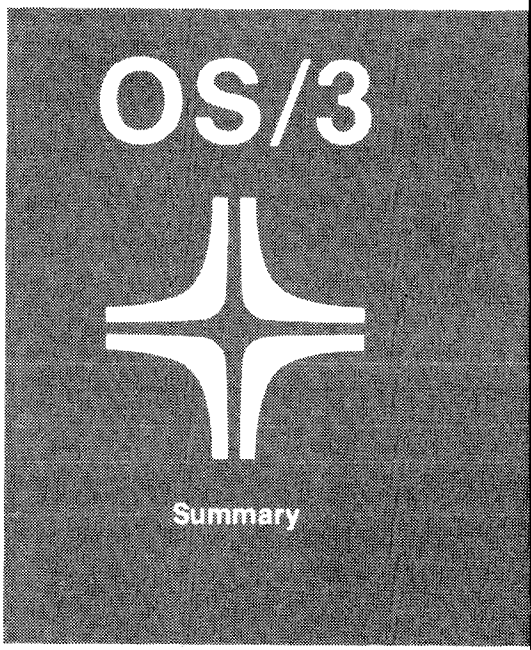
BASIC



SPERRY UNIVAC

UP-9169
Rev. 2

H

RELEASE LEVEL: 8.0 Forward

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

Contents

This summary contains information for use in preparing BASIC programs to be compiled by the BASIC compiler, as implemented on the SPERRY UNIVAC Operating System/3 (OS/3).

Detailed information is covered in the BASIC programmer reference, UP-9168 (current version).

LOGON PROCEDURE

For initial connection with the operating system, the user must enter the LOGON command in the system mode from a terminal. This command identifies the user to the operating system and initiates the user task. The format of the LOGON command is:

LOGON user-id [,acct][,password]

where:

user-id
: Is a 1- to 6-character alphanumeric code identifying the user to the system. The user-id is used by the system to correctly route messages as well as job and command output and to determine which commands may be used on the system. *The user-id must begin with an alphabetic character.*

acct
: Is a 1- to 4-character alphanumeric code used for system time accounting.

password
: Is a 1- to 6-character alphanumeric code that controls user access to the overall system.

SOURCE PROGRAM CONSTRUCTION

After logging on and being accepted by the system, the user calls BASIC by issuing the following executive command:

BASIC

Control is transferred to BASIC, which immediately responds:

BA001 OS/3 BASIC READY (VER xx.xx) BEGIN

At this time the user is at the command level in BASIC. If a command other than NEW or OLD is entered, the syntax checker is called immediately to process the first source statement.

After the compiler is called, the system responds with an asterisk, which indicates a request for source input. A line of input consists of a single BASIC source language or a BASIC editing command, followed by the TRANSMIT function.

LANGUAGE ELEMENTS

Characters

letter	ABCDEFGHIJKLMNOPQRSTUVWXYZ
digit	0123456789
delimiter (operator)	+ — * / ( ) < > & ↑
delimiter (separator)	, . ; △ " :
special character	$ @ # ? ' %
open-string character	letter, digit operator, special character, period (.), or semicolon (;)
string character	letter, digit, operator, special character, period (.), semicolon (;), comma (,), or a blank (△)

NOTES:

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

2. *The character quote (") delimits the beginning and end of a closed-string constant. If a quote is required within a closed string, use two consecutive quotes.*

3. *Exponentiation is specified with a pair of asterisks (**). A vertical arrow ↑ is also permitted for exponentiation, where applicable.*

Constants

decimal number	A fraction followed by optional exponent field. Fraction: Series of one or more 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: "ABZ154△84"
open string	A series of 1 to 4095 open-string characters, blanks, or quotes. Example: A△B
line number	Series of one to five digits without sign, decimal point, or exponent field. It must be in the range 1 to 99999.

LANGUAGE ELEMENTS (cont)

NOTES:

1. *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 floating-point format, the mantissa occupies 24 bits and contains a 6-digit hexadecimal number in normalized form. In BASIC, 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. *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 contain a comma. Moreover, an open-string constant is permitted only as input to the READ and INPUT statements. Note that it is not possible to enter a string constant longer than 74 characters in a program, because the maximum line length is 80 characters.*

3. *A line number is an integer between 1 and 99999, and must precede each statement in a BASIC program. The line numbers specify the logical sequence of statements in a program (ascending order). They are also used as statement labels for transferring control during program execution.*

   Leading zeros in a line number are ignored.

