IBM PL/I for VSE/ESA

Reference Summary

Release 1

SX26-3836-00

IBM PL/I for VSE/ESA

Reference Summary

Release 1

SX26-3836-00

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page iv.

First Edition (April 1995)

This edition applies to Version 1 Release 1 of IBM PL/I for VSE/ESA, 5686-069, and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department J58
P.O. Box 49023
San Jose, CA, 95161-9023
United States of America

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594, U.S.A.

Programming Interface Information

This book is intended to help the customer write programs using IBM PL/I for VSE/ESA. This book documents General-use Programming Interface and Associated Guidance Information provided by IBM PL/I for VSE/ESA.

General-use programming interfaces allow the customer to write programs that obtain the services of IBM PL/I for VSE/ESA.

Trademarks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
Language Environment
VSE/ESA

About this book

This book is intended to be used as a quick reference and backup to the IBM* PL/I for VSE/ESA* publications.

This book is divided into sections that contain the following categories:

- Quick reference to:

  Keywords
  Operands
  Options

- Lists of:

  Attributes (data elements)
  Built-in functions, pseudovariables, subroutines
  Characters, symbols, delimiters, operators
  Compiler options
  Conditions
  ENVIRONMENT attribute options
  Format items
  Source language keywords
  Statements

For information about IBM Language Environment* for VSE/ESA run-time options, see the *LE/VSE Programming Guide.*

Conventions used in this book

This book uses format conventions and syntax diagramming conventions in describing language and compiler elements.

Format conventions

The title of each option, command, or keyword contains the following:

- The name of the option, command, or keyword
- In parentheses, the abbreviation for the option, command, or keyword, if any
- After a dash, the classification of the option, command, or keyword

Syntax notation

Throughout this book, syntax is described using the following structure:

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line. The following table shows the meaning of symbols at the beginning and end of syntax diagram lines.

Symbol	Indicates
►►—	the syntax diagram starts here
—►	the syntax diagram is continued on the next line
►—	the syntax diagram is continued from the previous line
—►◄	the syntax diagram ends here

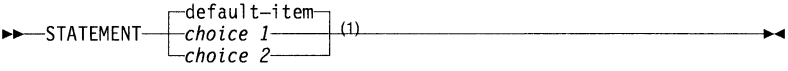
- Required items appear on the horizontal line (the main path).

- Optional items appear below the main path.

- When you can choose from two or more items, the items appear vertically, in a stack. If you **must** choose one of the items, one item of the stack appears on the main path. The default, if any, appears above the main path and is chosen by the compiler if you do not specify another choice.

  Note: In some cases, the default is affected by the:

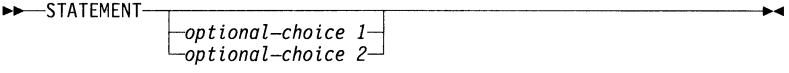
  - System in which the program is being run
  - Environmental parameters specified



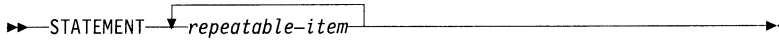
  Note:

  [1] Because *choice 1* appears on the horizontal bar, one of the items must be included in the statement. If you don't specify either *choice 1* or *choice 2*, the compiler implements the default for you.

  If choosing one of the items is optional, the entire stack appears below the main path.



- An arrow returning to the left above the main line is a *repeat arrow*, and it indicates an item that can be repeated.



  A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

- If there is a comma as part of the repeat arrow, you must use a comma to separate items in a series.

  If the comma appears below the repeat arrow line instead of on the line as shown in the previous example, the comma is optional as a separator of items in a series.

- A syntax fragment is delimited in the main syntax diagram by a set of vertical lines. The corresponding meaning of the fragment begins with the name of the fragment followed by the syntax, which starts and ends with a vertical line.

  fragment:

- Keywords appear in uppercase (for example, STATEMENT). They must be spelled exactly as shown. Variables appear in all lowercase letters (for example, *item*). They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or other symbols are shown, you must enter them as part of the syntax.

Part 1. Quick reference to keywords, operands, and options

Keywords, operands, and options

This chapter contains an alphabetized reference to:

- Compiler options
- Source language keywords

A — Format item

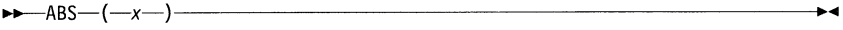
Specifies that *field-width* number of characters in the data stream is a character string.

Note: *field-width* is required on input.

Example: `GET EDIT(CHARSTR) (A(55));`

ABS — Built-in function

Returns the absolute value of x.



Example: `DIFF = ABS(X - Y);`

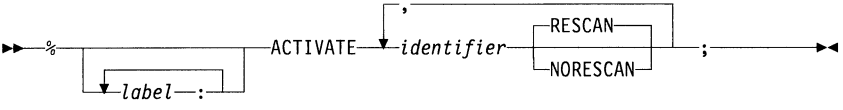
ACOS — Built-in function

Returns the arc cosine of x in radians.

Example: `A = ACOS(Y*F);`

%ACTIVATE (%ACT) — Preprocessor statement

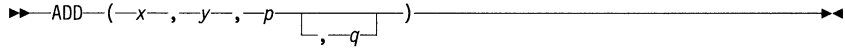
Activates an identifier for replacement in nonpreprocessor statement.



```
Example: %I = 15;
         ... %ACTIVATE I;
         ... %I=I+5;
         ... R=I*T*I;
         /* gives R=20*T*20 */
```

ADD — Built-in function

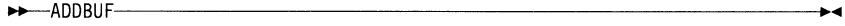
Returns the sum of x and y with a resulting precision given by p and scale factor q. q must be omitted if x or y is floating point.




```
Example: C = ADD(N,M,4,0);
```

ADDBUFF — Option of ENVIRONMENT attribute

Syntax checked only; has no effect. Kept for compatibility with previous releases of PL/I.




```
Example: ENVIRONMENT(ADDBUFF)
```

ADDR — Built-in function

Returns a pointer value identifying the location of x.

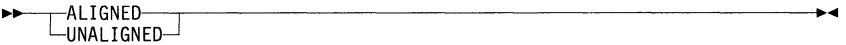

```
Example: P = ADDR(TABLE);
```

AGGREGATE (AG) — Compiler option

Lists aggregates and their size.

ALIGNED — Attribute

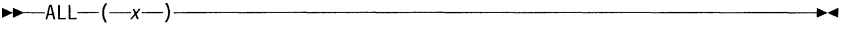
Specifies data is to be mapped according to alignment requirement of data type.



Example: DECLARE 1 A ALIGNED, 2 B, 3 C, 2 D;

ALL — Built-in function

Returns bit string (with length of longest element of array) resulting from logical AND of all the elements of array x.

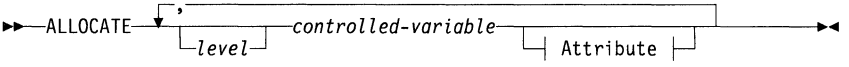


Example: IF ALL(LIST) THEN CALL SUB;

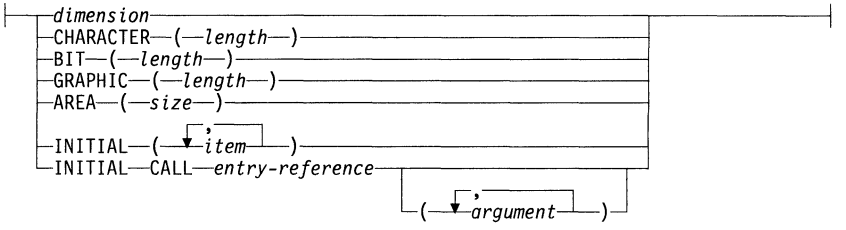
ALLOCATE (ALLOC) — Statement

Allocates storage for controlled or based variables.

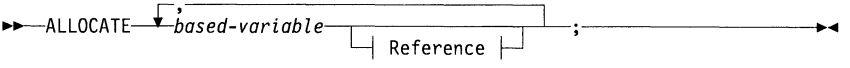
For controlled variables:



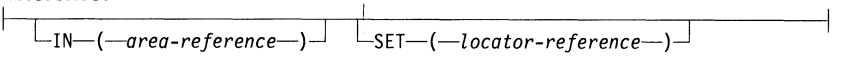
Attribute:



For based variables:



Reference:



Example: DECLARE A(N1,N2) CONTROLLED;
... ALLOCATE A;

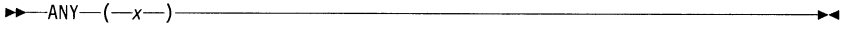
ALLOCATION (ALLOCN) — Built-in function

Returns the number of allocations of controlled variable x. Returns zero for no allocations.


```
Example: DO WHILE(ALLOCATION(A)>0);
           FREE A;
         END;
```

ANY — Built-in function

Returns bit string (with length of longest element of array) resulting from logical OR of all the elements of array x.




```
Example: IF ANY(TABLE) THEN CALL SUB;
```

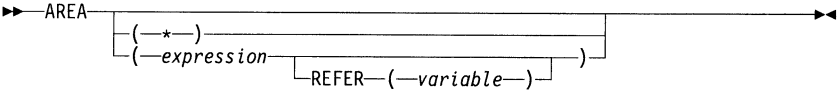
AREA — Condition

Occurs when (1) attempt is made to allocate a based variable in an area that has insufficient free storage, or (2) attempt is made to perform an area assignment and target area is too small to accommodate the source area.


```
Example: ON AREA CALL AREA_OVERFLOW;
```

AREA — Attribute

Reserves storage (in bytes) for the area. Default size is 1000 bytes.




```
Example: DECLARE MC AREA(4095);
```

ARGn — Option qualifier for the options NOMAP, NOMAPIN, and NOMAPOUT in OPTIONS attribute

Applies option to *n*th argument. If ARG*n* is not specified, option applies to all arguments.

See the syntax for "OPTIONS — Attribute" on page 79.

```
Example: DECLARE COBOLA ENTRY OPTIONS
           (COBOL NOMAP(ARG2)
           NOMAPOUT(ARG3));
```

ASCII — Option of ENVIRONMENT attribute

Specifies that the data set is in ASCII.


```
Example: ENVIRONMENT(ASCII)
```

ASIN — Built-in function

Returns arcsine of x in radians.


```
Example: B = ASIN(L*M);
```

ASSEMBLER (ASM) — Option of OPTIONS attribute

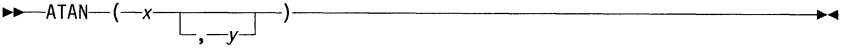
Specifies that entry point is in an assembler routine.

See the syntax for "OPTIONS — Attribute" on page 79.

```
Example: DECLARE ENTB ENTRY OPTIONS(ASM);
```

ATAN — Built-in function

Returns arctangent of x or of x/y radians.




```
Example: B = ATAN(L,M);
```

ATAND — Built-in function

Returns arctangent of x or of x/y degrees.

Example: `B = ATAND(L);`

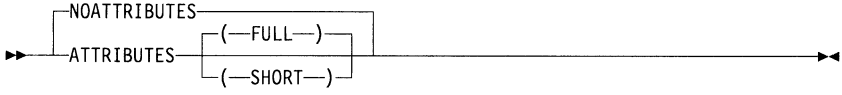
ATANH — Built-in function

Returns hyperbolic arc tangent of x.

Example: `B = ATANH(L + M / 2);`

ATTRIBUTES (A) — Compiler option

Specifies that the compiler includes a table of source-program identifiers and their attributes in the compiler listing. If SHORT is used, unreferenced identifiers are omitted, making the listing more manageable.



AUTOMATIC (AUTO) — Attribute

Specifies that storage is to be allocated upon each entry to the block in which the declaration is contained. Storage is freed when the block is terminated.

Example: `DECLARE TABLE(4,6) AUTO;`

B — Format item

Specifies that *field-width* number of characters in the data stream is a bit string. Each bit is represented by the character zero or one.

Note: *field-width* is required on input.

Example: `GET EDIT (STRING) (B(12));`

BACKWARDS — Option of OPEN statement

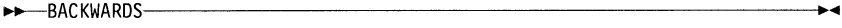
Augments the attributes specified in the file declaration.

See the syntax for "OPEN — Statement" on page 77.

Example: `OPEN FILE(X) BACKWARDS;`

BACKWARDS — Attribute

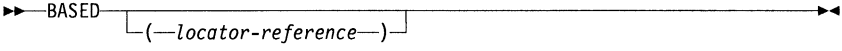
Causes the (implied) RECORD SEQUENTIAL INPUT tape file to be accessed in reverse order.



Example: `DECLARE TAPE FILE INPUT BACKWARDS ... ;`

BASED — Attribute

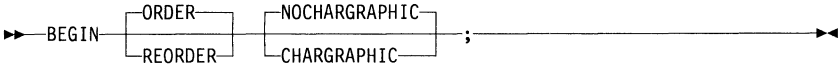
Specifies that the actual area of storage is identified by locator values. A based variable can be used to refer to variables of any storage class; it can also control its own storage by means of ALLOCATE and FREE statements.



Example: `DECLARE REC CHAR(30) BASED(P);`

BEGIN — Statement

Starts begin block.



Example: ON ZDIV BEGIN: ... END;

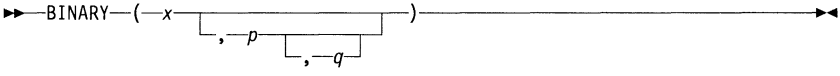
BINARY (BIN) — Attribute

Specifies base of data item as binary with optional precision.

Example: DECLARE A FIXED BIN(5,2);

BINARY (BIN) — Built-in function

Converts x to binary base with a resulting precision given by p and scale factor q. q must be omitted if x is floating point.



Example: B = BIN(I**M,25,6);

BINARYVALUE (BINVALUE) — Built-in function

Returns a REAL FIXED BIN(31,0) value that is the converted value of its pointer expression, x.

Example: N = BINARYVALUE(P);

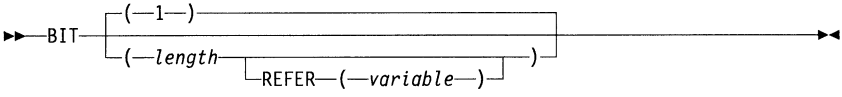
B — Bit constant

Data is represented by a series of the digits 0 and 1, enclosed in single quotation marks and followed immediately by the letter B.

Example: SWITCH = '1'B;

BIT — Attribute

Specifies data to be in bit-string form. The default length is one.



Example: DECLARE DATA BIT(8);

BIT — Built-in function

Converts expression x to a bit string of y bits. If y is omitted, the length is determined according to the rules for type conversion.

Example: IF BIT(C1||C2) = SW THEN ... ;

BKWD — Option of ENVIRONMENT attribute

Specifies backwards processing for a VSAM data set.

Example: ENVIRONMENT(VSAM,BKWD)

BLKSIZE — Option of ENVIRONMENT attribute

Specifies block size of data set associated with a file.

Example: ENVIRONMENT(BLKSIZE(200))

BOOL — Built-in function

Performs Boolean operation defined by z on x and y.

Example: IF TEST = BOOL(PATTERN,MASK,'0110'B)
THEN ... ;

BUFFERED (BUF) — Attribute

Requires each record of a SEQUENTIAL file to pass through buffers.

Example: DECLARE SALES FILE BUFFERED INPUT;

BUFFERED — Option of OPEN statement

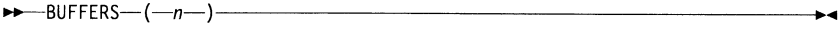
Specifies attributes that augment attributes specified in file declaration.

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(X) BUFFERED;

BUFFERS — Option of ENVIRONMENT attribute

Specifies number of buffers to be used.



Example: ENVIRONMENT(BUFFERS(1))

BUFND — Option of ENVIRONMENT attribute

Specifies number of data buffers for VSAM data set.

Example: ENVIRONMENT(VSAM,BUFND(4))

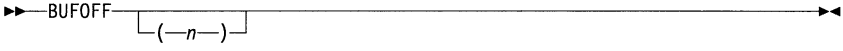
BUFNI — Option of ENVIRONMENT attribute

Specifies number of index buffers for VSAM data set.

Example: ENVIRONMENT(VSAM,BUFNI(2))

BUFOFF — Option of ENVIRONMENT attribute

Specifies the length of the block prefix on an ASCII data set; n, if specified, must be in the range 0 to 99; default is 4.



Example: ENVIRONMENT(BUFOFF(10))

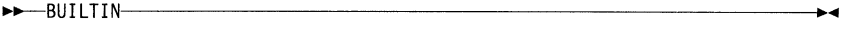
BUFSP — Option of ENVIRONMENT attribute

Specifies number of bytes required for buffers for VSAM data set.

Example: ENVIRONMENT(VSAM,BUFSP(6000))

BUILTIN — Attribute

Specifies that the associated name is a built-in function.



Example: DECLARE DATE BUILTIN;

BX — Bit hexadecimal constant

BX is a synonym for B4 (see "B4 — Bit hexadecimal constant").

B4 — Bit hexadecimal constant

Describes a BIT string constant in hexadecimal notation.

Example: `A = '80'B4;`

BY — Option of type 3 DO statement

Specifies the control-variable increment in an iterative DO statement.

See the syntax for "DO — Statement" on page 28.

Example: `DO I = 1 TO 8 BY NUM;`

BY — Option of repetitive specification

Specifies how much a quantity increments or decrements during each loop of the iterative DO statement.

See the syntax for "DO — Statement" on page 28.

Example: `PUT EDIT((A(I) DO I = 1 BY 2 TO N)) (F(4,2));`

BYADDR — Option of OPTIONS attribute

Specifies that the argument or parameter is passed or received by the address.

See the syntax for "OPTIONS — Attribute" on page 79.

Example:
```
DECLARE F ENTRY(FIXED BIN, PTR, CHAR(4))
          OPTIONS(BYADDR);
```

BYADDR — Option of OPTIONS option

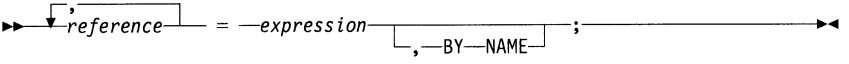
Specifies that the argument or parameter is passed or received by the address.

See the syntax for "OPTIONS — Attribute" on page 79.

```
Example: EXTR: PROC(A,B) OPTIONS(BYADDR);
               DCL (A,B) FLOAT;
```

BY NAME — Option of assignment statement

Limits structure assignment to those elements whose names, other than highest-level names, are common to all structures in the statement. (Comma must precede BY NAME.)




```
Example: PRINT = INPUT, BY NAME;
```

BYVALUE — Option of OPTIONS attribute

Specifies that the argument or parameter is passed or received by the value (contents).

See the syntax for "OPTIONS — Attribute" on page 79.

```
Example: DECLARE EXTR ENTRY (FIXED BIN(31), PTR) OPTIONS(BYVALUE);
```

BYVALUE — Option of OPTIONS option

Specifies that the argument or parameter is passed or received by the value (contents).

See the syntax for "OPTIONS — Attribute" on page 79.

```
Example: EXTR: PROC(P,Q) OPTIONS(BYVALUE);
               DCL (P,Q) POINTER;
```

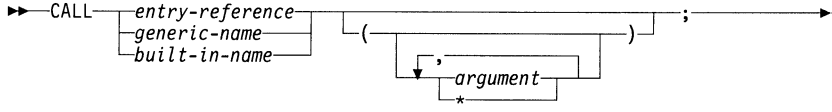
C — Format item

Describes external representation of the real and imaginary parts of complex data. If only one format item is given, both parts must have the same format.

Example: PUT EDIT (CUR) (C(F(10,2),F(8,2)));

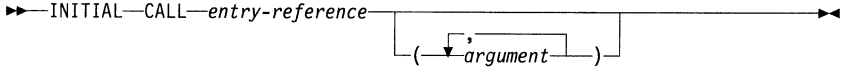
CALL — Statement

Invokes named procedure.



Example: CALL FRED;

CALL — Option of INITIAL attribute



Example: DECLARE A(8) INIT CALL A;

CEIL — Built-in function

Determines smallest integer greater than or equal to x.

Example: DECLARE ST CHAR(CEIL(LST/2));

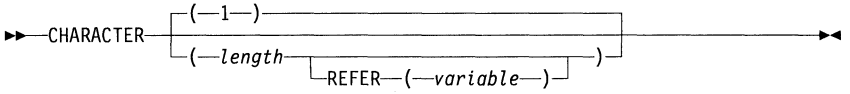
CHAR — Built-in function

Converts x to character string of length y. If y is omitted, the length is determined according to the rules for type conversion.

Example: CALL BC(SUBSTR(CHAR(B3,10),10,1));

CHARACTER (CHAR) — Attribute

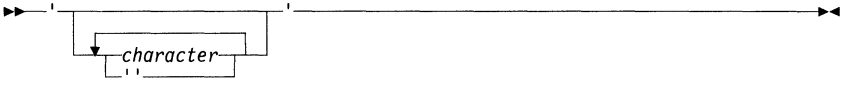
Specifies data to be in character form. The default length is one.



