60483700

CONTROL DATA CORPORATION

PL/I VERSION 1 INSTANT

CDC® OPERATING SYSTEMS: NOS 1 NOS/BE 1

60483700

CONTROL DATA CORPORATION

PL/I VERSION 1 INSTANT

CDC® OPERATING SYSTEMS:
NOS 1
NOS/BE 1

REVISION RECORD

REVISION	DESCRIPTION
A	Original release.
(11/09/79)	
Publication No. 60483700	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV
Cover	--
Title Page	--
ii	A
iii/iv	A
v/vi	A
vii	A
viii	A
ix	A
1 thru 55	A
Back Cover	--

PAGE	REV

PREFACE

PL/I is a free-format block-structured programming language designed for scientific and commercial applications. PL/I Version 1 is a subset of the language defined by the American National Standard Programming Language PL/I, X3.53-1976, document.

PL/I Version 1 operates under control of the following operating systems:

NOS 1 for the CONTROL DATA® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems

NOS/BE 1 for the CDC® CYBER 170 Series; CYBER 70 Models 71, 72, 73, 74; and 6000 Series Computer Systems

This instant provides a brief description of the major PL/I language features. It is intended for programmers familiar with PL/I.

More detailed information can be found in the publications listed below.

Publication	Publication Number
CYBER Record Manager Advanced Access Methods Version 2 Reference Manual	60499300
CYBER Record Manager Basic Access Methods Version 1.5 Reference Manual	60495700
FORTRAN Common Library Mathematical Routines Reference Manual	60498200
NOS Version 1 Reference Manual, Volume 1 of 2	60435400
NOS/BE Version 1 Reference Manual	60493800
PL/I Version 1 Reference Manual	60388100

CDC manuals can be ordered from Control Data Corporation, Literature and Distribution Services, 308 North Dale Street, St. Paul, Minnesota 55103.

CONTENTS

NOTATIONS

UPPER CASE	words are PL/I keywords.
Lower case	words are generic terms that represent the words or symbols supplied by the programmer.
Brackets [ ]	enclose optional portions of syntax. All of the syntax within the brackets can be omitted or included at programmer option. If items are stacked vertically within brackets, only one of the stacked items can be used.
Braces { }	enclose required portions of syntax. If items are stacked vertically within braces, only one of the stacked items can be used.
Vertical Bars Within Brackets [| |]	enclose two or more vertically stacked items when each of the stacked items can be either used once, or omitted. Any items used can be written in any order.
Ellipses . . .	immediately follow an item, a pair of brackets, or a pair of braces to indicate that the item or enclosed syntax can be repeated at programmer option.
Commas , , ,	immediately follow an item, a pair of brackets, or a pair of braces to indicate that the item or enclosed syntax can be repeated at programmer option. A comma is required between repeated entries.
Bullet •	separates adjacent items to indicate they can be written in any order. Two or more bullets separating items at the same bracket or brace level indicate all separated items can be permuted in any order.

Punctuation symbols shown within syntax are required unless enclosed in brackets.

LANGUAGE ELEMENTS

PL/I is a free-format block-structured programming language that can be used for both scientific and business applications. This section describes the components of a PL/I program.

BLOCKS AND GROUPS

A block is a segment of code that delimits the scope of names, determines storage allocation, and influences the flow of control. Procedure blocks are delimited by PROCEDURE and END statements; begin blocks are delimited by BEGIN and END statements; do-groups are delimited by DO and END statements.

STATEMENTS

Statements consist of keywords and programmer-supplied elements. They declare names and describe the operations to be performed at run time.

[prefix]...[statement body];

IDENTIFIERS

Identifiers reference and identify data, statements, and blocks. They consist of 1 to 40 letters, digits, or special characters @, #, $, or _. Identifiers must begin with a letter, @, #, or $.