Variables

scaler variable	Defined as a numeric variable or a string variable.
numeric variable	A letter optionally followed by a single digit. Examples: X X2
string variable	A letter followed by a dollar sign ($), or a letter followed by a single digit followed by a dollar sign. Examples: A$ Q6$
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 ($) followed by one or two subscript expressions enclosed in parentheses. Examples: M$(2) C$(A+B) D$(A,C)

LANGUAGE ELEMENTS (cont)

NOTES:

1. *Numeric variables may only be assigned decimal numeric values.*

   Numeric array variables may only be assigned decimal numeric values.

2. *A subscript may be defined using any arithmetic expression. During execution, the value used to locate the array element referenced is computed by rounding the subscript expression to the nearest integer. 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.

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

4. *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	A string primary or string expression followed by an ampersand (&) denoting concatenation, followed by another string expression. Example: "ABC"&B$
string primary	A closed-string reference or function reference. Example: A$ SEG$ (D$,6,8) "AB"

NOTES:

1. *The exponentiation operator (**) may be written (where applicable) as a vertical 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*

NOTES (cont):

- *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.*

Function References

Numeric Valued Functions	
SIN(x)	sin(x)①
COS(x)	cos(x)①
TAN(x)	tan(x)①
COT(x)	cot(x)①
ATN(x)	$\tan^{-1}(x)$②
EXP(x)	e^x
LOG(x)	ln(x)
ABS(x)	$\lvert x \rvert$
SQR(x)	$\sqrt{x}$
RND(x)	if $x > 0$, function of x on (0,1) if $x < 0$, new random seed on (0,1) if $x = 0$, random on (0,1) if no argument, $x = 0$ assumed
INT(x)	Largest integer $\leq x$
SGN(x)	$+1$, if $x > 0$ 0, if $x = 0$ -1, if $x < 0$
FNA to FNZ (argument list)	User-defined numeric function
DET	Determinant value of last matrix inverted
LEN(A$)	Number of characters in string A$
MOD(x,y)	X—Y*INT(X/Y)
POS(A$,B$,X)	Position of B$ within A$ starting at X
TIM	Seconds since RUN command was issued
VAL(A$)	Value of string in A$
EBC(string)	EBCDIC value of 3-character maximum string
LOC (#N)	File pointer location for file in channel N
LOF (#N)	End-of-file value for file in channel N
MAR (#N)	Margin size for file in channel N
PER (#N,A$)	See BASIC programmer reference, UP-9168 (current version).

LANGUAGE ELEMENTS (cont)

Function References (cont)

Numeric Valued Functions	
TYP (#N,A$)	See BASIC programmer reference, UP-9168 (current version).
NUM	Number of values last vector MAT INPUT statement
String Valued Functions	
STR$(x)	Character string representation of value of x
USR$	User's LOGON command identifier
CHR$(x)	EBCDIC character code MOD(INT(x),256)
CLK$	Military time of day in form hh:mm:ss
DAT$	Date in form mm/dd/yy
SEG$ (A$,x,y)	Substring of A$ from position x for length of y
FNA$ to FNZ$ (argument list)	User-defined string functions

NOTES:

①*x in radians*

②*Result in radians*

Statements

statement	Line number followed by an executable statement or nonexecutable statement
executable statement	Assign, control, input/output, matrix, or data file statement
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 this reference and described in detail in the BASIC programmer reference, UP-9168 (current version).*

SYNTAX CONVENTIONS

In describing the statements, the following conventions are used:

1. Keywords that may be used in the statement are shown in capital letters.

2. Names constructed using lowercase letters and embedded hyphens designate syntactic variables.

3. Brackets, [ ], are used to enclose optional parameters.

4. Braces, { }, are used to enclose alternatives.

SYNTAX CONVENTIONS (cont)

5. Ellipsis, . . ., following an operand parameter indicates that the programmer may specify more than one parameter of that type. For example, the syntax

    READ variable [,variable . . .]

    allows the statements

    READ A

    READ A,B

    READ A,B,C

    to contain many input variables in the READ list.