Example: DECLARE STR CHAR(8);

Character — Constant

A contiguous sequence of characters enclosed in single quotation marks.



Example 1: 'LOGARITHM TABLE'

Example 2: 'DON''T'

Note: Example 2 has a length of 5.

CHARGRAPHIC (CHARG) — Option of PROCEDURE and BEGIN statement

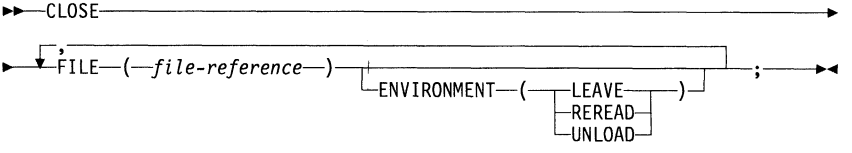
When in effect, all character string assignments are assumed to be mixed character assignments.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "BEGIN — Statement" on page 9.

```
Example: CH:PROCEDURE CHARGRAPHIC;
         BEGIN CHARGRAPHIC;
```

CLOSE — Statement

Dissociates a named file from its data set.



Example: CLOSE FILE(MASTER);

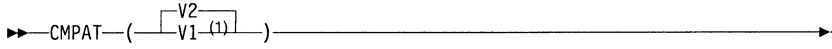
CMDCHN — Option of ENVIRONMENT attribute

(Only for IBM 3540 Diskettes.) Simulates blocked records by allowing you to specify a channel command word (CCW) chaining factor. (Actual blocked (FB) records are invalid for the 3540, because PL/I considers the 3540 to be a unit record device.) If specified, you must explicitly open the file.

Example: `ENVIRONMENT(CMDCHN(2) MEDIUM(SYS012))`

CMPAT (CMP) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.



Note:

[1] If specified, compilation proceeds as if CMPAT(V2) had been coded.

COBOL — Option of OPTIONS attribute

Specifies that the designated entry point is in a COBOL routine.

See the syntax for "OPTIONS — Attribute" on page 79.

Example: `DECLARE E ENTRY OPTIONS(COBOL);`

COBOL — Option of OPTIONS option

Specifies that the entry point is to be invoked only by a COBOL routine.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78.

Example: `C:PROCEDURE(X,Y,Z) OPTIONS(COBOL);`

COBOL — Option of ENVIRONMENT attribute

Specifies that structures in the data set associated with the file will be mapped according to the COBOL mapping algorithm.

Example: ENVIRONMENT(COBOL)

COLUMN (COL) — Format item

Positions the file to the character position specified by n.

Example: PUT EDIT (X) (COL(8),A);

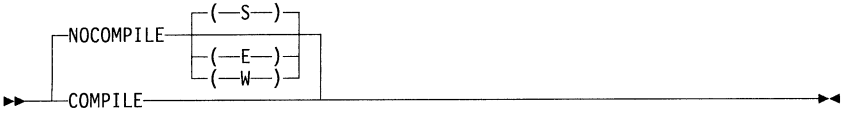
COMPAT — Option of ENVIRONMENT attribute

Provides REGIONAL(3) compatibility with DOS PL/I file format.

Example: ENVIRONMENT(COMPAT REGIONAL(3))

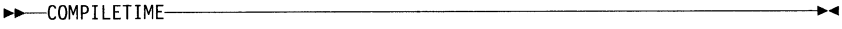
COMPILE (C) — Compiler option

Specifies that the compiler compiles the source program unless an unrecoverable error is detected during preprocessing or syntax checking.



COMPILETIME — Preprocessor built-in function

Returns the time and date of compilation as a preprocessor character string.



Example: %STR = COMPILETIME;

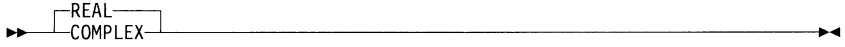
COMPLETION (CPLN) — Built-in function/pseudovariable

Refers to completion of event x; '1'B for event complete or '0'B if incomplete.


```
Example: IF COMPLETION(E2) THEN ... ;
         COMPLETION(E2)='1'B;
```

COMPLEX (CPLX) — Attribute

Specifies mode of arithmetic data as complex.




```
Example: DECLARE AMP FIXED DEC (5,3) CPLX;
```

COMPLEX (CPLX) — Built-in function/pseudovariable

Refers to a complex value as the two real numbers x and y.


```
Example: ROOT = CPLX(P1,8);
```

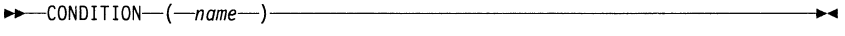
CONDITION (COND) — Attribute

Specifies that the associated identifier is a condition name.


```
Example: DECLARE X COND;
```

CONDITION (COND) — Condition

Specifies a programmer-named condition, identified by name, that can be raised only by a SIGNAL statement.




```
Example: ON CONDITION(F) A = B;
          ... SIGNAL CONDITION(F);
```

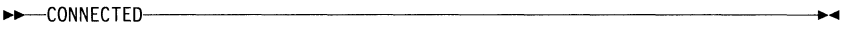
CONJG — Built-in function

Returns the conjugate of the complex value x.

Example: `C_ROOT = CONJG(ROOT);`

CONNECTED (CONN) — Attribute

Specifies that a parameter refers to connected storage only.




```
Example: SUB1:PROC(A);
         DECLARE A(10) CHAR(1) CONN;
```

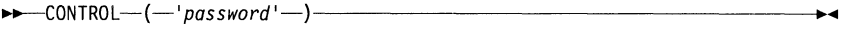
CONSECUTIVE — Option of ENVIRONMENT attribute

Specifies records to be in physical order. Default for all files.

Example: `ENVIRONMENT(CONSECUTIVE)`

CONTROL — Compiler option

Gives access to any compiler options restricted at installation.



CONTROLLED (CTL) — Attribute

Specifies that storage is controlled by ALLOCATE and FREE statements. Declared name refers to current generation only.


```
Example: DECLARE PROFIT CTL;
          ... ALLOCATE PROFIT;
```

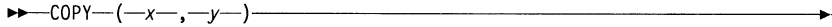
CONVERSION (CONV) — Condition/condition prefix

Raised for invalid conversion attempt on character-string data.


```
Example: ON CONVERSION BEGIN;
           ... END;
```

COPY — Option of GET statement

Causes source data to be written on specified file. Default is SYSPRINT file.




```
Example: GET FILE(MASTER) LIST(A,B,C)
           COPY(MASTOUT);
```

COS — Built-in function

Returns cosine of x, where x is in radians.


```
Example: A = COS(SIN(D)**2);
```

COSD — Built-in function

Returns cosine of x, where x is in degrees.


```
Example: X = COSD(B);
```

COSH — Built-in function

Returns hyperbolic cosine of x.


```
Example: A = COSH(E*F);
```

COUNT — Built-in function

Determines number of data items transmitted during last GET or PUT statement on file x.

Example: `N = N + COUNT(PAYROLL);`

COUNTER — Preprocessor built-in function

Returns a character-string containing a unique decimal integer in the range 00001 to 99999.

Example: `%LAB = 'AA'||COUNTER;`

CTLASA — Option of ENVIRONMENT attribute

Specifies that an American National Standard carriage control character is the first byte in every record.

Example: `ENVIRONMENT(CTLASA)`

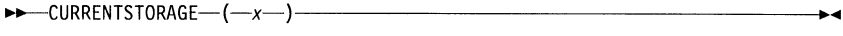
CTL360 — Option of ENVIRONMENT attribute

Specifies that an IBM system carriage control character is the first byte in every record.

Example: `ENVIRONMENT(CTL360)`

CURRENTSTORAGE (CSTG) — Built-in function

Returns the size in bytes of the storage currently occupied by the variable x.



Example: `I = CURRENTSTORAGE(X);`

D — Option of ENVIRONMENT attribute

Specifies unblocked variable-length records for ASCII data sets.

Example: ENVIRONMENT(D)

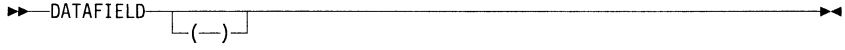
DATA — Option of GET or PUT statements

Specifies data-directed transmission of data names and values.

Example: PUT DATA (SUM,PROF);

DATAFIELD — Built-in function

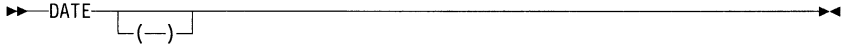
Returns a character string giving contents of data field responsible for raising of NAME condition.



Example: ON NAME(INFILE) PUT SKIP
EDIT('UNKNOWN:', DATAFIELD) (A);

DATE — Built-in function

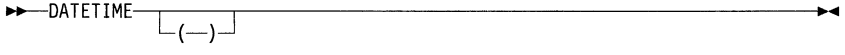
Returns a current date in character-string form, yymmdd for year, month, and day.



Example: PUT LIST (DATE());

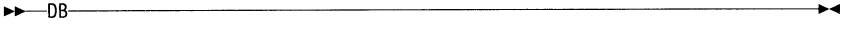
DATETIME — Built-in function

Returns a current date in character-string form, yyyymmddhhmmssttt, for year, month, day, hour, minute, second, and millisecond.



DB — Option of ENVIRONMENT attribute

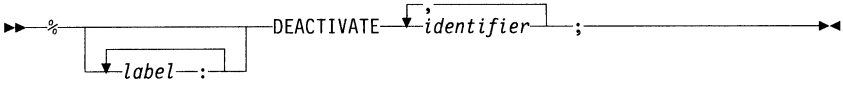
Specifies blocked variable-length records for ASCII data sets.



Example: ENVIRONMENT(DB)

%DEACTIVATE (%DEACT) — Preprocessor statement

Makes identifier ineligible for replacement by preprocessor.



Example: %DEACTIVATE I;

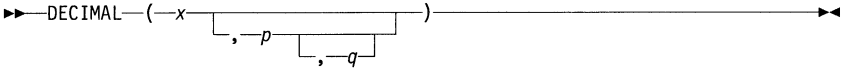
DECIMAL (DEC) — Attribute

Specifies data to have decimal base with optional precision.

Example: DECLARE B FIXED DECIMAL (4,2);

DECIMAL (DEC) — Built-in function

Converts the value of x to decimal base with a resulting precision given by p and q. q must be omitted if x is floating point.



Example: CALL SNO(DECIMAL(X,8,2));

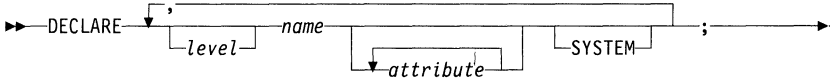
DECK (D) — Compiler option

Produces an object module in punched card format.

Note: This option can only be specified via the JCL OPTION statement (OPTION DECK). It will be ignored if specified as a compiler option.

DECLARE (DCL) — Statement

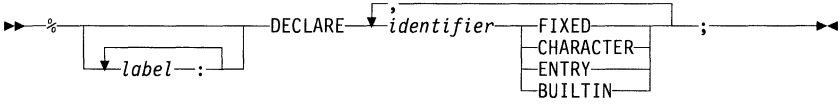
Used to declare variables.



Example: DECLARE ((A,B) FIXED(10), C FLOAT(5)) EXTERNAL;

%DECLARE (%DCL) — Preprocessor statement

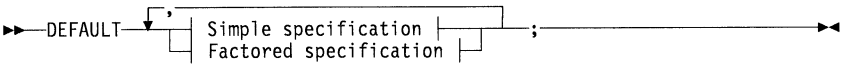
Identifies and activates as a preprocessor variable or preprocessor procedure name.



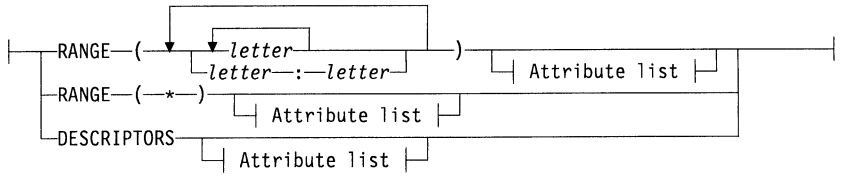
Example: %DECLARE A CHARACTER;

DEFAULT (DFT) — Statement

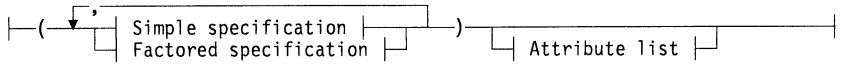
Specifies default attributes for designated identifiers.



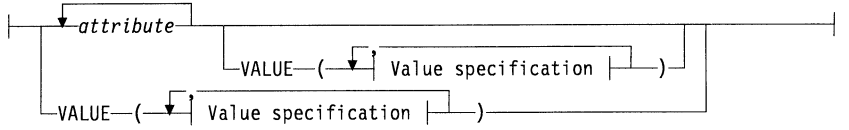
Simple specification:



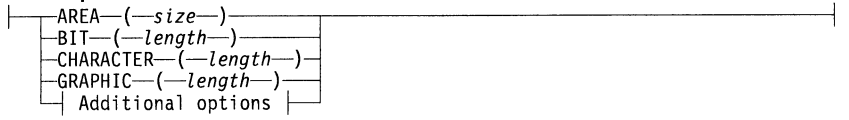
Factored specification:



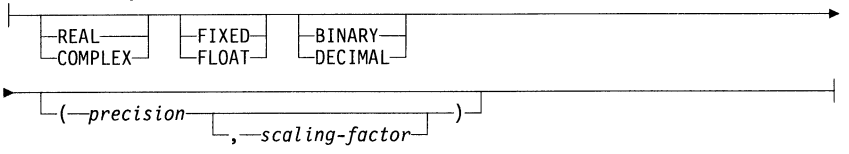
Attribute list:



Value specification:



Additional options:



Example: DEFAULT RANGE(X:Y) FIXED;

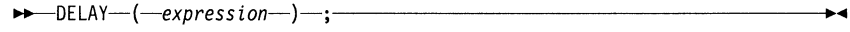
DEFINED (DEF) — Attribute

Specifies that the defined variable is to occupy all or part of the same storage as the designated base variable

Example: DECLARE A(8,8,8), D(2,2,2) DEFINED A;

DELAY — Statement

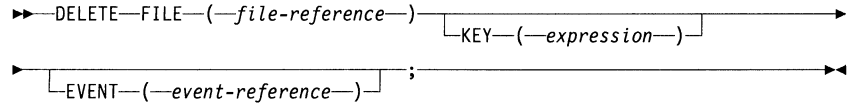
Suspends execution for specified number of milliseconds.



Example: DELAY(15);

DELETE — Statement

Deletes record from UPDATE file.



Example: DELETE FILE(STOCK) KEY('GX436211');

DESCRIPTORS — Option of DEFAULT statement

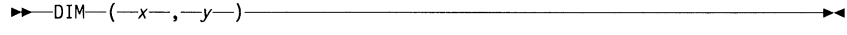
Specifies default attributes for parameter descriptors.

See the syntax for "DEFAULT (DFT) — Statement" on page 25.

Example: DEFAULT DESCRIPTORS FIXED;

DIM — Built-in function

Finds current extent of dimension y of array x.



Example: DECLARE TABLE2(DIM(TABLE1,3));

DIRECT — Option of OPEN statement

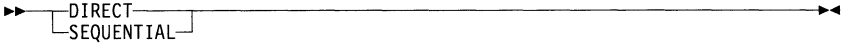
Specifies attributes that augment attributes specified in the file declaration.

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(X) DIRECT;

DIRECT — Attribute

Specifies records of a file to be accessed in any order by use of a key.

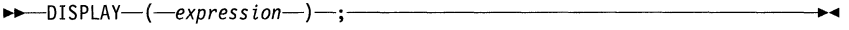


Example: DECLARE PAYROLL FILE DIRECT ... ;

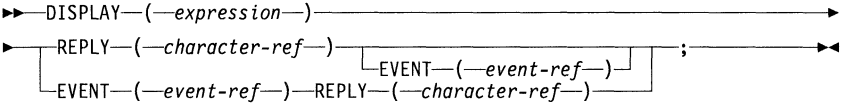
DISPLAY — Statement

Displays message to operator.

Type 1:



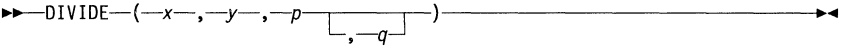
Type 2:



Example: DISPLAY ('MOUNT TAPE');

DIVIDE — Built-in function

Divides x by y with a resulting precision given by p and scale factor of q. q must be omitted if x or y is floating point.

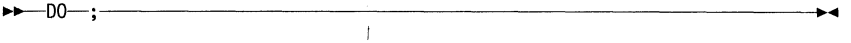


Example: PUT SKIP LIST(DIVIDE(KM,HRS,5,2));

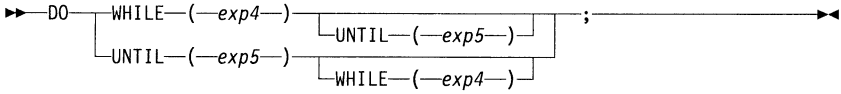
DO — Statement

Delimits a group of statements and can specify repeated execution.

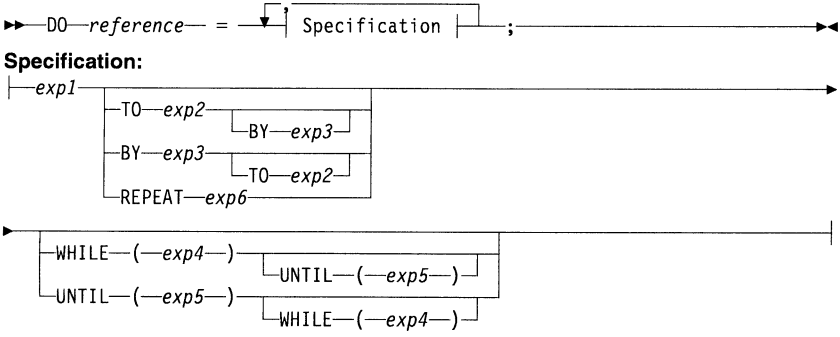
Type 1:



Type 2:



Type 3:




```
Example: IF A = B THEN DO;
         DO I = 40 TO 0 BY -2 WHILE(X>Z) UNTIL (Z = 0);
           ... END;
           ... END;
```

DO — Repetitive specification

Delimits a DO-group and can specify repeated execution.

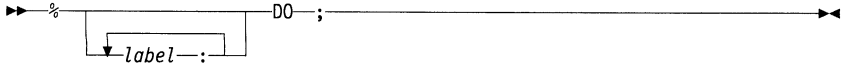
See the syntax for "DO — Statement" on page 28.

```
Example: PUT EDIT((A(I,J) DO I = 1 TO N))                (F(4,2));
```

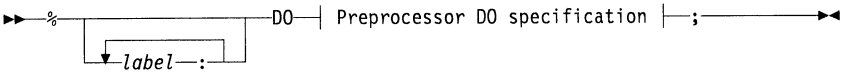
%DO — Preprocessor statement

Delimits a preprocessor DO-group and can specify repeated execution.

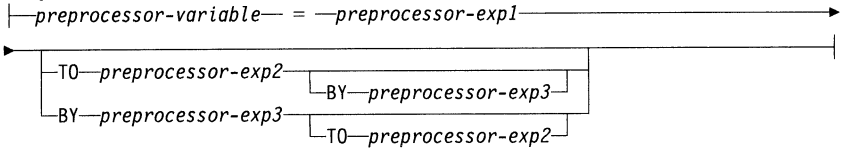
Type 1:



Type 3:



Preprocessor DO specification:



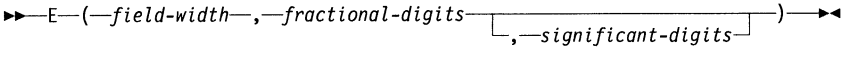

```
Example: %DECLARE I FIXED;
         %DO I=1 TO 10;
         Z(I)=X(I);
         %END;
```

DSN — Option of ENVIRONMENT attribute

Data set name sharing. Specifies the number of strings to be allocated to the data set for shared access.

E — Format item

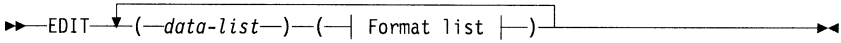
Describes decimal arithmetic data in the data stream to be in floating point formal; w=field width, d=fractional digits, s=significant digits, s is not required on input, and if it is omitted on output, s=d+1 is assumed.



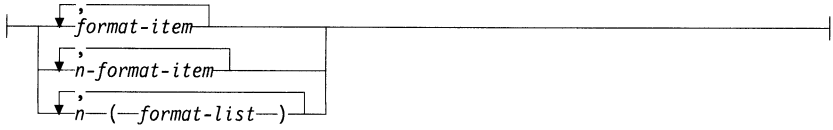

```
Example: GET EDIT (TEMP) (E(8,6));
```

EDIT — Option of GET or PUT statements

Specifies edit-directed transmission of data.



