FN<letter> (<unsubscripted variable>) = <expression>

RANDOMIZE

OPTION STRINGSIZE <integer>

REM <any string of characters>

FILES <file name> [,<file name>] ...

SCRATCH# <integer expression>

WRITE# <integer expression> , <numeric/string expression> , <numeric/string constant>...

READ# <integer expression> , <numeric/string variable> [,<numeric/string variable>] ...

RESTORE# <integer expression>

IF END# <integer expression> { THEN / GOTO } <line number>

Table 2. Control Commands

OLD <program name>	Loads named program from disk library into current program area in memory, and clears data area.	LIST <line number>	Lists current program beginning at line number specified.
NEW [<program name>]	Clears current program and data area in memory, and sets current program name as specified.	LIST <line number> , <line number>	Lists current program beginning at first line number specified and ending with last line number specified.
NAME [<program name>]	Changes current program name as specified.	SAVE [<program name>]	Writes current program on disk library, using program name specified.
CLEAR	Clears data area only in memory.	RESAVE [<program name>]	Replaces named program on the disk library with current program.
RUN [<line number>]	Starts execution of current program, beginning at lowest-numbered line or at line number specified.	QUIT	Exits from BASIC Interpreter environment, returning control to BES2 Command Processor.
LIST	Lists entire current program on teletypewriter.		

Table 3. Built-in BASIC Functions

Function Reference	Mathematical equivalent	Function Reference	Mathematical equivalent
SIN(X)	sin (x) The trigonometric sine of the argument x.	INT(X)	The largest integer less than or equal to the argument x. INT (+1.5) = 1 INT (−1.5) = −2
COS(X)	cos (x) The trigonometric cosine of the argument x.	SGN(X)	The sign of the argument x. X<0, SGN(X) = −1 X=0, SGN(x) = 0 X>0, SGN(X) = 1
TAN(X)	tan (x) The tangent of the argument x.	SQR(X)	$\sqrt{x}$, where $x \geq 0$ The positive square root of the argument x, where the argument must be greater than or equal to zero.
ATN(X)	$\tan^{-1}$ (x) The arctangent of the argument x.	RND	A random number with a value between 0 and 1.
EXP(X)	ex The exponential function.		
LOG(X)	$\log_e$ (x) The logarithm to the base e of the argument x.		
ABS(X)	x The absolute value or magnitude of the argument x.		

NOTE: The argument x is expressed in radians.

Table 4. Diagnostic Error Codes

AF	No data files attached but FILES statement found
AS	Array subscript out of bounds
C?	Unexpected character (c) found
CO	Close data output file problem
DA	Attempt to READ more data than available (in DATA statement or data file)
DF	Data file numeric designator error: not integer, not in range, or not defined
DL	Statement terminator error
DP	Two decimal points in number
DV	Dummy variable in DEF statement subscripted
DZ	Division by zero
FD	Invalid delimiter in FOR statement
FI	Data file input problem: output file only or hardware problem
FN	Characters FN misplaced in DEF statement
FO	Data file output problem: cannot open or hardware problem
GS	GOSUBs nested more than eight deep
IC	Incorrect condition in IF statement
ID	Illegal statement or variable name
IV	Illegal index variable in FOR statement
LG	Negative logarithmic function argument
LI	Library input file not attached or not present
LO	Library output file not attached or not present
MA	Mixed assignment: string assigned to or compared with numeric variable
MO	Memory overflow
M,	Missing or misplaced comma
M=	Missing or misplaced equals sign
M)	Missing or misplaced right parenthesis
M(	Missing or misplaced left parenthesis
NO	Numerical overflow
NU	Numerical underflow
NX	NEXT statement has no matching FOR
ON	Nonpositive expression in ON statement, or missing GOTO
OS	OPTION STRINGSIZE format error
PD	Illegal item delimiter in PRINT statement
RF	READ# format error: missing comma after file number
RT	RETURN statement found outside of subroutine
SF	SCRATCH# format error
SL	String length specified over 60 characters
SN	Statement number error (range 1 to 9999)
SO	String overflow
SQ	Negative square root function argument
TH	THEN or GOTO left out of IF statement
TX	Missing or misplaced quotation marks
UF	Undefined function
UM	Unitary minus error
US	Undefined statement number
UV	Undefined variable
WF	WRITE# format error

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

16371, 5876, Printed in U.S.A.

AW70, Rev. 0