The following syntactic units occur several times in the specification of the editing commands:

line-number: a series of one to five digits

line-number-list: list-item [,list-item . . .]

list-item:

$$\left\{ \begin{array}{l} \text{line-number} \\ \text{line-number} \left\{ \begin{array}{c} - \\ \text{TO} \end{array} \right\} \text{line-number} \end{array} \right\}$$

range: line-number (increment)

increment: a series of one to four digits

filename:

$$\left\{ \begin{array}{l} \text{letter} \left[\left\{ \begin{array}{l} \text{letter} \\ \text{digit} \end{array} \right\} \ldots \right] \\ \$\text{character[character . . .]} \end{array} \right\}$$

NOTES:

1. *A line-number-list may contain list-items that reference single lines and others that 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 line number range specifies the starting line number and an increment for calculating successive line numbers.*

    Example:

    100 (10)

    The range specified is 100, 110, 120, . . .

3. *In general, a file name consists of from one to eight letters or digits, the first of which is a letter. The length of the file name can be a maximum of 44 characters. Embedded characters, such as ., $, #, @, or —, may be included in this file name.*

4. *Except for the command verb and commands in which file names are specified, blanks are ignored.*

STATEMENT FORMATS

Format	Examples
CALL string-constant[:param-list]	17 CALL "SUBR":3+4,A,B() 18 CALL "FIND":#3,SIN,(A) 19 CALL "SEND":C(,),K(3,4),B$
CHAIN { string-expression / channel-setter } [WITH channel-setter,...]	23 CHAIN "PROGRAM2,CHAINLIB,PACK34" 24 CHAIN A$ WITH #3 25 CHAIN #4 WITH #1,#4,#J8
CHANGE { string TO array / array TO string-variable } [BIT expression]	34 CHANGE A$ TO V 35 CHANGE M TO B3$ 36 CHANGE G TO K1$ BIT 12
DATA { string-constant / numeric-constant }	45 DATA 1,3,6,1E3,—.34,17,3E34 47 DATA "STRING ONE",STRING TWO 49 DATA THIRD STRING,33
DEF FNletter[$][(param-list)] [,local-list] [expression]	54 DEF FND(X,Y)=SQR(X**2+Y**2) 55 DEF FNS$(X,Y$)=SEG$(Y$,X,X)$" " 56 DEF FNQ 57 DEF FNG$,I,J,K 58 DEF FNE(A,B,C),W,Z
DIM letter[$](integer[,integer]),...	67 DIM A(3),B(4,5) 68 DIM G$(45) 69 DIM C(100),H$(2,40)
END	78 END
FILE channel-setter:string-expression	82 FILE #3:"*" 83 FILE #R:"SQ,ERRORS,SPOOL3" 84 FILE #7:"COBLPR,LIBFILE(/WRPASS)" 85 FILE #J:A$
FNEND	88 FNEND
FOR numeric-variable=numeric-expression TO numeric-expression [STEP numeric-expression]	93 FOR I=3 TO 10 94 FOR J2=I TO POS(A$,B$,I) 95 FOR K=J2 TO L3 STEP 4
GOSUB line-number	102 GOSUB 943
GOTO line-number	111 GOTO 130
IF Format 1: expression test expression { GOTO / GOSUB / THEN } line-number Format 2: { END / MORE } channel-setter { GOTO / GOSUB / THEN } line-number	120 IF A$="YES" THEN 340 122 IF SIN(X)=0.5 GOTO 43 123 IF END#3 GOSUB 230
INPUT [channel-setter:]variable-name,...	130 INPUT A,B$ 140 INPUT #I:D(3,4),J
LET Format 1: numeric-variable [=numeric-variable...] =numeric-variable Format 2: string-variable [=string-variable...] =string-expression Format 3: FNletter[$]=expression	143 LET A$=SEG$(A$,3,4) 145 LET B(3,4)=SIN(Y) 147 LET FND=B(3,4)*A(4)+I
LIBRARY string-constant,...	155 LIBRARY "SUBLIBRARY,PACK11" 157 LIBRARY "CATALOGEDSUBLIBRARY(ALLOWD)"
MARGIN [channel-setter:]numeric-expression	160 MARGIN 120 164 MARGIN #3:64

STATEMENT FORMATS (cont)

