BEM:BASIC — OS/3

Instruction Summary

Code Card

SPERRY UNIVAC

UA-0191 Rev. 1

The BEM programs described in this document are confidential information and proprietary products of the Sperry Univac Division.

This document contains the latest information available at the time of publication. However, Sperry Univac reserves the right to modify or revise its contents. To ensure that you have the most recent information, contact your local Sperry Univac representative.

SPERRY UNIVAC NEWSCOMP Newspaper Composition Program was used by Application Services in typesetting this publication.

Sperry Univac is a divison of Sperry Rand Corporation.

OS/3 BEM:BASIC SUMMARY

BEM COMMANDS

/DELETE (file-parameters) — Deletes an element from a library file.

/DISPLAY JOB — Displays information about currently active batch jobs.

/DISPLAY VOLUMES — Displays the names of any disk pack mounted at the main site.

/EXEC BASIC — Invokes the BASIC compiler; when loaded the following is displayed:

OS/3 BASIC READY (VER n.n) BEGIN

/FSTATUS library (password),volume
Displays the directory of the requested file.

/HELP — Displays additional information about the previous error message.

/INTR — Interrupts current command in execution.

/LOGOFF — Terminates a user's session and releases all work space to the system.

/LOGON id — Identifies the user to the system. Must always be the first command entered. The system will respond with a bulletin followed by:

USER LOGGED ON, BEM SYSTEM READY

/PAUSE question — Displays the user's question on the computer console and pauses the user until the operator's reply is available for display to the user.

/PRINT (file-parameters) — Prints a library element at the central site.

/PUNCH (file-parameters) — Punches a library element at the central site.

/RUN [jobname] — Initiates a job at the main site.

/SCREEN [**ROLL** | **NOROLL**] [**,COP** | **NOCOP**] [**,height × width**]

Defines the UNISCOPE terminal characteristics.

/STATUS RESOURCES — Displays the amount of memory and disk space available to BEM.

/STATUS TERM — Displays a list of terminals currently in use on the system.

/TYPE message	Displays the user's message on the computer operator's console.
/VTOC volume	Displays the names of files on a disk volume.

Language Elements

General

1. blanks

   The character blank, which may be used in constructing the BASIC programs, is designated in the syntax by the symbol Δ. Any spaces which appear in the syntax equations do not denote blanks in the BASIC language. Blanks are only significant in BASIC when they appear in a comment or in a string constant.

2. quote

   The character quote (") is used to delimit the beginning and end of a closed-string constant.

3. asterisk

   Exponentiation is specified by means of a pair of asterisks. An up-arrow (↑) is also permitted, where applicable.

Constants

decimal-number	A fraction followed by optional exponent field. Fraction: series of 1 to 15 digits containing optional decimal point preceding, following, or embedded in series of digits. Examples: 85 85. .85 85.6438 Exponent: indicates the power of 10 by which the fraction is to be multiplied and consists of the letter E followed by optional sign and one or two digits. Sign is + or —; if omitted, + is assumed. Examples: E5 E+14 E+8 E—04 E—2
closed-string	Quote followed by a series of 0 to 4095 string characters followed by a quote. Example: "ABZ154Δ84"

open-string	A character followed by a series of 0 to 4095 characters or blanks (△) or quotes ("), terminated by a comma or end of line. Example: ABZ154△84
line-number	Series of one to five digits, ranging from 1 to 99999 in value. Leading zeros are ignored. Each statement must have a unique line-number. Example: 01250

NOTES:

1. *decimal numbers*

   All decimal numbers are converted and stored internally in floating-point format. The exponent occupies seven bits and indicates the power to which the number 16 must be raised. The sign occupies one bit. In BASIC the mantissa occupies 24 bits, and contains a 6 digit hexadecimal number in normalized form. If the value of the fraction part of a decimal number, disregarding the decimal point, exceeds $2^{24} - 1$, the number is rounded and trailing digits are lost; for example:

   12.3456789

   is acceptable, but is (effectively) rounded to

   12.345679

   If the mantissa is nonzero, the magnitude of the floating-point number has the following range:

   $16^{-65} \leq M < 16^{63}$ *(approximately* $10^{-78} \leq M < 10^{75}$ *)*

   Overflow and underflow conditions for numeric constants are processed as errors.

2. *string constants*

   All string constants are stored in EBCDIC code. A 2-byte length field is prefixed to each string before it is stored; the value of the length byte is not included. If a given string constant contains more than 4095 characters, it is truncated at the right. Note that an open-string constant, as opposed to a closed-string constant, cannot begin with a quote and cannot contain a comma. Moreover, an open-string constant is permitted only as input to the READ, and INPUT statement.

   Within a closed-string constant, two consecutive quotes are interpreted as a single quote.