Format list:



Example: PUT EDIT('PAY = $',P) (A(5),F(6,2));

ELSE — Clause of IF

Defines action to be taken if value of expression is '0'B.

See the syntax for "IF — Statement" on page 45.

```
Example: IF A = B THEN C = D + E;
         ELSE H=F;
```

%ELSE — Clause of %IF preprocessor statement

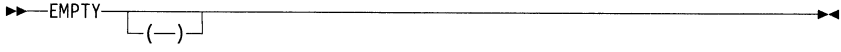
Defines action to be taken if value of expression is '0'B.

See the syntax for "%IF — Preprocessor statement" on page 45.

```
Example: %IF A = B %THEN C - D + E;
         %ELSE H=F;
```

EMPTY — Built-in function

Returns an area of zero extent and can be used to free all allocations of based variables in an area variable.



Example: TABLE = EMPTY;

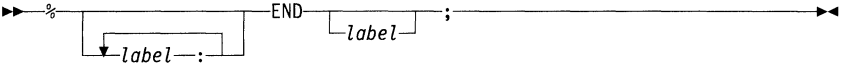
END — Statement

Delimits blocks (PROCEDURE and BEGIN) and groups (DO, SELECT).


```
Example: DO;
           ... END;
         LAST:PROC ... ;
           ... END LAST;
```

%END — Preprocessor statement

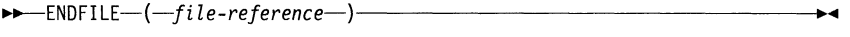
Delimits preprocessor DO-groups or procedures.




```
Example: %PP:PROC;
           ... %END;
```

ENDFILE — Condition

Raised when an attempt is made to read past the file mark.




```
Example: ON ENDFILE(IN) ... ;
```

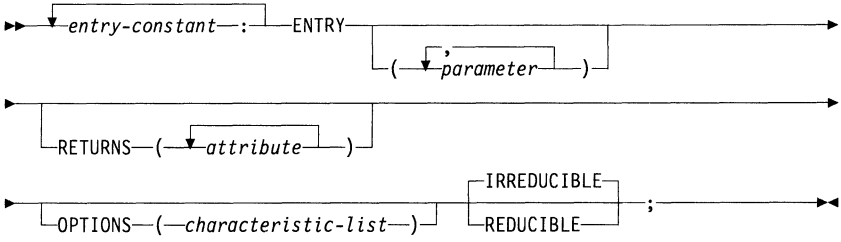
ENDPAGE — Condition

Raised when an attempt is made to write past the PAGE-SIZE limit.


```
Example: ON ENDPAGE(PRINTER) PUT FILE(PRINTER)
           PAGE EDIT(HEADING) (A);
```

ENTRY — Statement

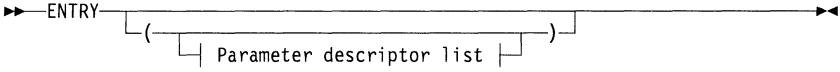
Specifies secondary entry point to a procedure.



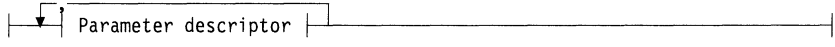
Example: A3: ENTRY;

ENTRY — Attribute

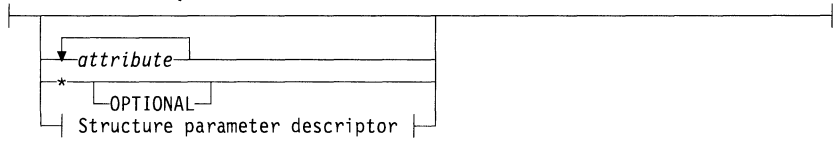
Specifies identifier as an external entry constant or as an entry variable.



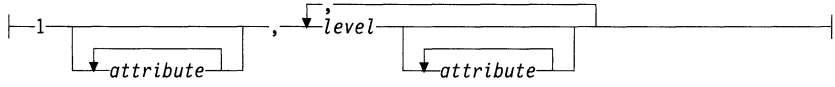
Parameter descriptor list:



Parameter descriptor:



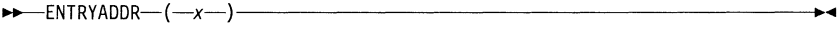
Structure parameter descriptor:



Example: DECLARE TEST ENTRY(DEC FIXED(4));

ENTRYADDR — Built-in function/pseudovariable

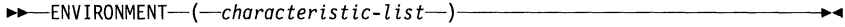
Returns a pointer value that is the address of the first executed instruction if a CALL is made to the module associated with an ENTRY variable or constant entry reference, x. The module must be an external entry.



Example: P = ENTRYADDR(E);

ENVIRONMENT (ENV) — Attribute of DECLARE statement

Describes data set or file characteristics. The options in the characteristic-list are separated by blanks or commas.



Example:
```
DECLARE LIST FILE STREAM
          ENVIRONMENT(CONSECUTIVE F RECSIZE(80));
```

ENVIRONMENT (ENV) — Option of CLOSE statement

Specifies disposition of the data set being closed.

See the syntax for "CLOSE — Statement" on page 16.

Example: CLOSE FILE(F) ENVIRONMENT(LEAVE);

ERF — Built-in function

Returns the value of the error function for x.

Example: S = ERF(8);

ERFC — Built-in function

Returns the value 1-ERF(x).

Example: S = ERFC(SIN(X)**2);

ERROR — Condition

Raised by an error for which there is no specific condition, or by standard system action.

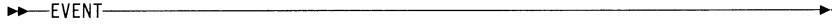
Example: ON ERROR SNAP SYSTEM;

ESD — Compiler option

List of external symbol dictionary.

EVENT — Attribute

Specifies identifier as an event name.



Example: DECLARE S1 EVENT;

EVENT — Option of data transmission statement

Specifies that the input or output operation is to take place while other processing continues and that no I/O conditions (except for UNDEFINEDFILE) are raised until a WAIT statement, specifying the same event variable, is executed.


```
Example: READ FILE (MASTER) INTO (REC_VAR)
           EVENT (RECORD_1);
             .
             .
           WAIT (RECORD_1);
```

EXCLUSIVE (EXCL) — Attribute

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Example: DECLARE FILE1 EXCLUSIVE;

EXCLUSIVE — Option of OPEN statement

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Specifies EXCLUSIVE attribute for file.

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(LISTING) EXCLUSIVE;

EXIT — Statement

Causes abnormal termination of program.

Example: EXIT;

EXP — Built-in function

Returns e**x where e is base of natural logarithms.

Example: POWER = (EXP(X1 + X2));

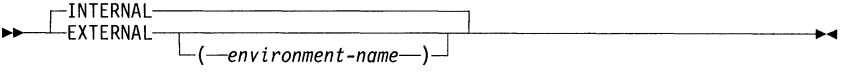
EXTENTNUMBER — Option of ENVIRONMENT attribute

Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.

Example: ENVIRONMENT(REGIONAL(1) EXTENTNUMBER(2))

EXTERNAL (EXT) — Attribute

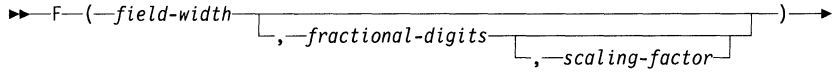
Indicates identifier is known in other external procedures in which it has the EXTERNAL attribute.



Example: DECLARE XYZ ENTRY EXTERNAL('ABC') OPTIONS(ASM);

F — Format item

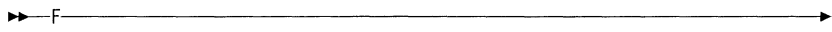
Specifies that *field-width* number of characters in the data stream is decimal arithmetic data in fixed point form. If *fractional-digits* or *scaling-factor* is omitted, zero is assumed.



Example: PUT EDIT (HRS) (F(4,2));

F — Option of ENVIRONMENT attribute

Specifies fixed-length unblocked records.



Example: ENVIRONMENT(F RECSIZE(80))

FB — Option of ENVIRONMENT attribute

Specifies fixed-length blocked records.


```
Example: ENVIRONMENT(FB BLKSIZE(480)
           RECSIZE(60))
```

FETCH — Statement

Loads the named procedure.


```
Example: PROG:  PROCEDURE;
                FETCH PROGA;
```

FETCHABLE — Option of OPTIONS option

Indicates that an external procedure can be fetched and is entered through this entry point.

See the syntax for "OPTIONS — Attribute" on page 79.

Example: EXTP: PROC OPTIONS(FETCHABLE);

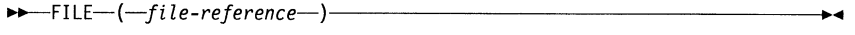
FILE — Attribute

Specifies an identifier as a file constant or variable.

Example: DECLARE TAX FILE ... ;

FILE — Option of input/output, OPEN, and CLOSE statements

Specifies file to be operated on.



Example: GET FILE (REPORT) LIST (A,B);

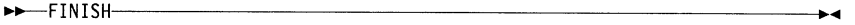
FILESEC — Option of ENVIRONMENT attribute

(Only for IBM 3540 Diskette output files.) Specifies that the operator must authorize any future attempts to read the diskette volume.

Example: ENVIRONMENT(CMDCHN(2) FILESEC MEDIUM(SYS013))

FINISH — Condition

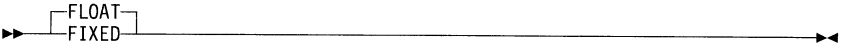
Raised by standard system action for the ERROR condition, by the END or RETURN statement of the initial procedure, by the STOP statement, or by the EXIT statement.



Example: ON FINISH PUT DATA (SUMMARY);

FIXED — Attribute

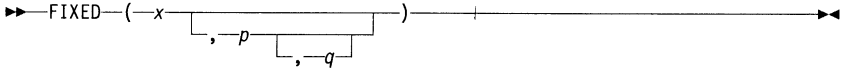
Specifies that a variable represents a fixed-point data item with optional precision.



Example: DECLARE A FIXED(3,1);

FIXED — Built-in function

Converts x to fixed-point with a resulting precision given by p and scale factor q. If p and q are omitted, default precision is assumed. If q is omitted, zero is assumed.



Example: `T3A = FIXED(1,6,5) / T3;`

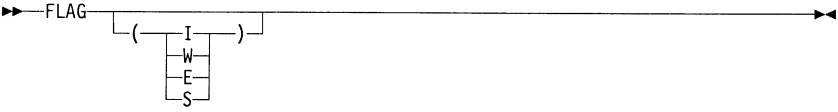
FIXEDOVERFLOW (FOFL) — Condition/condition prefix

Occurs when length of FIXED result exceeds implemented maximum.

Example: `ON FOFL PUT DATA(A,B,C);`

FLAG (F) — Compiler option

Suppresses diagnostic messages below a certain severity.



FLOAT — Attribute

Specifies that a variable represents a floating-point data item with optional precision.

Example: `DECLARE A FLOAT(8);`

FLOAT — Built-in function

Converts x to floating-point scale with a resulting precision given by p. If p is omitted, default precision is assumed.

Example: CALL PB2(FLOAT(C));

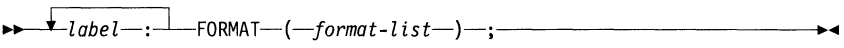
FLOOR — Built-in function

Returns largest integer not exceeding x.

Example: DO I = FLOOR(K / 2) TO FLOOR(K);

FORMAT — Statement

Specifies remote format list for edit-directed transmission.



Example: PUT EDIT(KTOWN) (R(FOR));
... FOR:FORMAT(SKIP, COL(5),A(14));

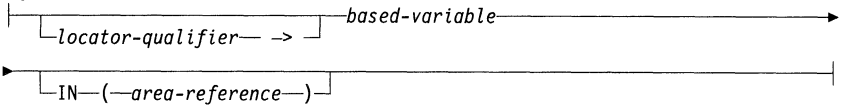
FREE — Statement

Frees allocated storage.

For controlled variables:

For based variables:

Option:



Example: FREE A,B,C;

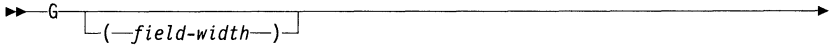
FROM — Option of WRITE or REWRITE statements

Specifies the variable to be written into a data set.

Example: `WRITE FILE (MASTER) FROM (BB);`

G — Format item

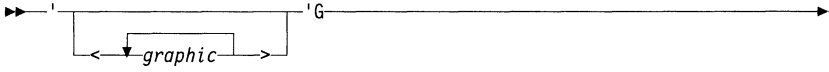
Describes the representation of a graphic string. *field-width* is the number of DBCS characters in the string.



Example: `PUT EDIT(GNAME) (G(12));`

G — GRAPHIC constant

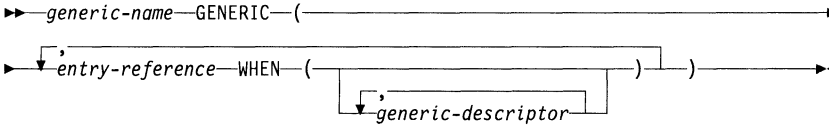
Can contain values in the range '00'X through 'FF'X in both bytes, except for the shift-out/shift-in values '0E'X and '0F'X. Constant must contain an even number of bytes and be enclosed in single quotes followed by the suffix G.



Example: `'<GGGG>'G`

GENERIC — Attribute

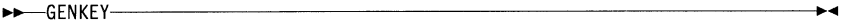
Defines a set of entry references.



Example:
```
DECLARE COMP GENERIC
        (A WHEN(FIXED), B WHEN(FLOAT));
```

GENKEY — Option of ENVIRONMENT attribute

Indicates records on data set are selected according to initial character(s) of recorded keys.

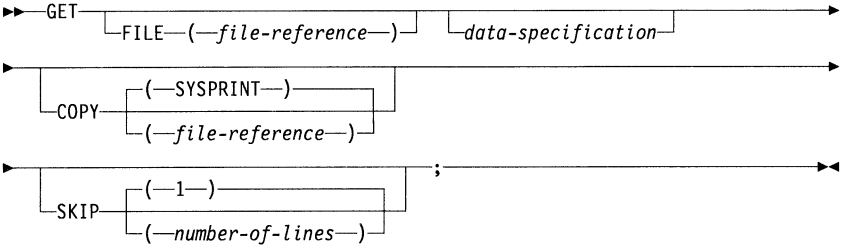



```
Example: DECLARE IND SEQUENTIAL KEYED INPUT
           FILE ENVIRONMENT(INDEXED GENKEY
           KEYLENGTH(6) F ... );
         READ FILE(IND) INTO(INFIELD)
           KEY('POP');
```

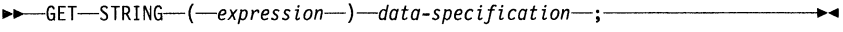
GET — Statement

Used for stream-oriented input or for data movement in main storage.

For a stream input file:



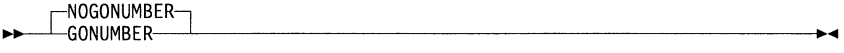
For transmission from a string:




```
Example: GET DATA;
         GET LIST (A,B,C);
         GET STRING(STR) EDIT(A,B) (A(10),A(7));
```

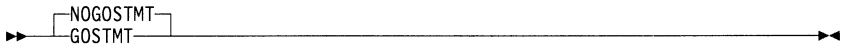
GONUMBER (GN) — Compiler option

Specifies that the compiler is to produce additional information that allows line numbers from the source program to be included in run-time messages.



GOSTMT (GS) — Compiler option

Specifies that the compiler produces additional information that allows statement numbers from the source program in run-time messages.



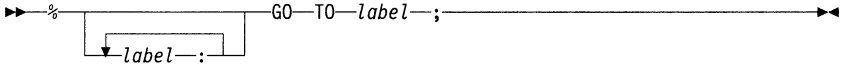
GO TO, GOTO — Statement

Transfers control to a specified label.

Example: GOTO LOOP;

%GO TO (%GOTO) — Preprocessor statement

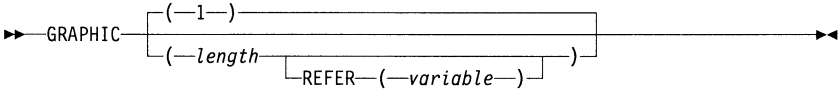
Causes preprocessor to continue scanning at specified preprocessor label.



Example: %GOTO CALC;

GRAPHIC (G) — String attribute

Specifies a graphic variable. Each graphic string element uses two bytes of storage. Its primary use is in support of kanji characters.



Example: DECLARE DBCS_VAR GRAPHIC(16);

GRAPHIC — Option of ENVIRONMENT attribute

Specifies stream I/O file contains graphics.

Example: DECLARE OUTFILE FILE OUTPUT STREAM
ENVIRONMENT(GRAPHIC);

GRAPHIC — Built-in function

Explicitly converts character (or mixed character) data to GRAPHIC data.

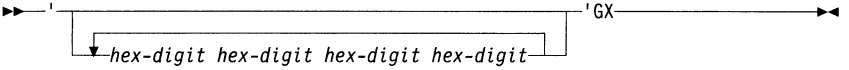

```
Example: DECLARE X CHAR (10) VARYING;
         DECLARE A GRAPHIC (8);
         A = GRAPHIC(X);
```

GRAPHIC (GR) — Compiler option

Specifies that DBCS is used in source.

GX — Graphic hexadecimal constant

Used to express a GRAPHIC constant using hexadecimal notation. Can be used in all places GRAPHIC string constants are allowed.

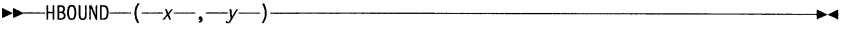



```
Example: '42C142C242C3'GX = ' < .A.B.C > 'G
```

Note: The number of hexadecimal digits must be a multiple of four.

HBOUND — Built-in function

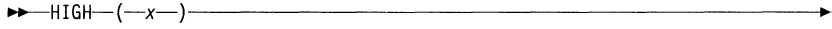
Returns current upper bound of dimension y of array x.




```
Example: SUBSTR(A,1,HBOUND(AZ,1)) = STRING(AZ);
```

HIGH — Built-in function

Returns a character string of length x, each character being the highest character of the collating sequence (that is, hexadecimal FF).



Example: IF INSTR.CDE = 999 THEN
INSTR.SEQ=HIGH (1);

HIGHINDEX — Option of ENVIRONMENT attribute

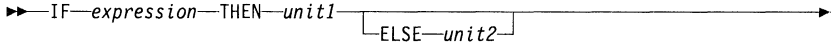
Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.

Specifies the device type on which high-level indexes reside, if this differs from the device type specified in the MEDIUM option.

Example: ENVIRONMENT(... HIGHINDEX ...)

IF — Statement

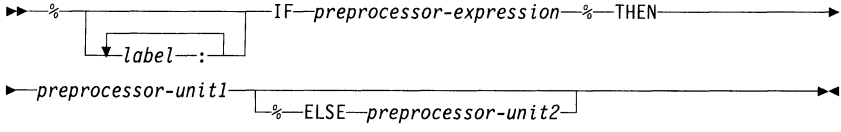
Specifies a conditional action.



Example: IF A > 15 THEN A = B;

%IF — Preprocessor statement

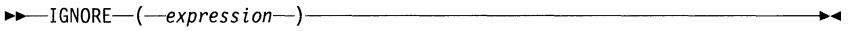
Same as IF.



Example: %IF A > 15 %THEN %A = B;

IGNORE — Option of READ statement

Ignores *expression* number of records in input file.



Example: READ FILE(AU) IGNORE(N);

IMAG — Built-in function/pseudovariable

Refers to imaginary part of complex expression x.

Example: IMAG(A1) = IMAG((A + 21) / IB);

IMPRECISE (IMP) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Allows imprecise interrupts to be handled correctly.

IN — Option of ALLOCATE and FREE statements

Specifies storage in an area variable.

See the syntax for "ALLOCATE (ALLOC) — Statement" on page 4 or "FREE — Statement" on page 40.

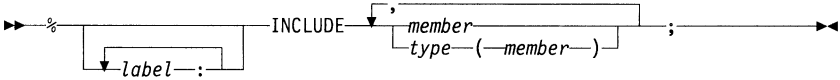
Example: ALLOCATE B SET(P) IN (A);

INCLUDE (INC) — Compiler option

Allows inclusion of secondary input without using preprocessor.

%INCLUDE — Preprocessor statement

Includes strings of external text in source program.



Example: `%INCLUDE P (ABC);`

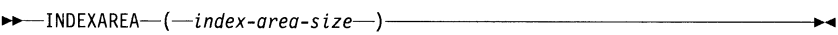
INDEX — Built-in function

Returns the starting position of the string y within string x; returns 0 if y not present in x.