KEYWORDS

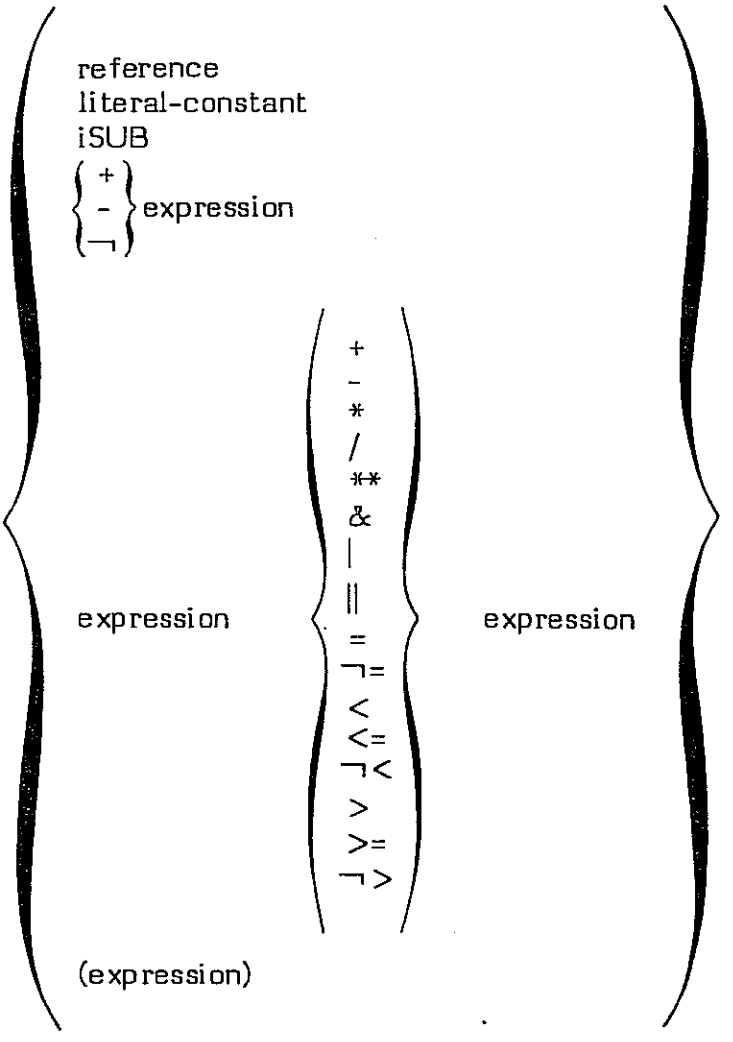
Keywords are identifiers that have special meaning when used in a particular context. PL/I keywords are not reserved words.

COMMENTS

Comments document the source code. Comments begin with /* and end with */.

EXPRESSIONS

An expression is a series of operands and operators that represent the rules for computing a value.



where reference can be one of:

simple-reference [([argument,,,])]

{structure-qualifier.}... member-reference

locator-reference ->based-reference

identifier

identifier(subscript,,,)

DECLARATIONS

Declarations establish a set of attributes for each identifier used in a program. Attributes describe and define data elements.

SCOPE OF DECLARATIONS

Identifiers with the INTERNAL attribute are known in the block that immediately contains the declaration. They are also known in those contained blocks that do not contain another declaration for the same identifier.

Identifiers with the EXTERNAL attribute are known in all blocks except those that contain a declaration of the same identifier with the INTERNAL attribute.

DECLARE

The DECLARE statement explicitly declares identifiers in a program. DECLARE statements are processed at compile time.

$$[\text{label:}]\ldots\ \begin{Bmatrix}\text{DECLARE}\\ \text{DCL}\end{Bmatrix}\ \text{declaration},,,;$$

where declaration is

$$[\text{level}]\ \begin{Bmatrix}\text{identifier}\\ (\text{declaration},,,)\end{Bmatrix}[\text{dimension-suffix}]\ [\text{attribute}]\ldots$$

ATTRIBUTES

Attributes describe identifiers. Table 1 summarizes the attributes. If certain attributes are omitted from the DECLARE statement, default attributes are assigned by the compiler. The standard defaults are listed in table 2. The INRULE parameter on the PLI control statement can change the defaults.