Variables

scalar variable	Defined as a numeric variable or a string variable.
numeric variable	A letter optionally followed by a single digit. Example: S S8
string variable	A letter followed by a dollar sign ($) or a letter and single digit followed by a dollar sign ($). Example: V$, B$, F7$, B0$
array variable	Defined as a numeric array variable or a string array variable.
numeric array variable	A letter followed by one or two subscript expressions enclosed in parentheses. Examples: M(2) P(8,92) X(A+B)
string array variable	A letter followed by a dollar sign ($) followed by one or two subscripts enclosed in parentheses. Examples: M$(2) C$(20) D$(A+B9,10)

NOTES:

1. *numeric variables*

   Numeric variables may only be assigned decimal numeric values. All such variables are initialized to zero (0).

2. *numeric array variables*

   Numeric array variables may only be assigned decimal numeric values. All such variables are initialized to zero (0).

3. *subscripts*

   A subscript may be defined using any arithmetic expression. During execution, the value used to locate the array element referenced is computed by taking the integer part of the subscript expressions. Rounding occurs for each subscript. If the subscript value is not within the bounds specified (or implied) for that dimension of the referenced array, then the user is given an error message and program execution terminates.

   Two-dimensional numeric arrays are stored in row-major order.

4. *string variables*

   String variables may only be assigned character string values. All such variables are initialized to the null string (zero length).

5. *string array variables*

   String array variables may only be assigned character string values. All elements of these string array variables are initialized to the null string (zero length).

   The rules for numeric array variables regarding bounds and subscript evaluation apply to string array variables as well.

Expressions

arithmetic-expression	Defined as a term optionally preceded by a minus (−) or plus (+); or an arithmetic expression plus (+) or minus (−) a term. Example: A**2*B − 3
term	A factor or a term multiplied(*) or divided (/) by a factor. Example: A**2*B
factor	A primary or a factor raised to a power (**) designated by a primary. Example: A**2
primary	A decimal number, numeric reference, function reference, or an arithmetic expression enclosed in parentheses. Example: 2,A,RND(X), (C − D)
string-expression	String-primary or a string-expression followed by an ampersand (&), denoting concatenation, followed by another string-expression. Example: "ABC" & F$
string-primary	A closed string, string reference, or string function reference. Example: A$, SEG$ (D$, 6, 8), "AB"

NOTES:

1. *The exponentiation operator (**) may be written where applicable as an up arrow (↑).*

2. *A**B**C is compiled as (A**B)**C.*

3. *Parentheses may be used to factor subexpressions.*

4. *The following are treated as errors:*

   - *Mixed mode expressions and any operations on string data*
   - *Division by zero*
   - *Zero to a negative power*

- *Overflow and underflow conditions existing during the evaluation of arithmetic expressions.*

5. *A negative number can only be raised to a nonzero positive integer number. The maximum value of this positive integer is 15. Any violation of this rule is treated as an error.*

6. *The concatenation operator, &, may be used to combine two or more strings:*

   "ABC" & "DEF" results in "ABCDEF".

Function References

built-in-function	A function name optionally followed by a series of arguments, enclosed in parenthesis.
function-name	ABS ATN CHR$ CLK$ COS COT DAT$ DET EXP INT LEN LOC LOF LOG MAR MOD NUM PER POS RND SEG$ SGN SIN STR$ TAN TIM USR$ VAL
user-defined-function	FN followed by a letter, an optional dollar sign ($), and an argument list enclosed in parentheses. Example: FNA (A + B, X–Y), FNC$ (B$, A)
argument-list	An expression optionally followed by up to 15 expressions separated by commas. Example: A + B, X–Y, A$, . . .

NOTES:

1. *SIN(x), COS(x), TAN(x), COT(x), and ATN(y) designate the functions sine, cosine, tangent, cotagent and arctangent respectively, and the argument x and the result of ATN are angles measured in radians.*

2. *EXP(x) designates exponentiation e. Overflow occurs if x is too large (i.e., $x > 174.6$)*

3. *LOG(x) designates the natural logarithm of x, ln x. The LOG of zero or a negative number is treated as an error.*

4. *ABS(x) designates the absolute value of x, $|x|$.*

5. *SQR(x) designates the square root of x. A negative argument is treated as an error.*

6. *RND(x) designates a pseudo random number as follows:*

   a. *If $x > 0$, then RND(x) is a function of x, whose value is in the open interval (0,1).*

   b. *If $x < 0$, the system supplies an arbitrary random number in the open interval (0,1).*

c. *If x = 0, the system supplies a pseudo random number which is a function of the previous random number generated by RND. If x = 0, the first time RND is called in a program, the system will supply a fixed number in the open interval (0,1).*

d. *If the argument is omitted (X = RND), then this is equivalent to an argument of zero.*

To generate a sequence of pseudo random numbers, the user would call any of these options followed by repeated calls to option (c).