Example: `IF INDEX(REC,'DUP') ¬= 0 THEN ... ;`

INDEXAREA — Option of ENVIRONMENT attribute

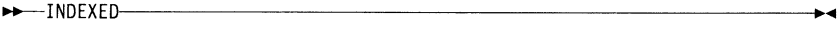
Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.



Example: `ENVIRONMENT(INDEXAREA(1000))`

INDEXED — Option of ENVIRONMENT attribute

Indicates file is used to process an indexed data set.

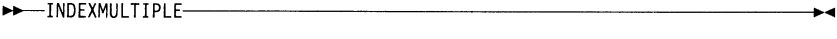


Example: `ENVIRONMENT(INDEXED)`

INDEXMULTIPLE — Option of ENVIRONMENT attribute

Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.

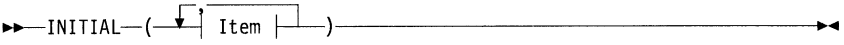
(For INDEXED data sets.) Specifies that a master index will be, or has been, built for this file.



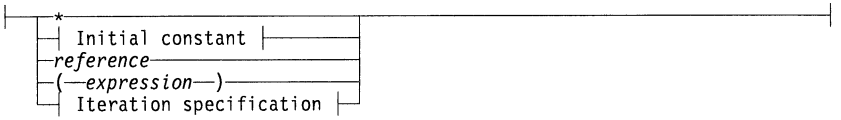
Example: `ENVIRONMENT( ... INDEXMULTIPLE ... )`

INITIAL (INIT) — Storage attribute

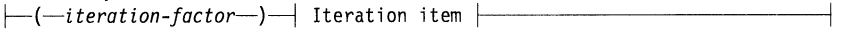
Specifies initial values for data items.



Item:

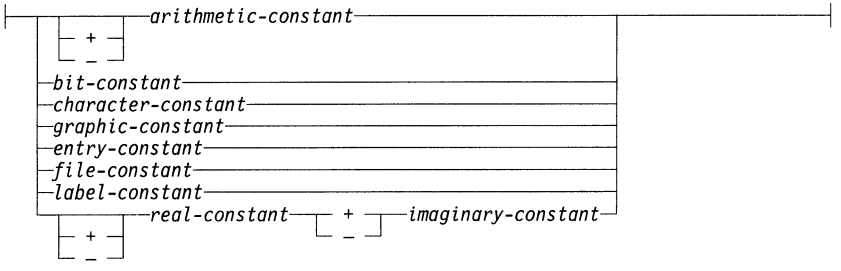


Iteration specification:

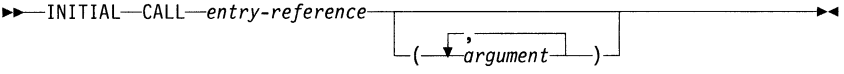


Iteration item:

Initial constant:



or




```
Example: DECLARE TAB (48) INIT((48)0);
         DECLARE X (10,10) INITIAL CALL XINIT;
```

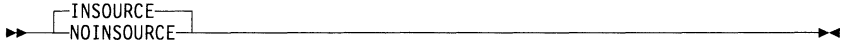
INPUT — Attribute/Option of OPEN statement

Indicates file to be used for input.

Example: DECLARE IN FILE INPUT ... ;
OPEN FILE(IN) INPUT;

INSOURCE (IS) — Compiler option

Specifies that the compiler includes a listing of the source program (including preprocessor statements) in the compiler listing.



INTER — Option of OPTIONS attribute

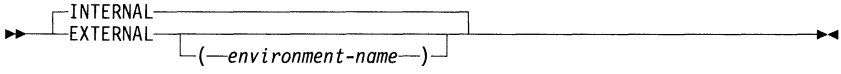
INTER is syntax-checked and ignored.

See the syntax for "OPTIONS — Attribute" on page 79.

Example: OPTIONS(COBOL INTER)

INTERNAL (INT) — Attribute

Indicates variable known only within declared block and within contained blocks.

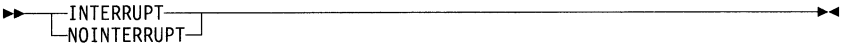


Example: DECLARE A CHARACTER(6) INT;

INTERRUPT (INT) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Specifies raising of ATTENTION condition after interrupt.



INTO — Option of READ statement

Specifies the variable to which a record is to be assigned.

Example: `READ FILE (TAX) INTO (RANGE);`

IRREDUCIBLE (IRRED) — Attribute

If used, it is checked for syntax errors and the implied attribute ENTRY is applied. Otherwise ignored.

Example: `DECLARE X ENTRY IRREDUCIBLE;`

IRREDUCIBLE (IRRED) — Option for PROCEDURE and ENTRY statements

Syntax checked and ignored.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "ENTRY — Statement" on page 33.

iSUB — Dummy variable of DEFINED attribute

Defines an array consisting of designated elements from a base array.

See the syntax for "DEFINED (DEF) — Attribute" on page 26.

Example: `DECLARE B(2,5), Y(5,2) DEF B(2SUB,1SUB);`

KEY — Condition

Raised by use of invalid key in a record I/O statement.


```
Example: ON KEY(FILE1) PUT SKIP
           EDIT('INVALID KEY:',ONKEY)(A(12));
```

KEY — Option of READ, DELETE, REWRITE, or UNLOCK statements

Identifies which record is processed.


```
Example: READ FILE(TAX) KEY('AU65') INTO(ALA);
```

KEYED — Attribute/option of OPEN statement

Specifies that one of the options KEY, KEYTO, or KEYFROM can be used in an I/O statement accessing the file.


```
Example: DECLARE INV FILE KEYED ... ;
```

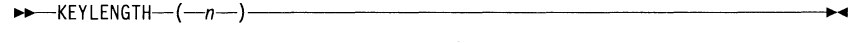
KEYFROM — Option of WRITE or LOCATE statements

Specifies the key of a new record to be written to a file.


```
Example: WRITE FILE(M) FROM(U) KEYFROM(B);
```

KEYLENGTH — Option of ENVIRONMENT attribute

Specifies length of recorded key in REGIONAL(3) data sets and VSAM KSDS.




```
Example: ENVIRONMENT(KEYLENGTH(24))
```

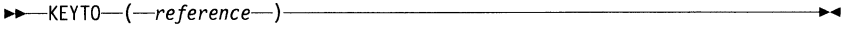
KEYLOC — Option of ENVIRONMENT attribute

Specifies start position of an embedded key in a record on an INDEXED data set.

Example: ENVIRONMENT(KEYLOC(2))

KEYTO — Option of READ statement

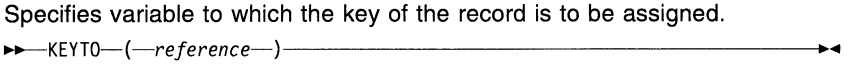
Specifies variable to which the key of the record is to be assigned.



Example: READ FILE(M) INTO(B) KEYTO(U);

KEYTO — Option of WRITE statement (VSAM ESDS and RRDS data sets only)

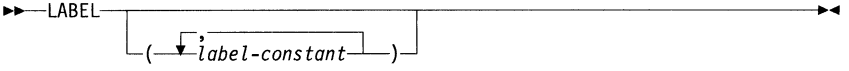
Specifies variable to which the key of the record is to be assigned.



Example: WRITE FILE(N) FROM(C) KEYTO(V);

LABEL — Attribute

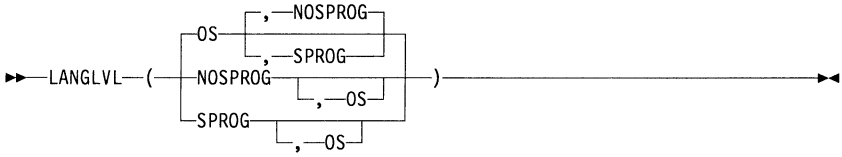
Specifies an identifier as label variable. Optimization is aided by including a list of label constants that the variable can take as values.



Example: DECLARE DOG LABEL (D1,D2);

LANGLVL — Compiler option

Defines the level of PL/I language supported, including whether pointers in operations are to be supported.



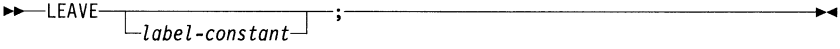
LBOUND — Built-in function

Finds current lower bound for dimension y of array x.

Example: `LIST(LBOUND(LIST,1)) = 0;`

LEAVE — Statement

Transfers control from within DO-group to statement following END statement.




```
Example: DO;
           ... LEAVE;
           ... END;
         A:DO;
           ... DO;
           ... LEAVE A;
           ... END A;
```

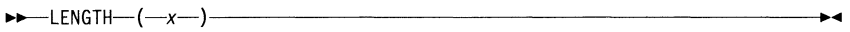
LEAVE — Option of ENVIRONMENT attribute

Specifies no rewind for tape data sets.

Example: `CLOSE FILE(OP) ENVIRONMENT(LEAVE);`

LENGTH — Built-in function

Returns current length of string x.



Example: SUBSTR(NS,2,LENGTH(S)) = S;

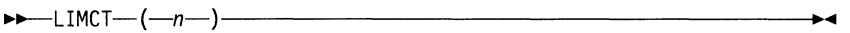
LIKE — Attribute

Specifies identical structuring.

Example: DECLARE 1 A EXTERNAL, 2 (B,C,D,) 1 E LIKE A;

LIMCT — Option of ENVIRONMENT attribute

Specifies the track search limit for REGIONAL(2) and REGIONAL(3) F-format files.



Example: ENVIRONMENT(REGIONAL(2) LIMCT(5))

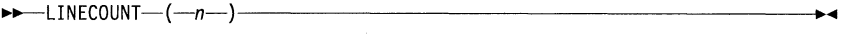
LINE — Option of PUT statement/format item

Prints on nth line.


```
Example: PUT LIST(X) LINE(7);
         PUT EDIT(A,B) (LINE(5),A(6),F(2));
```

LINECOUNT (LC) — Compiler option

Specifies the number of lines to be included in each page of the compiler listing, including heading lines and blank lines.



LINENO — Built-in function

Returns current line number of PRINT file x.


```
Example: IF LINENO(OUT) = N THEN
           PUT FILE(OUT) PAGE;
```

LINESIZE — Option of OPEN statement

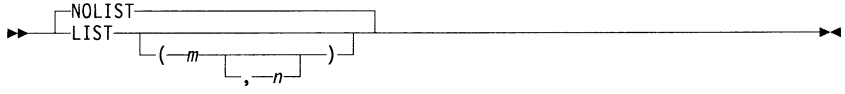
Specifies number of character positions in a line of a stream output file.

See the syntax for "OPEN — Statement" on page 77.

```
Example: OPEN FILE(SALES) LINESIZE(99);
```

LIST — Compiler option

Lists compiled code produced by compiler where *m* is the number of the first source statement for which an object listing is required, and *n* is the number of the last source statement for which an object listing is required.



LIST — Option of GET or PUT statements

Specifies list-directed transmission. Keyword can be omitted.

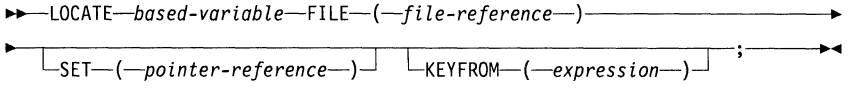

```
Example: GET (X);
           GET LIST(A,B);
           PUT FILE(WRITER) LIST(D,E);
```

LMESSAGE (LMSG) — Compiler option

Specifies long message format.

LOCATE — Statement

Causes allocation of a based variable in an output buffer and transmission of previous output buffer, if necessary.



Example: LOCATE RECORD SET(Q) FILE(PATIENT);

LOG — Built-in function

Returns logarithm of x to base e.

Example: L = LOG(A + B);

LOG2 — Built-in function

Returns logarithm of x to base 2.

Example: L = LOG2(A + B);

LOG10 — Built-in function

Returns logarithm of x to base 10.

Example: L = LOG10(A + B);

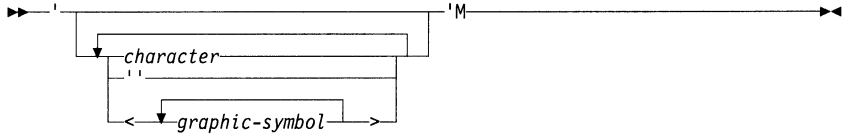
LOW — Built-in function

Returns a character string of length x, each character being the lowest character of the collating sequence (that is, hexadecimal 00).


```
Example: IF SUBSTR(RECD,1,1) = LOW(1) THEN
           GO TO ENDR;
```

M — Mixed constant

Can contain both SBCS and DBCS data.

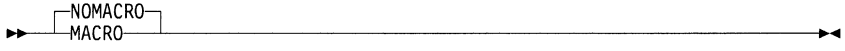


The following example shows cc as CHARACTER data and kk as non-EBCDIC DBCS character.

```
Example: MIXED = 'cc<kkkk>'M
```

MACRO (M) — Compiler option

Allows preprocessor to be used.



MAIN — Option of OPTIONS option

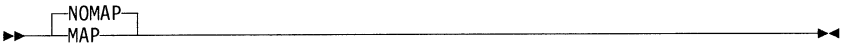
Identifies initial procedure.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78.

```
Example: FIRST:PROCEDURE OPTIONS(MAIN);
```

MAP — Compiler option

Lists offsets of variables in static control section and DSAs.



MARGINI (MI) — Compiler option

Highlights any source outside margins.

MARGINS (MAR) — Compiler option

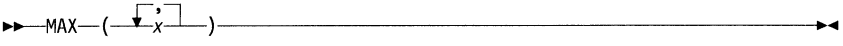
Sets the margins within which the compiler interprets the source code in your program file, and sets the position of the ANSI control character that formats listing, if SOURCE option applies.

Note: Data outside these margins is not interpreted as source code, though it is included in your source listing if you request one.

m is the column number of the left margin (first byte of source input line). It must not exceed 80. *n* is the column number of the right margin (last byte of source input line). It must exceed *m*, but must not exceed 80. *c* is the column number of the ANS printer control character (max 80 and outside values specified by *m* and *n*).

MAX — Built-in function

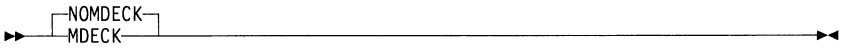
Returns the value of the highest-valued expression.



Example: `DO I = 1 TO MAX(DF,DG,DH);`

MDECK (MD) — Compiler option

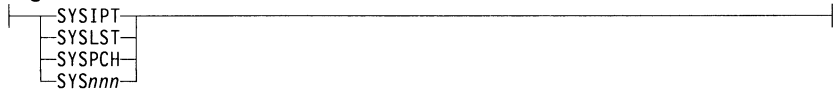
Produces a source deck from preprocessor output.



MEDIUM — Option of ENVIRONMENT attribute

Specifies the logical unit number of the device associated with the file being declared. Required for files associated with devices other than DASD.

Logical unit number:



Note:

[1] Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.

Example: `ENVIRONMENT(FB MEDIUM(SYS012))`

MIN — Built-in function

Returns the value of the lowest-valued expression.

Example:

```
DO WHILE(MIN(Y1,Y2) > 10);
```

MOD — Built-in function

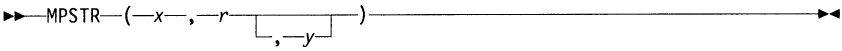
Returns the smallest positive value, R, such that: (x-R)/y=n where n is an integer. If x is positive, result is remainder from x/y.

Example:

```
SUBSTR(SED,MOD(X,256) = SEE;
```

MPSTR — Built-in function

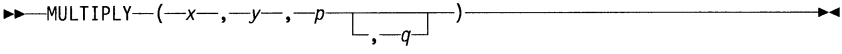
Processes and validates a mixed character string.



Example: `C = MPSTR(A,'S',15);`

MULTIPLY — Built-in function

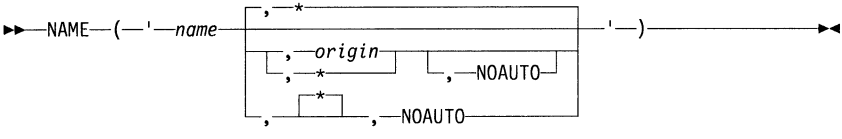
Multiplies x by y with a resulting precision given by p and scale factor q. q must be omitted if x or y is floating point.



Example: `TMT1 = MULTIPLY(TOT,NUM,31,16) / T;`

NAME (N) — Compiler option

Specifies the name of the produced object module.



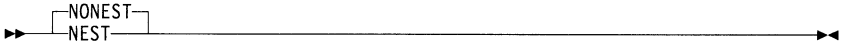
NAME — Condition

Raised during a data-directed GET FILE statement when an invalid identifier occurs in the data stream.


```
Example: ON NAME(INFILE) PUT SKIP
           EDIT('UNKNOWN:', DATAFIELD)(A);
```

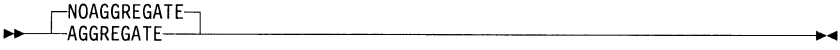
NEST — Compiler option

Indicates do-group and block level by numbering in margin.



NOAGGREGATE (NAG) — Compiler option

Specifies that a list of aggregates and their sizes is not produced.



NOATTRIBUTES (NA) — Compiler option

Specifies that the compiler does not include a table of source-program identifiers and their attributes in the compiler listing.

NOCHARGRAPHIC (NOCHARG) — Option of PROCEDURE and BEGIN

When in effect, all character assignments will be assumed to involve SBCS strings. Problems involving shift codes will not be detected.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "BEGIN — Statement" on page 9.

Example:
```
PROCEDURE(A,B) NOCHARGRAPHIC;
BEGIN NOCHARGRAPHIC;
```

NOCOMPILE (NC) — Compiler option

Specifies that the compiler does not compile the source program if errors occur.

NOCONVERSION (NOCONV) — Condition prefix

Disables the conversion condition for this statement or block.

Example: (NOCONVERSION): A = B - C;

NODECK (ND) — Compiler option

Suppresses the production of an object module in punched card format.

Note: This option can only be specified via the JCL OPTION statement (OPTION NODECK). It will be ignored if specified as a compiler option.

NOESD — Compiler option

Suppresses the listing of the external symbol dictionary.

NOEXECOPS — Option of OPTIONS option

Valid only with the MAIN procedure option. Specifies that PL/I run-time options are not specified on the invoking command or statement. Only parameters are specified.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78.

Example: `PROG: PROCEDURE OPTIONS(MAIN, NOEXECOPS);`

NOFEED — Option of ENVIRONMENT attribute

(Only for the IBM 3540 Diskette.) Specifies that the diskette is to remain in position at the end of a job step, so that it may be accessed later without operator intervention.

Example: `ENVIRONMENT(NOFEED MEDIUM(SYS015))`

NOFIXEDOVERFLOW (NOFOFL) — Condition prefix

Disables the FIXEDOVERFLOW condition for this statement or block.

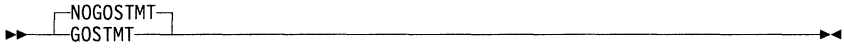
Example: `(NOFIXEDOVERFLOW): I = A + X**10;`

NOGONUMBER (NGN) — Compiler option

Specifies that the compiler is not to produce additional information that allows line numbers from the source program to be included in run-time messages.

NOGOSTMT (NGS) — Compiler option

Specifies that the compiler is not to produce additional information that allows statement numbers from the source program in run-time messages.



NOGRAPHIC (NGR) — Compiler option

Specifies that DBCS is not used in source.

NOIMPRECISE (NIMP) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

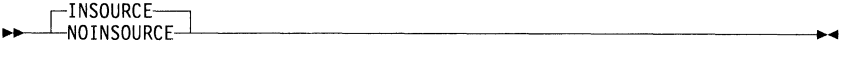
Prevents imprecise interrupts from being handled correctly.

NOINCLUDE (NINC) — Compiler option

Prevents inclusion of secondary input without using preprocessor.

NOINSOURCE (NIS) — Compiler option

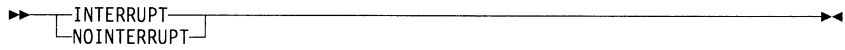
Specifies that the compiler does not include a listing of the source program (including preprocessor statements) in the compiler listing.



NOINTERRUPT (NINT) — Compiler option

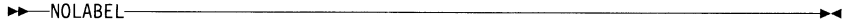
Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Suppresses raising of ATTENTION condition after interrupt.



NOLABEL — Option of ENVIRONMENT attribute

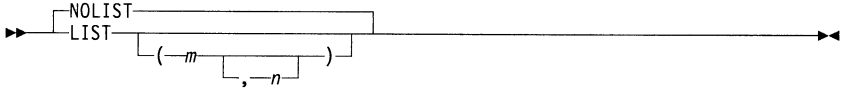
Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.