TABLE 1. SUMMARY OF ATTRIBUTES

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
ALIGNED	Alignment	variable parameter descriptor returns descriptor	UNALIGNED
AREA(size)	Noncomputational data type	variable parameter descriptor returns descriptor	any other data type structure
AUTOMATIC or AUTO	Storage type	variable	any other storage type EXTERNAL member
BASED	Storage type	variable	any other storage type EXTERNAL member

BINARY or BIN	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic DECIMAL structure
BIT(length)	String data type	variable parameter descriptor returns descriptor literal constant	any data type other than string CHARACTER structure
BUILTIN	None	identifier used as builtin function name	condition constant variable EXTERNAL
CHARACTER(length) or CHAR(length)	String data type	variable parameter descriptor returns descriptor literal constant	any data type other than string BIT structure

TABLE 1. SUMMARY OF ATTRIBUTES (Contd)

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
condition	None	identifier used in programmer-named condition	BUILTIN constant variable INTERNAL
constant	None	identifier used as named constant literal constant	BUILTIN condition variable
CONTROLLED or CTL	Storage type	variable	any other storage type member
DECIMAL or DEC	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic BINARY structure

DEFINED or DEF	Storage type	variable	any other storage type EXTERNAL member INITIAL VARYING
dimension	Aggregation	variable label constant parameter descriptor	ENTRY format FILE
DIRECT	File description	file constant	STREAM PRINT SEQUENTIAL
ENTRY	Noncomputational data type	variable named constant parameter descriptor	any data type other than RETURNS dimension structure any storage type other than parameter member
ENVIRONMENT or ENV	File description	file constant	None

TABLE 1. SUMMARY OF ATTRIBUTES (Contd)

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
EXTERNAL or EXT	Scope	variable named constant identifier used in programmer-named condition	BUILTIN INTERNAL LABEL format member AUTOMATIC BASED DEFINED parameter
FILE	Noncomputational data type	variable named constant parameter descriptor	any other data type any storage type other than parameter dimension structure member

FIXED	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic FLOAT structure
FLOAT	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic FIXED precision (p,q) structure
format	Noncomputational data type	named constant	any other data type dimension
INITIAL	Initialization	variable	parameter DEFINED structure ENTRY FILE format LABEL with STATIC

TABLE 1. SUMMARY OF ATTRIBUTES (Contd)

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
INPUT	File description	file constant	OUTPUT PRINT UPDATE
INTERNAL or INT	Scope	variable named constant builtin function	condition EXTERNAL
KEYED	File description	file constant	STREAM PRINT
LABEL	Noncomputational data type	variable named constant parameter descriptor	any other data type STATIC with INITIAL structure
LIKE	None	None	any data type INITIAL

member	Aggregation	variable parameter descriptor	EXTERNAL any storage type
nonvarying	String data type	variable parameter descriptor returns descriptor literal constant	any data type other than string VARYING structure
OFFSET	Noncomputational data type	variable parameter descriptor returns descriptor	any other data type structure
OUTPUT	File description	file constant	INPUT UPDATE
parameter	Storage type	variable	any other storage type member INITIAL
PICTURE or PIC	Pictured data type	variable parameter descriptor returns descriptor	any other data type structure

TABLE 1. SUMMARY OF ATTRIBUTES (Contd)

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
POINTER or PTR	Noncomputational data type	variable parameter descriptor returns descriptor	any other data type structure
POSITION(start-pos) or POS(start-pos)	None	variable (for string overlay defining only)	any storage type except DEFINED any data type other than string and pictured EXTERNAL ALIGNED member VARYING INITIAL
precision	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic structure

PRINT	File description	file constant	INPUT UPDATE RECORD SEQUENTIAL DIRECT KEYED
REAL	Arithmetic data type	variable parameter descriptor returns descriptor literal constant	any data type other than arithmetic structure
RECORD	File description	file constant	STREAM PRINT
RETURNS	Noncomputational data type	variable named constant parameter descriptor	any data type other than ENTRY structure member
SEQUENTIAL or SEQL	File description	file constant	STREAM PRINT

TABLE 1. SUMMARY OF ATTRIBUTES (Contd)