7. *INT(x) designates the largest integer not exceeding x.*

   For example: INT(2.985) = 2. INT(−2.015) = −3.

8. *SGN(x) designates the sign of x.*

$$SGN = \begin{cases} +1, \text{ if } x > 0 \\ 0, \text{ if } x = 0 \\ -1, \text{ if } x < 0 \end{cases}$$

9. *DET returns the value of the determinant of the last matrix inverted.*

10. *TIM designates the current running time in seconds for the program.*

11. *LEN(x$) returns the length in characters of string x$.*

12. *VAL(x$) returns the numeric value of the decimal number which is in string x$.*

13. *MOD(x,y) returns the modulus remainder of X MOD Y This is similar to X−Y* INT (X/Y).*

14. *POS(x$,y$,z) begins searching string x$ at position z attempting to find string y$. This will return the starting position at which y$ is found within x$, or zero if not found.*

15. *CHR$ (x) designates the one character EBCDIC string with a character value equal to x.*

16. *CLK$ designates the current time of day in the form "HH:MM:SS".*

17. *DAT$ designates the current date in the form "MM/DD/YY".*

18. *SEG$(x$,y,z) returns the substring of x$ from character position y to position z.*

19. *STR$(x) returns the character string representation of the number x.*

20. *USR$ returns the logon-id of the terminal user.*

21. *LOC(#n) returns the current location of the file pointer for the file assigned to channel #n.*

22. *LOF(#n) returns the current end-of-file (number of records) for the file specified by #n.*

23. *MAR(#n) returns the current margin setting for the file #n.*

24. *PER(#n,a$) returns +1 if the operation specified for file # n is valid, 0 if not, and −1 if a$ is not one of the operations: INPUT, LINPUT, PRINT, READ, RENAME, RESET, SCRATCH, WRITE, or MARGIN.*

25. *TYP(#n, a$) returns +1 if the type specified by a$ corresponds to the file type of file #n, 0 if not, and −1 if a$ is not one of the types: ANY, LIBRARY, NUMERIC, PERM, RANDOM, STRING, TERMINAL, TTY, or WORK.*

26. *NUM returns the count of values inputted by the last MAT INPUT.*

27. *FNA to FNZ designates one of the 26 user-defined numeric functions (see the DEF statement).*

28. *FNA$ to FNZ$ designate one of the 26 user-defined string functions.*

29. *EBC (string) converts a string of from one to three characters in length to its EBCDIC value. The argument may be a letter, number, character, etc., or a 2 — 3 letter mnemonic such as ETX, DEL, etc.*

Statements

statement	Line-number followed by an executable statement or nonexecutable-statement. A comment may be added to any statement by prefixing it with an apostrophe (').
executable-statement	Assign, control, input-output, matrix, data file.
nonexecutable-statement	Declaration or remark statement.

NOTES:

1. *Each BASIC statement entered into a program must be prefixed with a line number. These line numbers determine the logical order of statements within a program. They are used in several of the control statements to effect transfers of control.*

2. *Each BASIC statement is summarized in the reference and described in detail in the BASIC reference manual, UA-0140 (current version).*

Syntax Conventions

In describing the statements, the following conventions are used:

file-params:	*Format 1:* program, file [(password)] [, volume] *Format 2:* {SQ / DA} ,file [(readpass/writepass)] [,volume]
program:	1 to 8 character program name to be located or saved.
file:	1 to 44 character file name of the file containing the program to be located or saved.
password:	Is the cataloged password for the file, if any. The read password must be supplied for an "OLD" and the write password for "SAVE". This entry is optional.
volume:	6 character volume name. If the file is listed in the catalog with a volume, this field may be omitted.
line-number:	A series of one to five digits.
list-item:	line number line number—line number
search-string:	list-item [,list-item . . .] "characters"
channel-setter:	Identifies the file on which an operation is performed. This has the form: #expression

NOTES:

1. *A line-number-list may contain list-items which reference single lines and others which reference a sequence of lines (all lines between the first and second line numbers specified, inclusive).*

   Example:

   120, 200 — 250, 300

   This list references those lines numbered 120, 200 to 250 inclusive, and 300.