Example: ENVIRONMENT(MEDIUM(SYS016) NOLABEL VB)

NOLIST — Compiler option

Does not list compiled code produced by compiler where *m* is the number of the first source statement for which an object listing is required, and *n* is the number of the last source statement for which an object listing is required.



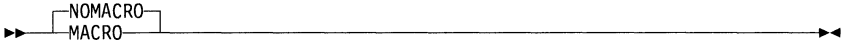
NOLOCK — Option of data transmission statements

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.


```
Example: READ FILE(STOCK) INTO(TYPES)
           KEY(KSTRING3)NOLOCK;
```

NOMACRO (NM) — Compiler option

Suppresses use of preprocessor.



NOMAP — Compiler option

Does not list offsets of variables in static control section and DSAs.

NOMAP — Option of OPTIONS option/attribute

Suppresses dummy argument creation between PL/I and COBOL on invocation, even if the mapping of an aggregate argument differs. See also ARG*n*.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78 and "OPTIONS — Attribute" on page 79.

```
Example: DECLARE COBSUB ENTRY( ... , ... , ... )
           OPTIONS(COBOL,NOMAP(ARG1,ARG3));
```

NOMAPIN — Option of OPTIONS option/attribute

Specifies that a dummy argument, if created, is not initialized. When used with NOMAPOUT, it is the same as using NOMAP. See also ARG*n*.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78 and "OPTIONS — Attribute" on page 79.

```
Example: DECLARE COBOLA OPTIONS(COBOL
           NOMAP(ARG1) NOMAPIN(ARG3));
```

NOMAPOUT — Option of OPTIONS option/attribute

Specifies that if a dummy argument is created, it is not to be assigned to the original argument on return. When used with NOMAPIN, it is the same as NOMAP. See also ARG*n*.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78 and "OPTIONS — Attribute" on page 79.

Example: `DECLARE COB OPTIONS(COBOL NOMAPOUT(ARG1,ARG4));`

NOMARGINI (NMI) — Compiler option

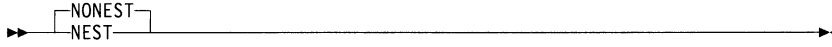
Does not highlight any source outside margins.

NOMDECK (NMD) — Compiler option

Suppresses the production of a source deck from preprocessor output.

NONEST — Compiler option

Does not indicate do-group and block level by numbering in margin.

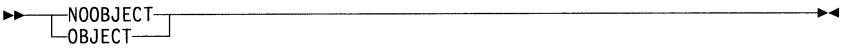


NONUMBER (NNUM) — Compiler option

Does not number statements.

NOOBJECT (NOBJ) — Compiler option

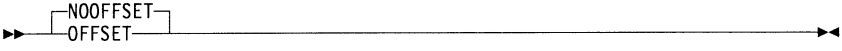
Suppresses the production of an object module from compiled output.



Note: This option will be ignored if specified as a compiler option. See "OBJECT (OBJ) — Compiler option" on page 74 for more information.

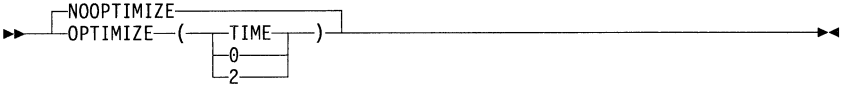
NOOFFSET (NOF) — Compiler option

Suppresses the generation of a listing associating statement numbers with offsets.



NOOPTIMIZE (NOPT) — Compiler option

NOOPTIMIZE reduces compilation time at the expense of run time.



NOOPTIONS (NOP) — Compiler option

Specifies that the compiler does not include a list in the compiler listing showing the compiler options used during this compilation.

NOOVERFLOW (NOOFL) — Condition prefix

Disables OVERFLOW condition.

Example: (NOOVERFLOW): A = X**Y;

%NOPRINT — Listing control statement

Specifies that source and insource printing is to be suspended.

Example: %NOPRINT;

NORESCAN — Option of %ACTIVATE preprocessor statement

Specifies replacement of identifier in output stream. Specifies no rescanning of text for further replacement.

See the syntax for "%ACTIVATE (%ACT) — Preprocessor statement" on page 2.

Example: %ACT I NORESCAN;

NOSEQUENCE (NSEQ) — Compiler option

Disables the SEQUENCE compiler option.

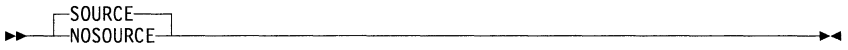
NOSIZE — Condition prefix

Disables the SIZE condition.

Example: (NOSIZE): I = A*Y;

NOSOURCE (NS) — Compiler option

Suppresses the listing of source program or preprocessor output.

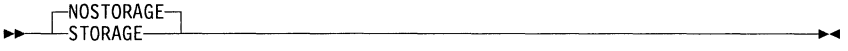


NOSTMT — Compiler option

Disables the STMT compiler option.

NOSTORAGE (NSTG) — Compiler option

Specifies that the compiler does not include in the compiler listing a table giving the main storage requirements for the object module.



NOSTRINGRANGE (NOSTRG) — Condition prefix

Disables the STRINGRANGE condition.


```
Example: (NOSTRINGRANGE): Y = SUBSTR(X,I,2*A);
```

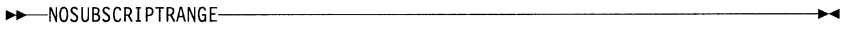
NOSTRINGSIZE (NOSTRZ) — Condition prefix

Disables the STRINGSIZE condition.


```
Example: (NOSTRINGSIZE): C1 = C2||C3||C4;
```

NOSUBSCRIPTRANGE (NOSUBRG) — Condition prefix

Disables the SUBSCRIPTRANGE condition.



Example: `(NOSUBRG): A(I) = B(J,K);`

NOSYNTAX (NSYN) — Compiler option

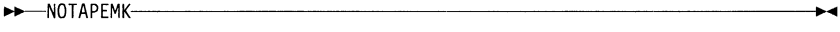
Specifies that processing stops or syntax checking is not done after an error is detected.

NOT — Compiler option

Specifies one or more alternate symbols for the logical NOT sign (¬).

NOTAPEMK — Option of ENVIRONMENT attribute

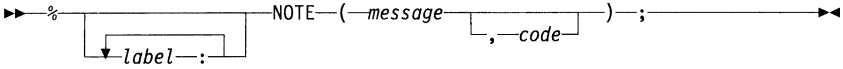
Prevents a leading tape mark from being written ahead of the data records on unlabeled tape files. The resulting data set cannot be read backward, unless it is an ASCII data set. NOTAPEMK may be used for tape OUTPUT files with NOLABEL specified. This option is not allowed for UNBUFFERED files.



Example: `ENVIRONMENT(MEDIUM(SYS017) NOLABEL NOTAPEMK FB)`

%NOTE — Preprocessor statement

Generates preprocessor diagnostic message of specified text and severity level.




```
Example: %IF PARM > 10 % THEN
           %NOTE('PARM VALUE IS'||PARM,4);
```

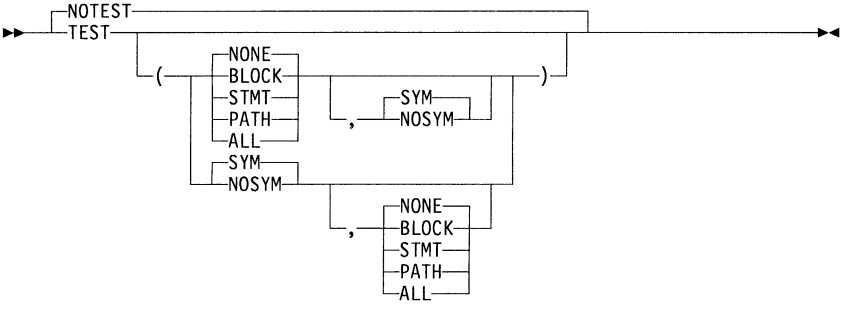
NOTERMINAL (NTERM) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Specifies that either a subset or all of the compiler listing produced during compilation does not print at the terminal. Applicable only in a conversational environment.

NOTEST — Compiler option

Suppresses the generation of testing information.



NOUNDERFLOW (NOUFL) — Condition prefix

Disables the UNDERFLOW condition.

Example: (NOUNDERFLOW): X = Y** - Z;

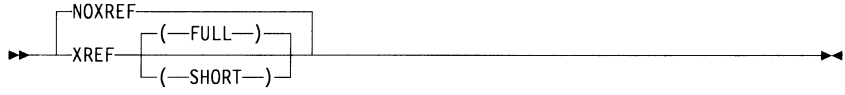
NOWRITE — Option of ENVIRONMENT attribute

Specifies that no records will be added to a DIRECT UPDATE file with INDEXED organization.

Example: ENVIRONMENT(NOWRITE)

NOXREF (NX) — Compiler option

Suppresses the generation of the compiler listing which includes both a cross-reference table of names used in the program and the numbers of the statements in which they are declared or referenced.



NOZERODIVIDE (NOZDIV) — Condition prefix

Disables the ZERODIVIDE condition.

Example: (NOZERODIVIDE): A = B / C;

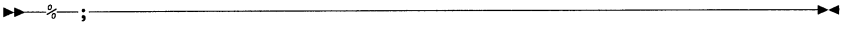
null — Statement

Does nothing and does not modify sequential statement execution.


```
Example: X = Y;
         ;
         PUT DATA(X);
```

%null — Statement

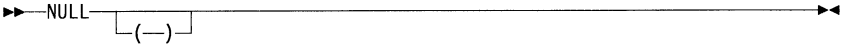
Does nothing and does not modify sequential statement execution.



Example: %;

NULL — Built-in function

Returns a null locator value.



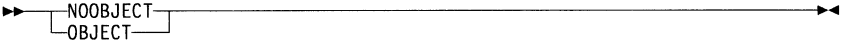
Example: P = NULL;

NUMBER (NUM) — Compiler option

Numbers statements according to line in which they start.

OBJECT (OBJ) — Compiler option

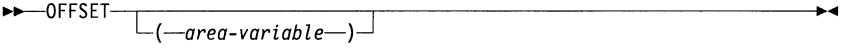
Produces an object module from compiled output.



Note: This option can only be specified via the JCL OPTION statement (OPTION CATAL). It will be ignored if specified as a compiler option.

OFFSET — Attribute

Specifies that the variable identifies a location relative to the start of an area.



Example:
```
DECLARE DATA AREA(50),
          (L1,L2) OFFSET(DATA);
```

OFFSET — Built-in function

Returns an offset value, relative to the start of area y, that identifies the generation identified by pointer x.

Example: `LOC = OFFSET(P1,TAB);`

OFFSET (OF) — Compiler option

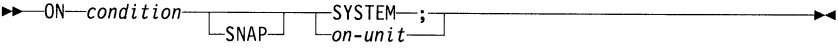
Specifies generation of a listing associating statement numbers with offsets.

OFLTRACKS — Option of ENVIRONMENT attribute

Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.

ON — Statement

Specifies action to be taken for a specified condition.




```
Example: ON FIXEDOVERFLOW BEGIN: ... ;
           END;
```

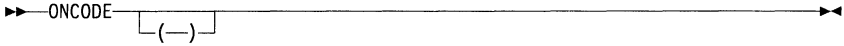
ONCHAR — Built-in function/pseudovariable

Refers to character causing CONVERSION condition.


```
Example: ON CONVERSION BEGIN;
         IF ONCHAR='A' THEN ONCHAR='1';
          ... END;
```

ONCODE — Built-in function

Identifies type of interrupt that activated the ON-unit.




```
Example: ON KEY (NEWMAST) BEGIN;
           IF ONCODE=52 THEN ... ;
           END;
```

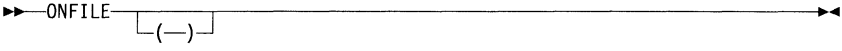
ONCOUNT — Built-in function

Specifies number of interrupts remaining to be handled.


```
Example: PUT LIST(ONCOUNT);
```

ONFILE — Built-in function

Returns name of file for which I/O condition was raised.




```
Example: ON CONVERSION BEGIN;
           IF ONFILE='INFILE' THEN ... ;
           END;
```

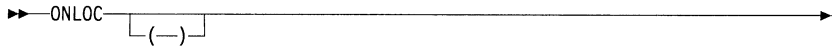
ONKEY — Built-in function

Returns a character string whose value is the key of the record that caused an I/O condition to be raised.


```
Example: ON RECORD(F) PUT LIST('ERROR IN RECORD'||ONKEY);
```

ONLOC — Built-in function

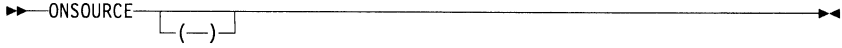
Returns name of entry point to PROCEDURE in which the ON-condition was raised.




```
Example: ON NAME BEGIN;
           IF ONLOC='SORT' THEN ... ;
           END;
```

ONSOURCE — Built-in function/pseudovariable

Refers to contents of field that was being processed when the CONVERSION condition was raised.

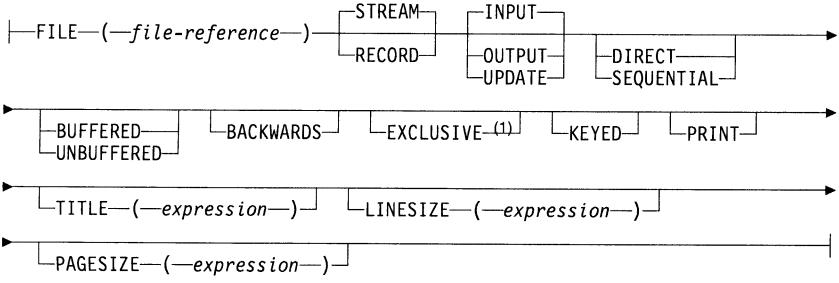



```
Example: ON CONVERSION BEGIN;
           IF ONSOURCE ¬='ABC' THEN
           PUT LIST (ONSOURCE);
           ... : END;
```

OPEN — Statement

Associates a file with a data set, and initiates the data set for access.

Options group:



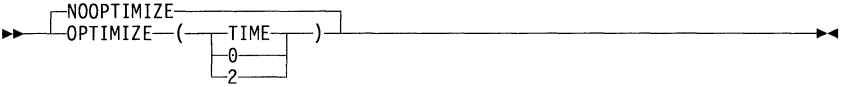
Note:

1 Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

```
Example: OPEN FILE(FILE1) INPUT, FILE(FILE2) OUTPUT;
```

OPTIMIZE (OPT) — Compiler option

OPTIMIZE reduces run time at the expense of compilation time.



OPTIONAL — Attribute

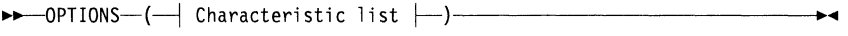
Specifies that an argument is optional and can be omitted.

Example: DCL E2 ENTRY(FIXED, OPTIONAL CHAR) EXT OPTIONS(ASM);

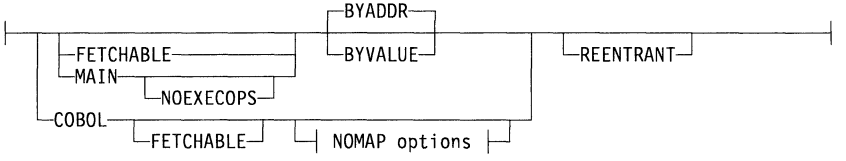
OPTIONS — Option of PROCEDURE and ENTRY statements

Specifies implementation-defined options.

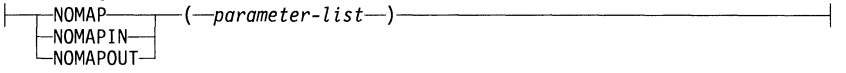
For the PROCEDURE statement:



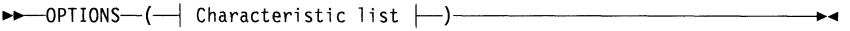
Characteristic list:



NOMAP options:

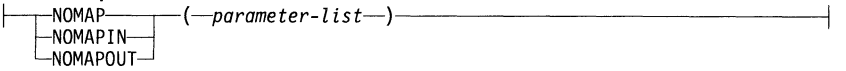


For the ENTRY statement:



Characteristic list:

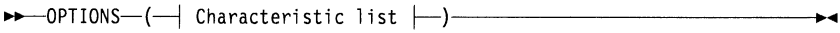
NOMAP options:



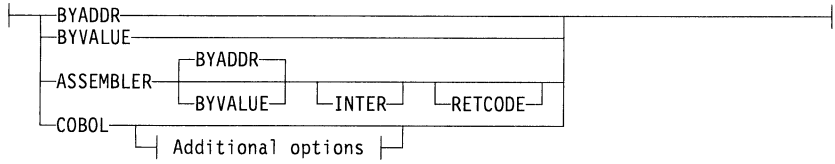
```
Example: FIRST:PROCEDURE OPTIONS(MAIN);
           CBL:ENTRY OPTIONS(COBOL);
```

OPTIONS — Attribute

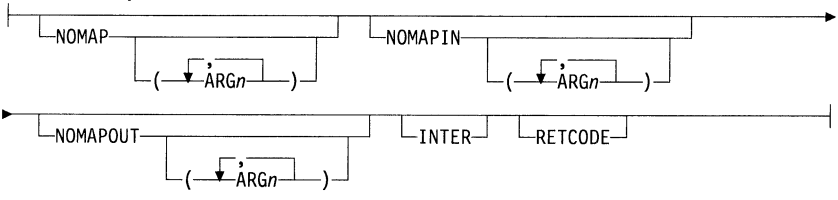
Specifies options that an entry point can have. Required if entry point is an external function or subroutine that has been compiled by a COBOL compiler.



Characteristic list:



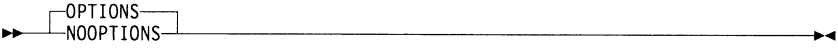
Additional options:




```
Example: DECLARE COB ENTRY OPTIONS(COBOL);
```

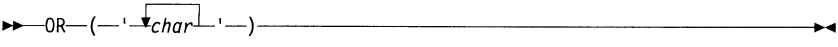
OPTIONS (OP) — Compiler option

Specifies that the compiler includes a list in the compiler listing showing the compiler options used during this compilation.



OR — Compiler option

Specifies one or more alternate symbols for the logical OR sign (|). This symbol is used as the concatenation symbol (when paired).



ORDER — Option of PROCEDURE and BEGIN statements

Specifies no relaxation of language rules to assist optimization of a block.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "BEGIN — Statement" on page 9.

```
Example: E:PROCEDURE OPTIONS(MAIN) ORDER;
```

OTHERWISE (OTHER) — Statement of select group

Specifies action to be taken when no WHEN statement is selected.

See the syntax for "SELECT — Statement" on page 96.

```
Example: SELECT (A);
           WHEN ... ;
           OTHERWISE CALL NOMATCH;
           END;
```

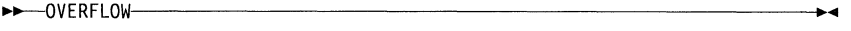
OUTPUT — Attribute/Option of OPEN statement

Indicates file is to be used for output.


```
Example: DECLARE F FILE STREAM OUTPUT;
           OPEN FILE(REPORT) OUTPUT;
```

OVERFLOW (OFL) — Condition/Condition prefix

Occurs when magnitude of floating-point number exceeds implementation limit.




```
Example: ON OFL CALL EDIT;
```

P — Format item

Describes character representation of data.

Example: PUT EDIT(PAY)(P'$$$,$$9.99');

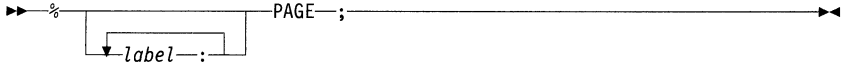
PAGE — Option of PUT statement/format item

Specifies new page with PRINT file.

Example: PUT LIST(A,B,C)PAGE;
PUT EDIT(VAL,ST)(F(8,2),PAGE,A(3));

%PAGE — Listing control statement

Causes start of new page in program listing. Applies to source listing, and, if preprocessor used, insource listing.



Example: %PAGE;

PAGESIZE — Option of OPEN statement

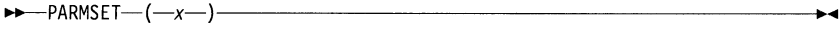
Specifies number of lines per page for a PRINT file.

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(REPORT) PAGESIZE(50);

PARMSET — Preprocessor built-in function

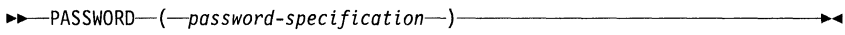
Tests whether a parameter of a preprocessor procedure was set.



Example: %A:PROCEDURE(X,Y) ... ;
IF PARMSET(X) THEN ... ;
%END;

PASSWORD — Option of ENVIRONMENT attribute

Specifies security control information required to access a VSAM data set.