Format	Examples
MAT letter=letter+letter	174 MAT A=B+C 175 MAT V=W+Z
MAT letter=CON[(trimmer)]	178 MAT A=CON 179 MAT V=CON(I)
MAT letter=IDN[(trimmer)]	185 MAT H=IDN(3,3) 188 MAT J=IDN
MAT letter=INV(letter)	190 MAT Q=INV(R)
MAT letter=letter*letter	198 MAT U=V*W 199 MAT A=V*B
MAT letter$=NUL$[(trimmer)]	201 MAT D$=NUL$ 205 MAT F$=NUL$(I,J) 206 MAT G$=NUL$(3)
MAT letter=(numeric-expression)*letter	212 MAT D=(J+4)*E 213 MAT V=(SIN(U))*W
MAT letter=letter—letter	221 MATD=F—E
MAT letter=TRN(letter)	234 MAT D=TRN(F)
MAT letter=ZER[(trimmer)]	244 MAT S=ZER 247 MAT E=ZER(3,4)
MAT INPUT [channel-setter:] letter[$][(trimmer)],...	253 MAT INPUT #3:B$
MAT LINPUT [channel-setter:] letter$[(trimmer)],...	255 MAT LINPUT #3:A$ 256 MAT LINPUT D$
MAT PRINT [channel-setter:] letter[$][separator],...	262 MAT PRINT A,B;C; 265 MAT PRINT #8:B$
MAT READ [channel-setter:] letter[$][(trimmer)],...	272 MAT READ A 277 MAT READ B$(3) 279 MAT READ #J+3:D(3,4)
MAT WRITE channel-setter:letter[$],...	281 MAT WRITE #3:A,B 283 MAT WRITE #1:K$,Y
NEXT numeric-variable	292 NEXT I 295 NEXT J5
ON numeric-expression {GOTO / GOSUB / THEN} line-number,...	320 ON J*(4+1) GOTO 120,300,120,430 111 ON K GOSUB 10,20,30,50,10,40
PAUSE	332 PAUSE
PRINT [channel-setter:] expression[separator],...	345 PRINT "THE ANSWER IS";A3 354 PRINT 1,J,K 356 PRINT TAB(I);L
RANDOMIZE	362 RANDOMIZE
READ [channel-setter:]variable,...	371 READ A,B,C 373 READ #4:A$(45) 377 READ #I:A3,B7$,C(2,3)
RESET [channel-setter:[numeric-expression]]	382 RESET 384 RESET #3 388 RESET #I:V3
REM [any characters for a comment]	391 REM THIS PROG COMPUTES WVB VALUES 392 REM FOR AN ARRAY 393 REM 394 REMARK
RETURN	395 RETURN
SCRATCH channel-setter	403 SCRATCH #3 404 SCRATCH #I—2
STOP	412 STOP
SUB string-constant:params	421 SUB "FINDSPAC"
SUBEND	437 SUBEND
SUBEXIT	449 SUBEXIT
SYSTEM string	476 SYSTEM "RUN"&P1$

STATEMENT FORMATS (cont)

Format	Examples
USING using-string,expression[,expression],...	127 PRINT USING A$,B,C 145 MAT PRINT USING "#.##III",B 167 PRINT #7:USING C1$,F$;G
WRITE channel-setter:expression,...	523 WRITE #3:A,SIN(X),B$

COMMAND FORMATS

Format	Examples
BYE	BYE
DELETE [line-number-list]["search-string"]	DELETE 10 DELETE 100—132 DELETE "INSTRUCTIONS" DELETE 1—100 "REM"
LIST [line-number-list]["search-string"]	LIST 3—4,10,100—200 LIST "PRINT" LIST 1—100 "REM"
MERGE element,library[(password)][,volume]	MERGE SUBR,SUBLIB,SUBPAK
NEW	NEW
OLD element,library[(password)][,volume]	OLD PRINTSIN,PROGRAMLIB,DISKPAK OLD COMPUTE,CATALOGUEDFILE
PRINT [line-number-list]["search-string"]	PRINT 3—4,10,100—200 PRINT "LINPUT" PRINT "END"9000—99999
RESEQUENCE start[:increment][:file-params]	RESEQ 100:50:RESPROG,PROGLIB,PACK57
RUN	RUN
RUNOLD element,filename[(password)][,volume]	RUNOLD COMPUTE,CATALOGUEDFILE
SAVE element,filename[(password)][,volume]	SAVE COMPUTE,CATALOGUEDFILE(PSWRD)
SYSTEM [system command]	SYSTEM SYSTEM FSTATUS PROGRAMLIB,PACK33
TIME	TIME 120

LOGOFF PROCEDURE

The LOGOFF command terminates the user session. Its purpose is to end the task and to return to the operating system any input/output devices used by the task.

This command, the last issued in the task, has the following format:

LOGOFF