2. *A channel-setter must result in a value between 0 and 4095. Channel zero, the terminal, may only be referenced by certain statements.*

Statement Formats

Format	Example
CALL closed-string [:param-list]	17 CALL "SUBR": 3+4, B (), #4
CHAIN {string-expression / channel-setter} [WITH channel-setter, . . .]	4197 CHAIN #4 WITH #I,#J 4522 CHAIN A$ WITH #3 9223 CHAIN "PROG, CATLIB"
CHANGE {string-name TO numeric-vector / numeric-vector TO string-name} [BIT expression]	512 CHANGE A$ TO V 675 CHANGE B TO A5$ BIT 4
DATA datum[,datum . . .]	330 DATA 4,2,1.7 340 DATA "YES", "NO", 3
DEF FNletter [$] [(param-list)] [,local-list] [= expression]	30 DEF FNE (X)=EXP (−X**2)
DIM letter [$] (integer [, integer]), . . .	1 DIM A(5), B$(6,3)
END	995 END
FILE channel-setter:string-expression	174 FILE #3: "MYFILE,MYLIB" 190 FILE #I: "*"
FNEND	37 FNEND
FOR numeric-variable=arithmetic-expression TO arithmetic-expression [STEP arithmetic-expression]	30 FOR X=0 TO 3 STEP D 80 NEXT X
GOSUB line number	90 GOSUB 210
GOTO	20 GOTO 25
If condition {GOTO / THEN / GOSUB} line number	40 IF SIN(X)=M THEN 80 41 IF A$ = "YES" GOSUB 79 42 IF END #I GO TO 4178
INPUT [channel-setter:] variable, variable . . .	40 INPUT B(5), C7$ 41 INPUT #3:K
[LET] {numeric-let / string-let / function-let}	259 LET W7 = X4**2 260 LET FNB$ = C$ 270 LET B9$ = X$(5)
LIBRARY closed-string, . . .	47 LIBRARY "SUBLIB, PACK22"
LINPUT [channel-setter:] string-variable, . . .	195 LINPUT L$, M5$, K$ (6) 423 LINPUT #7: Z$
MARGIN [channel-setter:] expression	2 MARGIN 120 4 MARGIN #6:64
MAT letter=letter+letter	615 MAT H=A + B
MAT letter= CON [(trimmer)]	100 MAT C=CON
MAT letter= IDN [(trimmer)]	20 MAT B=IDN
MAT letter=INV(letter)	550 MAT K=INV(L)

Format	Example
MAT INPUT [channel-setter:] letter [$], . . .	7 MAT INPUT #3: A, B$
MAT LINPUT [channel-setter;] letter $, . . .	8 MAT LINPUT L$, V$
MAT letter=letter*letter	612 MAT H=A*B
MAT PRINT [channel-setter:] letter [$] [;] . . .	140 MAT PRINT M,N;
MAT READ [channel-setter:] letter [$], . . .	159 MAT READ M,N
MAT letter=(arithmetic-expression)*letter	10 MAT F= (2.33 + M)*O
MAT letter = letter−letter	615 MAT H = A − B
MAT letter=TRN(letter)	300 MAT G=TRN(H)
MAT letter$=NUL$	47 MAT B$=NUL$
MAT WRITE [channel-setter:] letter [$], . . .	19 MAT WRITE #I: Q, V, W$
MAT letter = ZER [(trimmer)]	412 MAT X = ZER (5,3)
NEXT numeric-variable	80 NEXT X
ON arithmetic-expression {THEN / GOTO / GOSUB} line-number, . . .	150 ON X+Y GOTO 575, 490, 650
PAUSE	950 PAUSE
PRINT [channel-setter:] [item[;]] . . .	20 PRINT I; 27 PRINT #27: A$, "FILE"
RANDOMIZE	10 RANDOMIZE
READ [channel-setter:] variable, . . .	150 READ X, Y, Z$, V (5) 2144 READ #44: J,K
REM [character . . . .]	200 REM THIS SUBROUTINE 201 REM SOLVES EQUATIONS
{RESET / RESTORE} [channel-setter: [record-number]]	560 RESTORE 517 RESET #I:4 605 RESET #3
RETURN	350 RETURN
SCRATCH channel-setter	1900 SCRATCH #4
STOP	40 STOP
SUB closed-string [:param-list]	9000 SUB "INTEGRAL":A, FNC, J 1000 SUB "FINDSPAC"
SUBEND	9999 SUBEND
SUBEXIT	9510 SUBEXIT
TIME integer	10 TIME 30
WRITE channel-setter :expression, . . .	1445 WRITE #7 : A$, B, C
USING string-expression, expression, . . . [;]	145 MAT PRINT USING A$, B, C 170 PRINT #5 : USING "#.##", A (1), B (2)

Command Formats

Format	Example
BYE	BYE
DELETE [line-number-list] [search-string]	DELETE 1-200 "REM"
HELP	HELP
{LIST / PRINT} [line-number-list] [search-string]	LIST 110 — 150 LIST "FOR" PRINT 10,50
NEW	NEW
OLD file-params	OLD SORT, FILE, DISK01
RUNOLD file-params	RUNOLD PRINTIT, BASICLIB, PACK21
RUN	RUN
SAVE file-params	SAVE LOG, SRCE, DISK01