Example: ... ENVIRONMENT(PASSWORD('FILELOCK'))

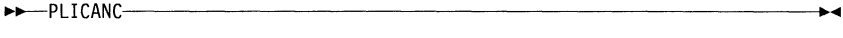
PICTURE (PIC) — Attribute

Specifies character representation of data.

Example: DECLARE FORM PICTURE'AA(3)9X99X(4)9';

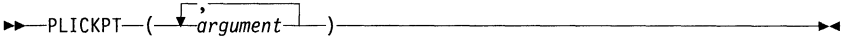
PLICANC — Built-in subroutine

Allows you to cancel automatic restart activity.



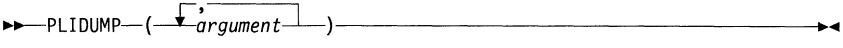
PLICKPT — Built-in subroutine

Allows you to take a checkpoint for later restart.



PLIDUMP — Built-in subroutine

Allows you to obtain a formatted dump of selected parts of storage used by your program.



PLIREST — Built-in subroutine

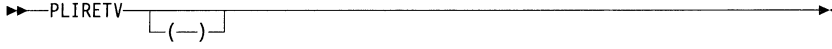
Allows you to restart program execution.

PLIRETC — Built-in subroutine

Allows you to set a return code that can be examined by the program, system, or subsystem that invoked this PL/I program or by another PL/I procedure via the PLIRETV built-in function.

PLIRETV — Built-in function

Returns value set by CALL PLIRETC statement or returned by COBOL or ASSEMBLER routine.

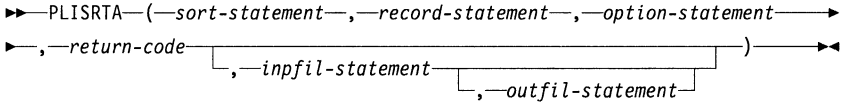


Example: `I = PLIRETV();`

See the *LE/VSE Programming Guide* for other services that can set the value returned by this built-in function.

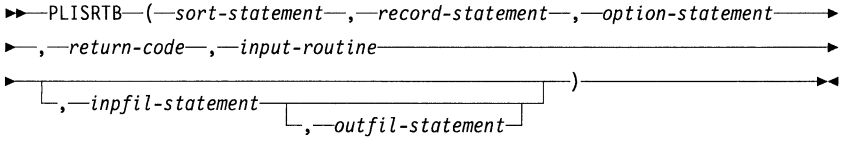
PLISRTA — Built-in subroutine

Allows you to sort an input file to produce a sorted output file.



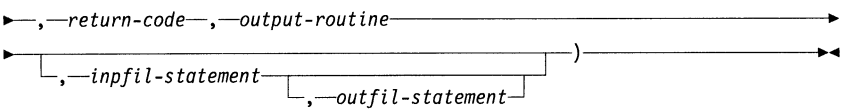
PLISRTB — Built-in subroutine

Allows you to sort input records provided by an E15 PL/I exit procedure to produce a sorted output file.



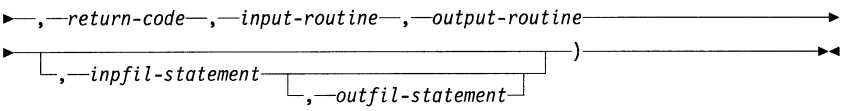
PLISRTC — Built-in subroutine

Allows you to sort an input file to produce sorted records that are processed by an E35 PL/I exit procedure.



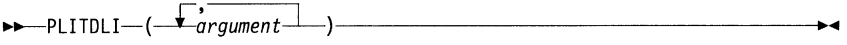
PLISRTD — Built-in subroutine

Allows you to sort input records provided by an E15 PL/I exit procedure to produce sorted records that are processed by an E35 PL/I exit procedure.



PLITDLI — Subroutine

For a description of this subroutine and its arguments, see *DL/I Application Programming.*



POINTER (PTR) — Attribute

Specifies pointer variable.

Example: DECLARE POINTER;

POINTER (PTR) — Built-in function

Returns a pointer value that identifies, in area y, a generation originally identified by the offset x.

Example: `G = POINTER(DISP,TAB);`

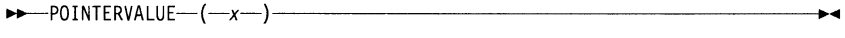
POINTERADD (PTRADD) — Built-in function

Returns a pointer value that is the sum of its expressions. x must be a pointer expression, and y must be a BIT, REAL FIXED BIN(p,0) or REAL FIXED DEC(p,0) expression. It can also be used to add and subtract.

Example: `P = POINTERADD(Q,N);`

POINTERVALUE (PTRVALUE) — Built-in function

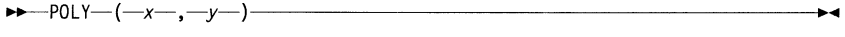
Returns a pointer value that is the converted value of x. x must be a BIT, REAL FIXED BIN(p,0) or REAL FIXED DEC(p,0) expression.



Example: `P = POINTERVALUE(N);`

POLY — Built-in function

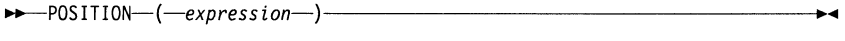
Returns the value of a polynomial derived from x and y.



Example: `A = POLY(TAB1,TAB2);`

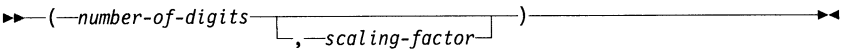
POSITION (POS) — Attribute

Specifies beginning of DEFINED string within base string.



Example: `DECLARE B CHAR(3) DEFINED C POS(3);`

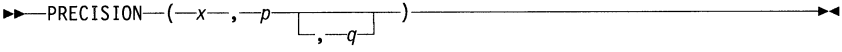
precision — Attribute



Example: DECLARE A FIXED DECIMAL(5,4);

PRECISION (PREC) — Built-in function

Converts x to a precision given by p and scale factor q. q must be omitted if x is floating point.



Example: DECLARE A FIXED DEC(5,0);
I=PREC(A/2,4,0);

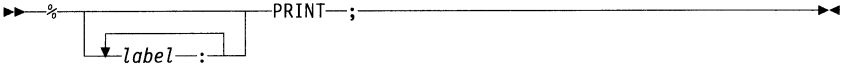
PRINT — Attribute/option of OPEN statement

Specifies that data of a file ultimately prints.

Example: DECLARE P1 FILE PRINT ... ;

%PRINT — Listing control statement

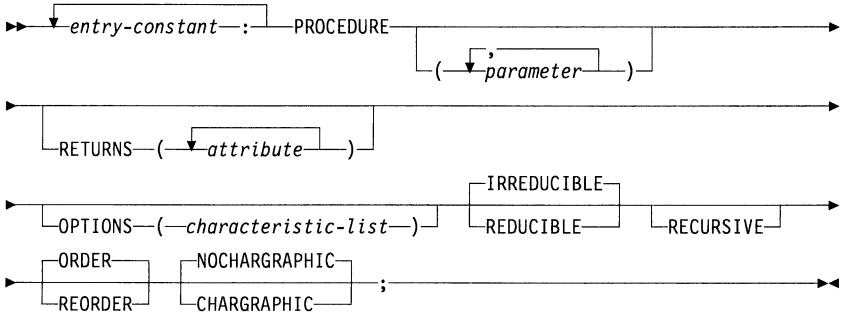
Specifies resumption of source and insource printing.



Example: %PRINT;

PROCEDURE (PROC) — Statement

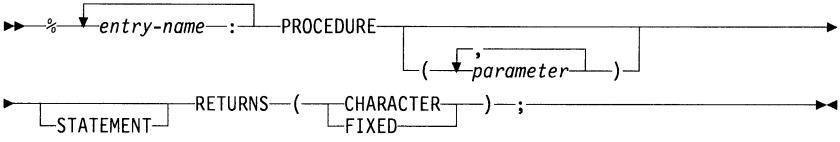
Identifies and defines primary entry points and characteristics of procedures.



Example: FIRST:PROCEDURE OPTIONS(MAIN);

%PROCEDURE (%PROC) — Preprocessor statement

Identifies a preprocessor procedure.



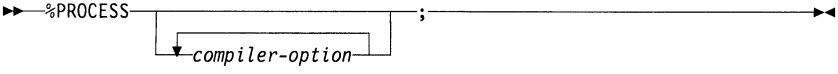

```
Example: %H:PROCEDURE(HEX) RETURNS(CHAR);
           ... ;
         %END H;
```

*PROCESS — Statement

*PROCESS is a synonym for %PROCESS (see "%PROCESS — Statement").

%PROCESS — Statement

Overrides compile-time options.



Example: %PROCESS NOCOMPILE

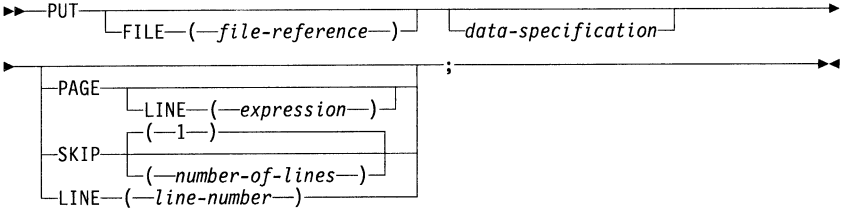
PROD — Built-in function

Returns product of all the elements of array x.

Example: PRODUCT = PROD(TABLE);

PUT — Statement

Used for stream-oriented output or for data movement in main storage.

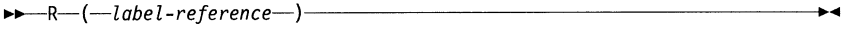


For transmission from a string:

Example: PUT DATA;
PUT LIST (A,B);
PUT STRING(STR) EDIT(S1,S2) (A);

R — Format item

Specifies the label of a FORMAT statement that is remote from a format list.



Example: F1:FORMAT(A(9),F(Y));
PUT EDIT(NAME,CODE)(R(F1));

RANGE — Option of DEFAULT statement

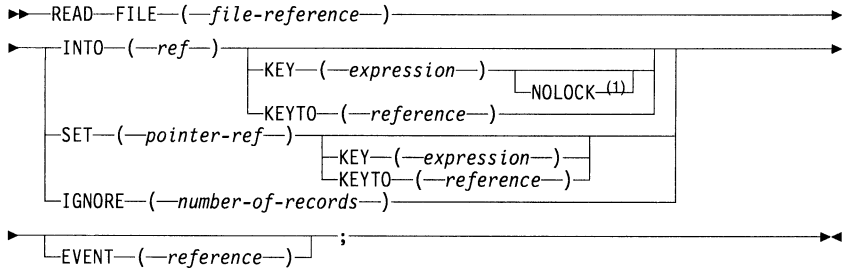
Specifies the range of initial characters of identifiers to which the DEFAULT specifications apply.

See the syntax for "DEFAULT (DFT) — Statement" on page 25.

Example: DEFAULT RANGE(X:Y) FIXED;
DEFAULT RANGE(COUNT) BIN;
DEFAULT RANGE(*) STATIC;

READ — Statement

Transfers a record from RECORD INPUT or RECORD UPDATE file to a variable or a buffer.



Note:

[1] Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

```
Example: READ FILE(FBI) INTO (EXEC);
         READ FILE(UDATE) SET (REC_P);
```

REAL — Attribute

Specifies mode of arithmetic data as real.


```
Example: DECLARE A COMPLEX, B REAL;
```

REAL — Built-in function/pseudovariable

Refers to the real part of the complex expression x.


```
Example: REAL(Q) = REAL(B + C);
```

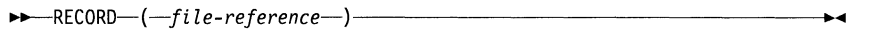
RECORD — Attribute

Specifies that data in a file is to be transmitted record by record.


```
Example: DECLARE MASTER RECORD ... ;
```

RECORD — Condition

Raised if size of record and variable do not correspond.




```
Example: ON RECORD(NEWINFO) BEGIN;
           ... END;
```

RECORD — Option of OPEN statement

Specifies attribute for file opened.

See the syntax for "OPEN — Statement" on page 77.

```
Example: OPEN FILE(ACCOUNT) RECORD UNBUF;
```

RECSIZE — Option of ENVIRONMENT attribute

Specifies record length (maximum length for U-, V-, or VB-format).


```
Example: ENVIRONMENT(RECSIZE(8))
```

RECURSIVE — Option of PROCEDURE statement

Allows procedure to be invoked recursively.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87.

```
Example: POW:PROCEDURE RECURSIVE;
           ... ;
           CALL POW;
           ... ;
           END POW;
```

REDUCIBLE (RED) — Attribute for PROCEDURE or ENTRY statement

If specified, it is checked for syntax errors, and the implied attribute ENTRY is applied. It is otherwise ignored.

```
Example: DECLARE E ENTRY REDUCIBLE;
```

REDUCIBLE (RED) — Option for PROCEDURE or ENTRY statement

Syntax checked and ignored.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "ENTRY — Statement" on page 33.

REENTRANT — Option of OPTIONS option

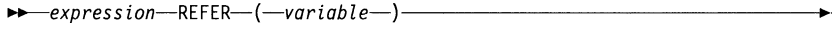
Allows a procedure to be reactivated when already active in another task.

See the syntax for "OPTIONS — Option of PROCEDURE and ENTRY statements" on page 78.

Example: `E:PROCEDURE OPTIONS(REENTRANT);`

REFER — Option of BASED attribute

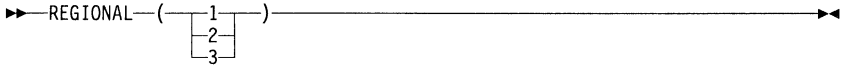
Used in a declaration of a based structure to indicate that a length, bound, or size is defined by another member of the structure.



Example: `DECLARE 1 STR BASED(P), 2 N, 2 Y (M REFER (N));`

REGIONAL — Option of ENVIRONMENT attribute

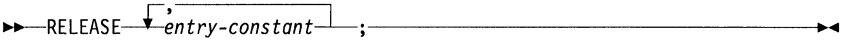
Specifies regional organization of data set on direct access device.
REGIONAL(1) has one record per region with no recorded key;
REGIONAL(2) has one record per region, with a recorded key;
REGIONAL(3) has one or more records per region, with recorded keys, and each region occupies one track.



Example: `ENVIRONMENT(REGIONAL(1))`

RELEASE — Statement

Frees main storage occupied by procedures identified by the specified entry-constants.




```
Example: PROG:  PROCEDURE;
           FETCH PROGA;
           CALL PROGA;
           RELEASE PROGA;
```

REORDER — Option of PROCEDURE and BEGIN statements

Specifies relaxation of language rules to allow optimization of block.

See the syntax for "PROCEDURE (PROC) — Statement" on page 87 or "BEGIN — Statement" on page 9.

```
Example: P:PROCEDURE REORDER;
         ... END;
```

REPEAT — Option of DO statement

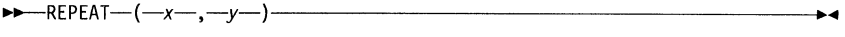
Specifies value assigned to the control variable before each repetition of the DO-group.

See the syntax for "DO — Statement" on page 28.

```
Example: DO I = 1 REPEAT 2*I ... ;  END;
           DO PTR1=HEAD REPEAT PTR1->FWD
           WHILE(PTR1¬=NULL()); ... END;
```

REPEAT — Built-in function

Returns the string x concatenated with itself y times.




```
Example: A = REPEAT(STR,2);
         /* for example, if STR='10'B then A='101010'B) */
```

REPLY — Option of DISPLAY statement

The character-string variable receives a message from the operator.

See the syntax (type 2) for "DISPLAY — Statement" on page 28.

Example: `DISPLAY('EOF') REPLY(MSG);`

REREAD — Option of ENVIRONMENT attribute

Rewinds magnetic tape to allow reprocessing of the data set.

Example: `CLOSE FILE(OP) ENVIRONMENT(REREAD);`

RESCAN — Option of %ACT preprocessor statement

Specifies that replacement strings will themselves be scanned.

See the syntax for "%ACTIVATE (%ACT) — Preprocessor statement" on page 2.

Example:
```
... %ACT I RESCAN;
... ;
```

RETCODE — Option of the OPTIONS attribute

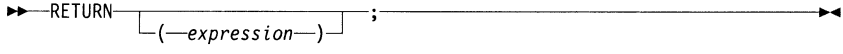
Specifies that, on return from a non-PL/I routine, the return code (in register 15) is saved. It can be accessed using the PLIRETV built-in function.

See the syntax for "OPTIONS — Attribute" on page 79.

Example: `DECLARE ASSEM OPTIONS(RETCODE)`

RETURN — Statement

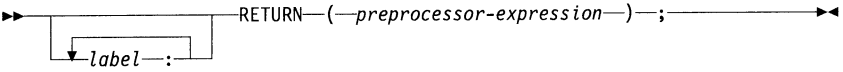
Terminates execution of its procedure. For a function procedure, an expression must be included.



Example:
```
RETURN;
RETURN(2*P+Q);
```

RETURN — Preprocessor statement

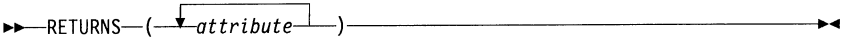
Returns a value as well as control back to the invocation point of the preprocessor procedure.




```
Example: %GEN: PROC(NAME):
         DO I=LOW TO HIGH;
         ... END;
         RETURN (STRING);
         %END;
```

RETURNS — Attribute/option of PROCEDURE or ENTRY statement

Specifies data attributes of a returned value.




```
Example: DECLARE A ENTRY RETURNS(FIXED);
         F:PROCEDURE RETURNS(FIXED);
```

REUSE — Option of ENVIRONMENT attribute

Specifies that a VSAM data set is to be used as a reusable work-file.


```
Example: ENVIRONMENT(VSAM,REUSE)
```

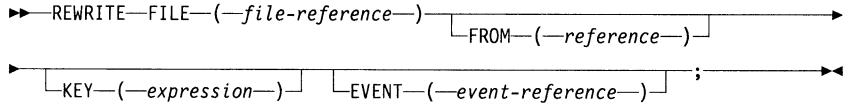
REVERT — Statement

Cancels effect of ON statement for a specified condition in the current block, and reestablishes the action specifications that existed at the activation of the block.


```
Example: REVERT ENDFILE(IN);
```

REWRITE — Statement

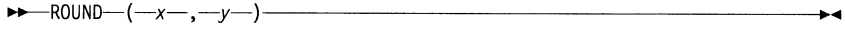
Replaces an existing record in an UPDATE file.



Example: REWRITE FILE(D2) KEY(APT) FROM(LIST);

ROUND — Built-in function

Rounds value of x on the (y)th digit to left (y negative) or right (y positive) of the decimal or binary point for a fixed-point value. For a floating-point value, the rightmost bit of the mantissa is set to one (y is ignored).



Example: CALL CTR(ROUND(A,16));

SAMEKEY — Built-in function

Returns '1'B if record accessed on a VSAM data set is followed by another record with the same key.

Example: IF SAMEKEY(INFILE) THEN ... ;

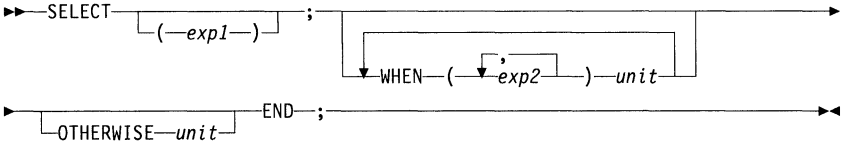
SCALARVARYING — Option of ENVIRONMENT attribute

Allows use of locate-mode I/O with element varying-length strings. The two-byte length prefix is transmitted.

Example: ENVIRONMENT(SCALARVARYING)

SELECT — Statement

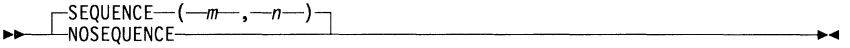
Starts a select group.




```
Example: SELECT(A + B);
           WHEN(C) ... ;
            .
            .
           OTHERWISE ... ;
           END;
```

SEQUENCE (SEQ) — Compiler option

Specifies the columns used for sequence numbers where *m* specifies the column number of the left-hand margin, and *n* specifies the column number of the right-hand margin.



SEQUENTIAL (SEQL) — Attribute

Indicates access to records of file in physical order (for CONSECUTIVE and REGIONAL data sets) or logical order (for VSAM key-sequenced data sets).


```
Example: DECLARE TEAM FILE SEQUENTIAL ... ;
```

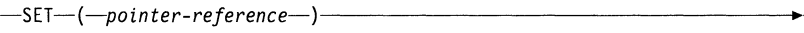
SEQUENTIAL — Option of OPEN statement

See the syntax for "OPEN — Statement" on page 77.

```
Example: OPEN FILE (INVNTRY)
           UPDATE SEQUENTIAL BUFFERED;
```

SET — Option of READ, ALLOCATE, and LOCATE statements

Specifies the locator variable that is to identify a new generation of a based variable.



Example:
```
DECLARE VAL BASED;
        ALLOCATE VAL SET (P);
```

SIGN — Built-in function

Returns sign of expression(x) as binary 1 if x>0, 0 if x=0, -1 if x<0.

Example:
```
DO X = 0 BY SIGN(A - B) TO SIGN(A - B) * 10;
```

SIGNAL — Statement

Raises the named condition.

Example:
```
SIGNAL ENDFILE(SALES);
```

SIN — Built-in function

Returns sine of x where x is in radians.

Example:
```
B = SIN(TAN(Q));
```

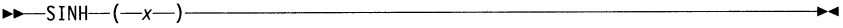
SIND — Built-in function

Returns sine of x where x is in degrees.

Example:
```
B = SIND(TAN(Q) / COS(A));
```

SINH — Built-in function

Returns hyperbolic sine of x.

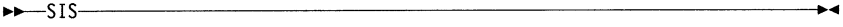


Example: `B = SINH(Q * P);`

SIS — Option of ENVIRONMENT attribute

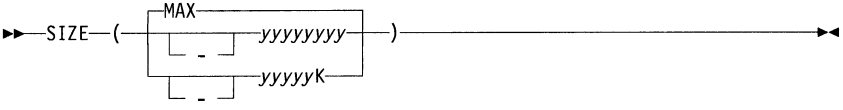
Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Applicable to key-sequenced data sets accessed using a DIRECT file.



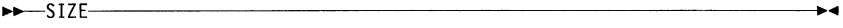
SIZE (SZ) — Compiler option

Controls the amount of main storage used by the compiler. This can be specified three ways: 1) *y* representing number of bytes, 2) *yK* being number of bytes × 1024, or 3) *MAX*, which specifies that the compiler is to obtain as much main storage as possible.



SIZE — Condition/Condition prefix

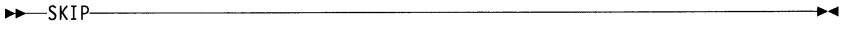
Raised when high-order digits are lost on assignment or during an arithmetic or I/O operation.



Example: `ON SIZE P = Q;`

SKIP — Option of ENVIRONMENT attribute

Specifies skip-sequential processing for a VSAM data set.



Example: ENVIRONMENT(VSAM,SKIP)

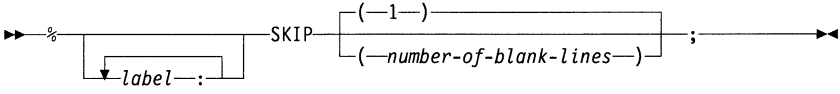
SKIP — Option of GET and PUT statement/format item

Causes transmission to continue from the start of the line *number-of-lines* after the current line. Default for *number-of-lines* is 1 (the line following the current line).

Example: PUT SKIP(2);
PUT EDIT(ST1,ST2) (A,SKIP,A);

%SKIP — Listing control statement

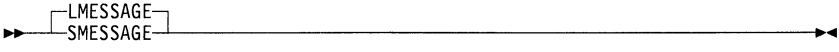
Inserts *number-of-blank-lines* in the program listing. Applies to source listing and, if preprocessor used, insource listing. Default for *number-of-blank-lines* is 1.



Example: %SKIP(2);

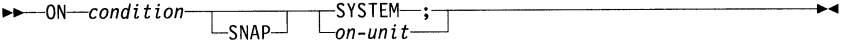
SMESSAGE (SMSG) — Compiler option

Specifies short message format.



SNAP — Option of ON statement

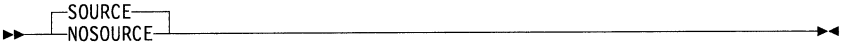
Lists all currently active blocks and statements most recently executed.



Example: `ON FIXEDOVERFLOW SNAP BEGIN ... ;`

SOURCE (S) — Compiler option

Lists source program or preprocessor output.



SQRT — Built-in function

Returns positive square root of x. (If x is complex, gives principal value.)

Example:
```
R = SQRT(M / MO + M));
```

STATEMENT (STMT) — Option of %PROCEDURE preprocessor statement

Specifies that keyword invocation of preprocessor procedure can be used.

See the syntax for "%PROCEDURE (%PROC) — Preprocessor statement" on page 87.

Example:
```
A X(4) Y(12);
   %A:PROCEDURE(X,Y) STATEMENT ... ;
```

STATIC — Attribute

Specifies storage is allocated before execution and remains allocated throughout execution.

Example: `DECLARE TABLE (6,6) STATIC;`

STATUS — Built-in function/pseudovariable

Refers to status value of event variable (x): zero for normal, nonzero for abnormal.

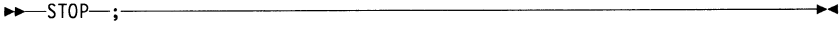
Example: `IF STATUS(T) = 0 THEN ... ;`

STMT — Compiler option

Specifies that numbering of statements in the source program are used to identify statements in the compiler listings.

STOP — Statement

Causes abnormal termination of program.



Example: `STOP;`

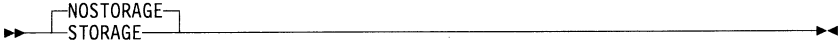
STORAGE (STG) — Built-in function

Returns the size in bytes of the storage required by the specified variable.

Example: `I = STORAGE(X);`

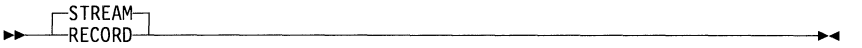
STORAGE (STG) — Compiler option

Specifies that the compiler includes in the compiler listing a table giving the main storage requirements for the object module.



STREAM — Attribute

Specifies data in a file transmits as a continuous stream of data items in character form.



Example: DECLARE MASTER STREAM ... ;

STREAM — Option of OPEN statement

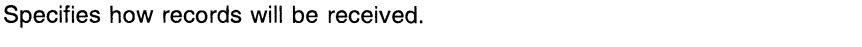
Specifies how records will be received.

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(LISTING) STREAM;

STRING — Built-in function/pseudovariable

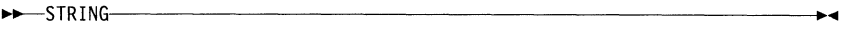
Refers to all elements of aggregate x as a single string.




```
Example: DECLARE (ST(5),CS) CHARACTER(10) VAR;
         CX=STRING(ST);
```

STRING — Option of GET and PUT statements

Data is to be formatted and (for PUT STRING) assigned to a string variable or (for GET STRING) obtained from a string expression. Data is not transmitted to a data set.



Example: GET STRING(STR) EDIT(A1,A2) (A(20),F(4,2));

STRINGRANGE (STRG) — Condition/condition prefix

Raised if values of arguments to SUBSTR reference are improper.

Example: STRG:SUBSTR(S1,I,J) = SUBSTR(S2,I,J);

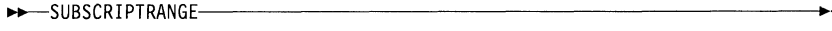
STRINGSIZE (STRZ) — Condition/condition prefix

Raised when a string is assigned to a shorter string.

Example: `(STRINGSIZE):A = B||C;`

SUBSCRIPTRANGE (SUBRG) — Condition/condition prefix

Raised when subscript value is outside dimensions of array.



Example: `ON SUBRG PUT DATA (I,J);`

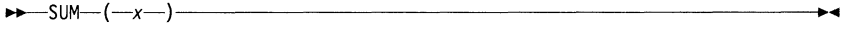
SUBSTR — Built-in function/pseudovariable

Refers to a substring of length z starting at position y of x. If z is omitted, the remainder of the string is included.

Example: `A = SUBSTR(N,4,6);`

SUM — Built-in function

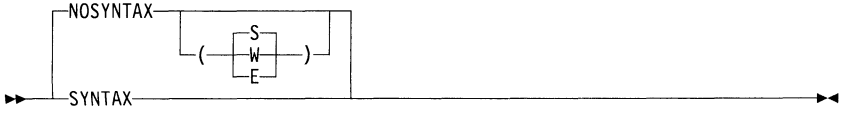
Returns sum of all elements of array x.



Example: `T = SUM(TABLE);`

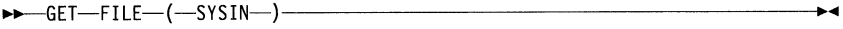
SYNTAX (SYN) — Compiler option

Specifies that the compiler continues into syntax checking after initialization (or after preprocessing if the MACRO option) unless it detects an unrecoverable error.



SYSIN — File name

Specifies standard system input file; supplied by default for GET statement.




```
Example: GET EDIT (N,M) (A(5),F(2));
         GET FILE(SYSIN) DATA(A,B);
```

SYSNULL — Built-in function

Returns the system null pointer value.


```
Example: P = SYSNULL();
```

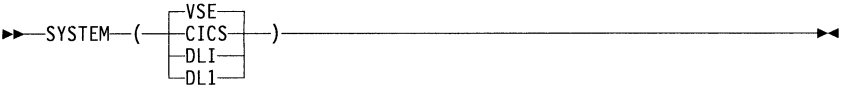
SYSPRINT — File name

Specifies standard system output file; supplied by default for PUT statement.


```
Example: PUT EDIT (N,M) (A(5),F(2));
         PUT FILE(SYSPRINT) LIST(X,Y);
```

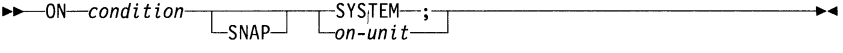
SYSTEM — Compiler option

Lets you indicate which parameter-passing convention is used when the main procedure is invoked.



SYSTEM — Option of ON statement

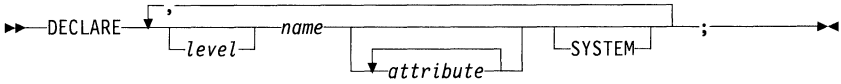
Specifies standard system action for interrupt.




```
Example: ON ERROR SNAP SYSTEM;
```

SYSTEM — Option of DECLARE statement

Specifies standard defaults are to be taken.



Example: DECLARE X(5) SYSTEM;

TAN — Built-in function

Returns tangent of x where x is in radians.

Example: A = TAN(B / C);

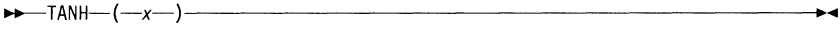
TAND — Built-in function

Returns tangent of x where x is in degrees.

Example: A = TAND(B / C);

TANH — Built-in function

Returns hyperbolic tangent of x.



Example: A = TANH(B / C);

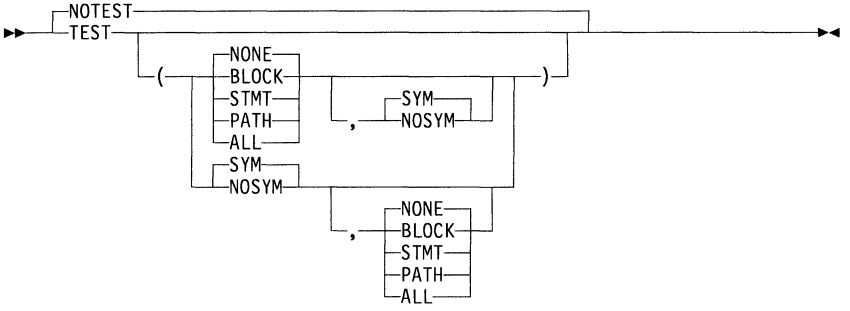
TERMINAL (TERM) — Compiler option

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Specifies that either a subset or all of the compiler listing produced during compilation is to print at the terminal. Applicable only in a conversational environment.

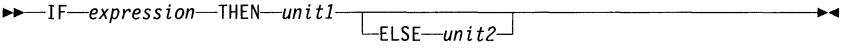
TEST — Compiler option

Specifies which debugging services are available for testing programs.



THEN — Clause of IF statement

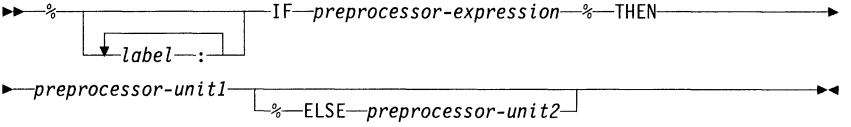
Defines action to be taken if value of expression is '1'B.



Example:
```
IF A = B THEN DO;
          ... END;
          ELSE ... ;
```

%THEN — Clause of %IF preprocessor statement

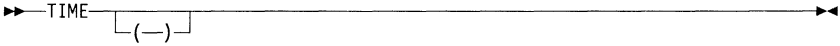
Same as THEN.



Example:
```
%IF A = B %THEN DO;
          ... END;
          %ELSE ... ;
```

TIME — Built-in function

Provides current time in character-string form, hhmmssttt for hours, minutes, seconds, milliseconds.



Example: `PUT LIST(TIME());`

TITLE — Option of OPEN statement

Identifies data set associated with a file.

See the syntax for "OPEN — Statement" on page 77.

Example: `OPEN FILE(W) TITLE('SUPER');`

TO — Option of DO statement

Specifies limit of control variable.

See the syntax for "DO — Statement" on page 28.

Example:
```
DO I = 1 TO 10;
   ... END;
```

TO — Option of repetitive specification

See the syntax for "DO — Statement" on page 28.

Example: `GET LIST(N,(A(I) DO I = 1 TO N));`

TO — Option of %DO preprocessor statement

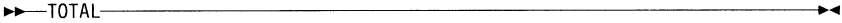
Usage is same as for the DO statement.

See the syntax for "%DO — Preprocessor statement" on page 29.

Example:
```
%DECLARE I FIXED;
%DO I = 1 TO 10;
```

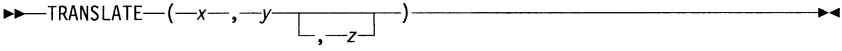
TOTAL — Option of ENVIRONMENT attribute

Aids the compiler in the production of efficient object code.



TRANSLATE — Built-in function

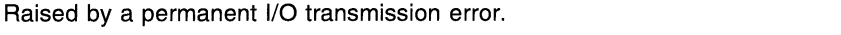
Translates source string x according to replacement string y and z. If z is omitted, string containing all possible characters (in ascending order) is assumed.



Example: `W = TRANSLATE (W,NEWCHARS,OLDCHARS);`

TRANSMIT — Condition

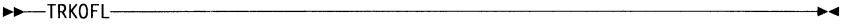
Raised by a permanent I/O transmission error.



Example: `ON TRANSMIT (PAYROLL) ... ;`

TRKOFL — Option of ENVIRONMENT attribute

Allows a record to overflow from one track to another, thus allow the size of a record to exceed the capacity of a track.




```
Example: DECLARE AFC FILE RECORD SEQUENTIAL
           INPUT ENVIRONMENT(VSAM TRKOFL):
```

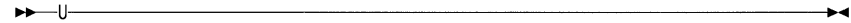
TRUNC — Built-in function

Truncates value of x to FLOOR(x) if x>0, CEIL(x) if x<0.

Example: `S = TRUNC(B);`

U — Option of ENVIRONMENT attribute

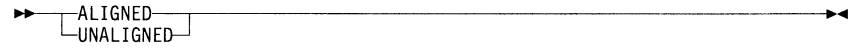
Specifies undefined-length records.



Example: ENVIRONMENT(U RECSIZE(480))

UNALIGNED (UNAL) — Attribute

Specifies mapping of data element is the next byte and independent of alignment requirement of data type.



Example: DECLARE 1 A UNAL, 2 (B CHAR(3),C);

UNBUFFERED (UNBUF) — Attribute

Specifies that records need not pass through intermediate storage.

Example: DECLARE CASH FILE UPDATE UNBUF ... ;

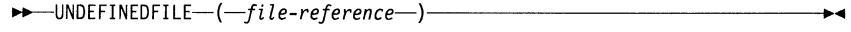
UNBUFFERED (UNBUF) — Option of OPEN statement

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(ACCOUNT) RECORD UNBUF;

UNDEFINEDFILE (UNDF) — Condition

Raised if file named cannot be opened.



Example: ON UNDF(MASTER) GO TO STOPRUN;

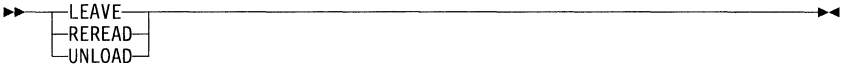
UNDERFLOW (UFL) — Condition/condition prefix

Raised when exponent of floating-point number is less than implementation minimum.

Example: ON UNDERFLOW ... ;

UNLOAD — Option of ENVIRONMENT attribute and CLOSE statement

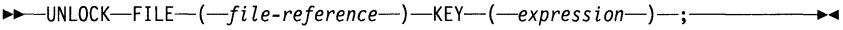
Specifies that the magnetic tape is to be rewound and unloaded when the file is closed, or (for input files) when a tape mark is read.



Example: ENVIRONMENT(F UNLOAD)

UNLOCK — Statement

Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.



Example: UNLOCK FILE (INVNTRY);

UNSPEC — Built-in function/pseudovariable

Refers to the internal code representation of the value of the expression x, as a bit string.

Example: M = UNSPEC(ITEM);

UNTIL — Option of DO statement

Specifies condition for terminating the specification. Tested at end of the DO-group.

See the syntax for "DO — Statement" on page 28.

Example: DO I = 1 REPEAT I * 2 UNTIL(I = 256);

UPDATE — Attribute

Specifies that file is used for both input and output.

Example: DECLARE F FILE UPDATE ... ;

UPDATE — Option of OPEN statement

See the syntax for "OPEN — Statement" on page 77.

Example: OPEN FILE(ACCOUNT) UPDATE BUFFERED;

V — Option of ENVIRONMENT attribute

Specifies variable-length unblocked records.

Example: ENVIRONMENT(V RECSIZE(64))

VALUE — Option of DEFAULT statement

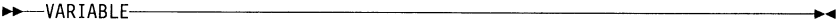
Establishes any default rules for a string length, area size, or precision.

See the syntax for "DEFAULT (DFT) — Statement" on page 25.

```
Example: DEFAULT RANGE(A:C) VALUE(FIXED
           DEC(10), FLOAT DEC(14), AREA(2000));
```

VARIABLE — Attribute

Identifies associated item to be a variable. May be specified only for file, entry, or label variables.



Example: DECLARE G FILE VARIABLE;

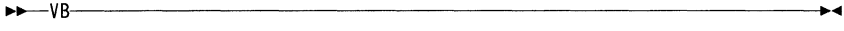
VARYING (VAR) — Attribute

Specifies string to be of varying (maximum given) length.

Example: DECLARE NAME CHAR(25) VAR;

VB — Option of ENVIRONMENT attribute

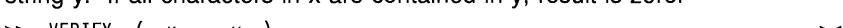
Specifies variable-length blocked records.



Example: ENVIRONMENT(VB BLKSIZE(128))

VERIFY — Built-in function

Indicates the position in string x of the first character or bit that is not in string y. If all characters in x are contained in y, result is zero.



Example: IF VERIFY(NAME,ALPHABET) THEN ... ;

VERIFY — Option of ENVIRONMENT attribute

(Only for files associated with DASDs.) Specifies that a read-check is to be performed after every write operation. This is assumed for an IBM 2321 Data Cell Drive.

Example: ENVIRONMENT (F REGIONAL(1) VERIFY)

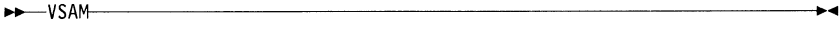
VOLSEQ — Option of ENVIRONMENT attribute

(Only for IBM 3540 Diskette input files.) Specifies that sequence checking must be performed on volume serial numbers, for multivolume data sets.

Example: ENVIRONMENT (FB VOLSEQ MEDIUM(SYS021))

VSAM — Option of ENVIRONMENT attribute

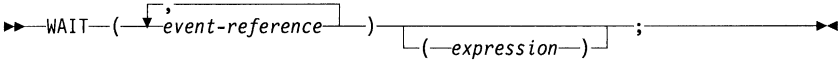
Causes VSAM input/output macros to be used for a file.



Example: ENVIRONMENT(VSAM)

WAIT — Statement

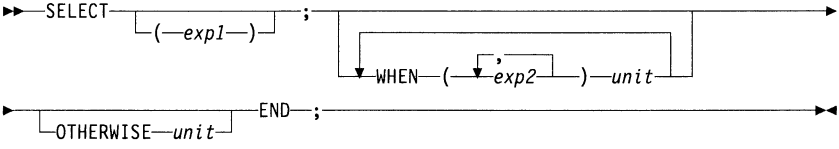
Suspends other operations until a specified number or, if no number specified, all the named events are complete.



Example: WAIT (E1,E2,E3)(2);

WHEN — Statement of select group

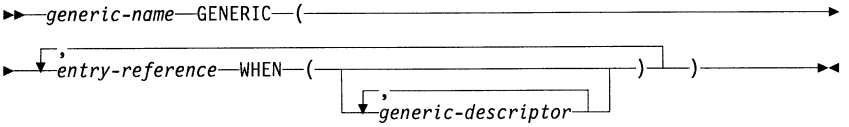
Specifies action to be taken when expression in WHEN statement equals expression in SELECT statement, or equals '1'B if SELECT expression omitted.




```
Example: SELECT(A + B);
           WHEN (C) CALL CASE1;
             .
             .
           END;
```

WHEN — Option of GENERIC attribute

Allows selection of generic entry-expression by providing descriptor list.



Example: DECLARE B GENERIC (C WHEN(FIXED),
D WHEN(FLOAT));

WHILE — Option of DO/repetitive specification

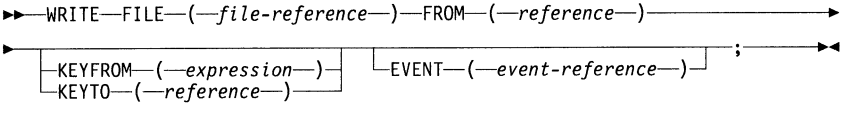
Specifies condition for execution of DO-group.

See the syntax for "DO — Statement" on page 28.

Example: DO I = 1 to 25 WHILE (X < Y);
... END;
PUT DATA((A(J) DO J=1 BY 2
(WHILE (J<N)));

WRITE — Statement

Transfers a variable from main storage to RECORD OUTPUT or UPDATE file.



Example: WRITE FILE(SALES) FROM (LOG);

WRTPROT — Option of ENVIRONMENT attribute

(Only for IBM 3540 Diskette output files.) Specifies that the data set is to be flagged as read-only.

Example: ENVIRONMENT (VB WRTPROT FILESEC MEDIUM(SYS022))

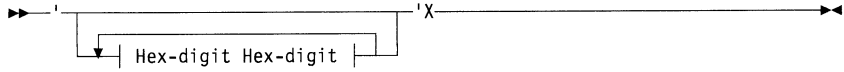
X — Format item

Specifies number of blanks that transmit on output, or number of skipped input characters.

Example: `PUT EDIT (NAME,ADDR)(A(20),X(5),A(10));`

X — Character hexadecimal constant

Used to describe a character string constant in hexadecimal notation.



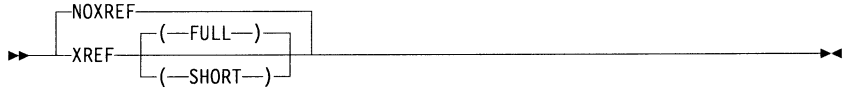
Hex-digit:

Example: `NAME = 'D7D361C9'X;`

Note: The number of hexadecimal digits must be a multiple of two.

XREF (X) — Compiler option

Specifies that the compiler listing includes both a cross-reference table of names used in the program and the numbers of the statements in which they are declared or referenced. If SHORT is specified, unreferenced names are not listed.



ZERODIVIDE (ZDIV) — Condition/condition prefix

Raised by attempt to divide by zero.

Example: `ON ZERODIVIDE SYSTEM;`

Part 2. Lists

Attributes (data elements)

ALIGNED	EVENT	OUTPUT
AREA	EXCLUSIVE[1]	PICTURE
AUTOMATIC	EXTERNAL	POINTER
BACKWARDS	FILE	POSITION
BASED	FIXED	PRINT
BINARY	FLOAT	REAL
BIT	GENERIC	RECORD
BUFFERED	GRAPHIC	REDUCIBLE
BUILTIN	INITIAL	REFER
CHARACTER	INPUT	RETURNS
COMPLEX	INTERNAL	SEQUENTIAL
CONDITION	IRREDUCIBLE	STATIC
CONNECTED	iSUB	STREAM
CONTROLLED	KEYED	UNALIGNED
DECIMAL	LABEL	UNBUFFERED
DEFINED	LIKE	UPDATE
DIRECT	OFFSET	VARIABLE
ENTRY	OPTIONAL	VARYING
ENVIRONMENT	OPTIONS	

Note:

1. Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Built-in functions, pseudovariables, subroutines

ABS
ACOS
ADD
ADDR
ALL
ALLOCATION
ANY
ASIN
ATAN
ATAND
ATANH
BINARY
BINARYVALUE
BIT
BOOL
CEIL
CHAR
COMPLETION
COMPLEX
CONJG
COS
COSD
COSH
COUNT
COUNTER
CURRENTSTORAGE
DATAFIELD
DATE
DATETIME
DECIMAL
DIM
DIVIDE
EMPTY
ENTRYADDR
ERF
ERFC
EXP
FIXED
FLOAT
FLOOR
GRAPHIC
HBOUND
HIGH
IMAG
INDEX
LBOUND
LENGTH
LINENO
LOG
LOG2
LOG10
LOW
MAX
MIN
MOD
MPSTR
MULTIPLY
NULL
OFFSET
ONCHAR
ONCODE
ONCOUNT
ONFILE
ONKEY
ONLOC
ONSOURCE
PLICANC
PLICKPT
PLIDUMP
PLIREST
PLIRETC
PLIRETV
PLISRTA
PLISRTB
PLISRTC
PLISRTD
PLITDLI
PLITEST
POINTER
POINTERADD
POINTERVALUE
POLY
PRECISION
PROD
REAL
REPEAT
ROUND
SAMEKEY
SIGN
SIN
SIND
SINH
SQRT
STATUS
STORAGE
STRING
SUBSTR
SUM
SYSNULL
TAN
TAND
TANH
TIME
TRANSLATE
TRUNC
UNSPEC
VERIFY

Characters, symbols, delimiters, operators

Alphabetic characters

Table 1. Alphabetic equivalents

Character	EBCDIC uppercase hex value	EBCDIC lowercase hex value	ASCII uppercase hex value	ASCII lowercase hex value
A	C1	81	41	61
B	C2	82	42	62
C	C3	83	43	63
D	C4	84	44	64
E	C5	85	45	65
F	C6	86	46	66
G	C7	87	47	67
H	C8	88	48	68
I	C9	89	49	69
J	D1	91	4A	6A
K	D2	92	4B	6B
L	D3	93	4C	6C
M	D4	94	4D	6D
N	D5	95	4E	6E
O	D6	96	4F	6F
P	D7	97	50	70
Q	D8	98	51	71
R	D9	99	52	72
S	E2	A2	53	73
T	E3	A3	54	74
U	E4	A4	55	75
V	E5	A5	56	76
W	E6	A6	57	77
X	E7	A7	58	78
Y	E8	A8	59	79
Z	E9	A9	5A	7A

Extralingual characters

Table 2. Extralingual equivalents

Character	Meaning	EBCDIC hex value	ASCII hex value
$	local currency symbol	5B	24
#	number sign	7B	23
@	commercial "at" sign	7C	40

Note: Code points for these symbols may vary between code pages.

Decimal digits

Table 3. Decimal digit equivalents

Character	EBCDIC hex value	ASCII hex value
0	F0	30
1	F1	31
2	F2	32
3	F3	33
4	F4	34
5	F5	35
6	F6	36
7	F7	37
8	F8	38
9	F9	39

Special characters

Table 4. Special character equivalents

Character	Meaning	Default EBCDIC hex value	Default ASCII hex value
b	Blank	40	20
=	Equal sign or assignment symbol	7E	3D
+	Plus sign	4E	2B
-	Minus sign	60	2D
*	Asterisk or multiply symbol	5C	2A
/	Slash or divide symbol	61	2F
(	Left parenthesis	4D	28
)	Right parenthesis	5D	29
,	Comma	6B	2C
.	Point or period	4B	2E
'	Single quotation mark	7D	27
%	Percent	6C	25
;	Semicolon	5E	3B
:	Colon	7A	3A
¬	Not symbol, exclusive-or symbol[1]	5F	5E
&	And symbol	50	26
|	Or symbol[1]	4F	7C
>	Greater than symbol	6E	3E
<	Less than symbol	4C	3C
_	Break character (underscore)	6D	5F
?	Question mark	6F	3F

Note:

1. The `or` (|) and the `not` (¬) symbols have variant code points and can get mistranslated. You can use the compiler options OR and NOT to define alternate symbols to represent these operators. For more information about defining symbols, see the *PL/I VSE Programming Guide.*

Composite symbols

You can combine special characters to create composite symbols. The following table describes these symbols and their meanings. Composite symbols cannot contain blanks.

Table 5. Composite symbol description

Composite symbol	Meaning
||	Concatenation
**	Exponentiation
¬<	Not less than
¬>	Not greater than
¬=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
/*	Start of a comment
*/	End of a comment
–>	Locator

Delimiters and operators

Table 6 (Page 1 of 2). Delimiters

Name	Delimiter	Use
Comment	/* */	The /* and */ enclose commentary. (This delimiter includes the /* and the */ and any characters between them.)
Comma	,	Separates elements of a list; precedes the BY NAME option
Period	.	Connects elements of a qualified name; decimal or binary point
Semicolon	;	Terminates a statement
Assignment symbol	=	Indicates assignment
Colon	:	Connects prefixes to statements; connects lower-bound to upper-bound in a dimension attribute; used in RANGE specification of DEFAULT statement
Blank	b	Separates elements

Table 6 (Page 2 of 2). Delimiters

Name	Delimiter	Use
Parentheses	( )	Enclose lists, expressions, iteration factors, and repetition factors; enclose information associated with various keywords
Locator	–>	Denotes locator qualification
Percent	%	Indicates % statements
Single quote	'	Encloses constants (indicates the beginning and end of a constant)

Note: Omitting certain symbols can cause errors that are difficult to trace. Common errors are unbalanced quotes, unmatched parentheses, unmatched comment delimiters, and missing semicolons.

Table 7. Operators

Operator type	Character(s)	Description
Arithmetic	+	Addition or prefix plus
	–	Subtraction or prefix minus
	*	Multiplication
	/	Division
	**	Exponentiation
Comparison	=	Equal to
	¬=	Not equal to
	<	Less than
	¬<	Not less than
	>	Greater than
	¬>	Not greater than
	<=	Less than or equal to
	>=	Greater than or equal to
Logical	¬	Not
	&	And
	|	Or
String	||	Concatenation

Compiler options

At installation, you can delete compiler options and modify their defaults.

AGGREGATE	NEST	NOSOURCE
ATTRIBUTES	NOAGGREGATE	NOSTMT
CMPAT[1]	NOATTRIBUTES	NOSTORAGE
COMPILE	NOCOMPILE	NOSYNTAX
CONTROL	NODECK	NOT
DECK	NOESD	NOTERMINAL[1]
ESD	NOGONUMBER	NOTEST
FLAG	NOGOSTMT	NOXREF
GONUMBER	NOGRAPHIC	NUMBER
GOSTMT	NOIMPRECISE[1]	OBJECT
GRAPHIC	NOINCLUDE	OFFSET
IMPRECISE[1]	NOINSOURCE	OPTIMIZE
INCLUDE	NOINTERRUPT[1]	OPTIONS
INSOURCE	NOLIST	OR
INTERRUPT[1]	NOMACRO	SEQUENCE
LANGLVL	NOMAP	SIZE
LINECOUNT	NOMARGINI	SMESSAGE
LIST	NOMDECK	SOURCE
LMESSAGE	NONEST	STMT
MACRO	NONUMBER	STORAGE
MAP	NOOBJECT	SYNTAX
MARGINI	NOOFFSET	SYSTEM
MARGINS	NOOPTIMIZE	TERMINAL[1]
MDECK	NOOPTIONS	TEST
NAME	NOSEQUENCE	XREF

Note:

1. Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

Conditions

AREA
ATTENTION
CONDITION
CONVERSION
ENDFILE
ENDPAGE
ERROR
FINISH
FIXEDOVERFLOW
KEY
NAME
OVERFLOW
RECORD
SIZE
STRINGRANGE
STRINGSIZE
SUBSCRIPTRANGE
TRANSMIT
UNDEFINEDFILE
UNDERFLOW
ZERODIVIDE

ENVIRONMENT attribute options

ADDBUFF[3]
ASCII
BKWD
BLKSIZE
BUFFERS
BUFFOFF
BUFND
BUFNI
BUFSP
CMDCHN
COBOL
CONSECUTIVE
CTLASA
CTL360
D
DB
DSN
EXTENTNUMBER[2]
F
FB
FBS
FILESEC
FS
GENKEY
GRAPHIC
HIGHINDEX[2]
INDEXAREA[1]
INDEXED
INDEXMULTIPLE[2]
KEYLENGTH
KEYLOC
LEAVE
MEDIUM
NOFEED
NOLABEL[2]
NOTAPEMK
NOWRITE
OFLTRACKS[1]
PASSWORD
RECSIZE
REGIONAL
REREAD
REUSE
SCALARVARYING
SIS[1]
SKIP
TOTAL
TRKOFL
U
UNLOAD
V
VB
VBS
VERIFY
VOLSEQ
VS
VSAM
WRTPROT

Notes:

1. Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.
2. Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.
3. Syntax checked only; has no effect. Kept for compatibility with previous releases of PL/I.

Format items

A
B
C
COLUMN
E
F
G
LINE
P
PAGE
R
SKIP
X

Source language keywords

A
ABS
ACOS
%ACTIVATE
ADD
ADDBUFF[3]
ADDR
ALIGNED
ALL
ALLOCATE
ALLOCATION
ANY
AREA
ARG*n*
ASCII
ASIN
ASSEMBLER
ATAN
ATAND
ATANH
AUTOMATIC
B
BACKWARDS
BASED
BEGIN
BINARY
BINARYVALUE
BIT
BKWD
BLKSIZE
BOOL
BUFFERED
BUFFERS
BUFND
BUFNI
BUFOFF
BUFSP
BUILTIN
BX
BY
B4
BYADDR
BYVALUE
BY NAME
C
CALL
CEIL
CHAR
Character
CHARACTER
CHARGRAPHIC
CLOSE
CMDCHN
COBOL
COLUMN
COMPAT
COMPILETIME
COMPLETION
COMPLEX
CONDITION
CONJG
CONNECTED
CONSECUTIVE
CONTROLLED
CONVERSION
COPY
COS
COSD
COSH
COUNT
COUNTER
CTLASA
CTL360
CURRENTSTORAGE
D
DATA
DATAFIELD
DATE
DATETIME
DB
%DEACTIVATE
DECIMAL
DECLARE
%DECLARE
DEFAULT
DEFINED
DELAY
DESCRIPTORS
DIM
DIRECT
DISPLAY
DIVIDE
DO
%DO
DSN
E
EDIT
ELSE
%ELSE
EMPTY
END
%END
ENDFILE
ENDPAGE
ENTRY
ENTRYADDR
ENVIRONMENT
ERF
ERFC
ERROR
EVENT
EXCLUSIVE[1]
EXIT
EXP
EXTENTNUMBER[2]
EXTERNAL
F
FB
FETCH
FETCHABLE
FILE
FILESEC
FINISH
FIXED
FIXEDOVERFLOW
FLOAT

FLOOR
FORMAT
FREE
FROM
G
GENERIC
GENKEY
GET
GO TO
GOTO
%GO TO
%GOTO
GRAPHIC
GX
HBOUND
HIGH
HIGHINDEX[2]
IF
%IF
IGNORE
IMAG
IN
%INCLUDE
INDEX
INDEXAREA[1]
INDEXED
INDEXMULTIPLE[2]
INITIAL
INPUT
INTER
INTERNAL
INTO
IRREDUCIBLE
iSUB
KEY
KEYED
KEYFROM
KEYLENGTH
KEYLOC
KEYTO
LABEL
LBOUND
LEAVE
LENGTH
LIKE
LIMCT
LINE
LINENO
LINESIZE
LIST
LOCATE
LOG
LOG2
LOG10
LOW
M
MAIN
MAX
MEDIUM
MIN
MOD
MPSTR
MULTIPLY
NAME
NOCHARGRAPHIC
NOCONVERSION
NOEXECOPS
NOFEED
NOFIXEDOVERFLOW
NOLABEL[2]
NOLOCK[1]
NOMAP
NOMAPIN
NOMAPOUT
NOOVERFLOW
%NOPRINT
NORESCAN
NOSIZE
NOSTRINGRANGE
NOSTRINGSIZE
NOSUBSCRIPTRANGE
%NOTE
NOUNDERFLOW
NOWRITE
NOZERODIVIDE
null
%null
NULL
OFFSET
ON
ONCHAR
ONCODE
ONCOUNT
ONFILE
ONKEY
ONLOC
ONSOURCE
OPEN
OPTIONAL
OPTIONS
ORDER
OTHERWISE
OUTPUT
OVERFLOW
P
PAGE
%PAGE
PAGESIZE
PARMSET
PASSWORD
PICTURE
PLICANC
PLICKPT
PLIDUMP
PLIREST
PLIRETC
PLIRETV
PLISRTA
PLISRTB
PLISRTC
PLISRTD
PLITDLI
PLITEST
POINTER
POINTERADD
POINTERVALUE
POLY
POSITION
precision
PRECISION
PRINT
%PRINT

PROCEDURE
%PROCEDURE
*PROCESS
%PROCESS
PROD
PUT
R
RANGE
READ
REAL
RECORD
RECSIZE
RECURSIVE
REDUCIBLE
REENTRANT
REFER
REGIONAL
RELEASE
REORDER
REPEAT
REPLY
REREAD
RESCAN
RETCODE
RETURN
RETURNS
REUSE
REVERT
REWRITE
ROUND
SAMEKEY
SCALARVARYING
SELECT
SEQUENTIAL
SET
SIGN
SIGNAL
SIN
SIND
SINH
SIS[1]
SIZE
SKIP
%SKIP
SNAP
SQRT
STATEMENT
STATIC
STATUS
STOP
STORAGE
STREAM
STRING
STRINGRANGE
STRINGSIZE
SUBSCRIPTRANGE
SUBSTR
SUM
SYSIN
SYSNULL
SYSPRINT
SYSTEM
TAN
TAND
TANH
THEN
%THEN
TIME
TITLE
TO
TOTAL
TRANSLATE
TRANSMIT
TRKOFL
TRUNC
U
UNALIGNED
UNBUFFERED
UNDEFINEDFILE
UNDERFLOW
UNLOAD
UNLOCK[1]
UNSPEC
UNTIL
UPDATE
V
VALUE
VARIABLE
VARYING
VB
VERIFY
VSAM
WAIT
WHEN
WHILE
WRITE
X
ZERODIVIDE

Notes:

1. Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.
2. Syntax checked only; has no effect. Kept for compatibility with DOS PL/I.
3. Syntax checked only; has no effect. Kept for compatibility with previous releases of PL/I.

Statements

%ACTIVATE	FREE	*PROCESS
ALLOCATE	GET	%PROCESS
BEGIN	GO TO	PUT
CALL	%GO TO	READ
CLOSE	IF	RELEASE
%DEACTIVATE	%IF	RETURN
DECLARE	%INCLUDE	REVERT
%DECLARE	LEAVE	REWRITE
DEFAULT	LOCATE	SELECT
DELAY	%NOPRINT	SIGNAL
DELETE	%NOTE	%SKIP
DISPLAY	null	STOP
DO	%null	UNLOCK[1]
%DO	ON	VERIFY
END	OPEN	VOLSEQ
%END	OTHERWISE	WAIT
ENTRY	%PAGE	WHEN
EXIT	%PRINT	WRITE
FETCH	PROCEDURE	WRTPROT
FORMAT	%PROCEDURE	

Notes:

1. Syntax checked only; has no effect. Kept for compatibility with PL/I MVS & VM.

We'd Like to Hear from You

IBM PL/I for VSE/ESA
Reference Summary
Release 1

Publication No. SX26-3836-00

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - Internet: COMMENTS@VNET.IBM.COM

  Be sure to include the following with your comments:
  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

IBM PL/I for VSE/ESA
Reference Summary
Release 1

Publication No. SX26-3836-00

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

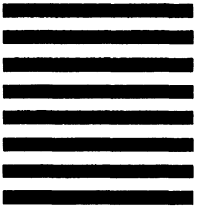
May we contact you to discuss your comments? ☐ Yes ☐ No

Name _____ Address _____

Company or Organization _____ _____

Phone No. _____ _____

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Department J58
International Business Machines Corporation
PO BOX 49023
SAN JOSE CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

Readers' Comments
SX26-3836-00

IBM®

Program Number: 5686-069

IBM PL/I for VSE/ESA Publications

GC26-8052	Fact Sheet
GC26-8055	Licensed Program Specifications
SC26-8056	Migration Guide
SC26-8057	Installation and Customization Guide
SC26-8053	Programming Guide
SC26-8054	Language Reference
SX26-3836	Reference Summary
SC26-8058	Diagnosis Guide
SC26-8059	Compile-Time Messages and Codes

SX26-3836-00