Attribute:	Belongs to the following category of attributes:	Can be in the completed attribute set for:	And conflicts with the following attributes:
STATIC	Storage type	variable	any other storage type member LABEL with INITIAL
STREAM	File description	file constant	RECORD UPDATE SEQUENTIAL DIRECT KEYED
structure	Aggregation	variable parameter descriptor	any data type INITIAL
UNALIGNED	Alignment	variable parameter descriptor returns descriptor	ALIGNED

UPDATE	File description	file constant	INPUT OUTPUT STREAM PRINT
variable	None	identifier used as variable	BUILTIN condition constant
VARYING	String data type	variable parameter descriptor returns descriptor	any data type other than string nonvarying DEFINED structure

TABLE 2. SUMMARY OF DEFAULT ATTRIBUTES FOR DECLARATIONS

An identifier with this attribute:	Will necessarily acquire these attributes:	And will acquire these attributes unless contradicted:
ALIGNED	variable	None
AREA with or without (size)	variable	ALIGNED† size = 150 words†
AUTOMATIC or AUTO	variable INTERNAL	None
BASED	variable INTERNAL	None
BINARY or BIN	variable REAL†	scale FIXED or FLOAT† precision†
BIT with or without (length)	variable	nonvarying† UNALIGNED† length = 1 bit†
BUILTIN	INTERNAL	None
CHARACTER or CHAR with or without (length)	variable	nonvarying† UNALIGNED† length = 1 character†

condition	EXTERNAL	None
constant	None	None
CONTROLLED or CTL	variable	INTERNAL
DECIMAL or DEC	variable REAL†	scale FIXED or FLOAT† precision†
DEFINED or DEF	variable INTERNAL	None
dimension	None	None
DIRECT	FILE constant RECORD (open time) KEYED (open time)	INPUT (open time)
ENTRY	variable (if parameter) constant (if not parameter)	ALIGNED (if parameter)†† EXTERNAL (if constant)
ENVIRONMENT or ENV	None	None
EXTERNAL or EXT	None	STATIC (if variable)
FILE	variable (if parameter) constant (if not parameter)	ALIGNED (if parameter)†† EXTERNAL (if constant) STREAM (open time) INPUT (open time)
FIXED	variable REAL†	base BINARY or DECIMAL† precision†

TABLE 2. SUMMARY OF DEFAULT ATTRIBUTES FOR DECLARATIONS (Contd)

An identifier with this attribute:	Will necessarily acquire these attributes:	And will acquire these attributes unless contradicted:
FLOAT	variable REAL	base BINARY or DECIMAL precision
format	constant INTERNAL	None
INITIAL	None	None
INPUT	FILE constant	STREAM (open time)
INTERNAL or INT	None	AUTOMATIC (if variable)
KEYED	FILE constant RECORD (open time)	SEQUENTIAL (open time) INPUT (open time)
LABEL	INTERNAL (if constant)	variable ALIGNED (if variable)†† INTERNAL (if variable)
LIKE	None	None
member	None	None
nonvarying	None	None
OFFSET	variable	ALIGNED†

OUTPUT	FILE constant	STREAM (open time)
parameter	variable INTERNAL	None
PICTURE or PIC with 'picture-specification'	variable	INTERNAL AUTOMATIC UNALIGNED†
POINTER or PTR	variable	ALIGNED†
POSITION or POS with or without (start-pos)	None	start-pos = 1
precision	REAL†	(15,0) for FIXED BINARY (48) for FLOAT BINARY (5,0) for FIXED DECIMAL (14) for FLOAT DECIMAL
PRINT	FILE constant STREAM (open time) OUTPUT (open time)	None
REAL	variable	scale FIXED or FLOAT base BINARY or DECIMAL precision
RECORD	FILE constant	SEQUENTIAL (open time) INPUT (open time)
RETURNS	ENTRY	None

TABLE 2. SUMMARY OF DEFAULT ATTRIBUTES FOR DECLARATIONS (Contd)

An identifier with this attribute:	Will necessarily acquire these attributes:	And will acquire these attributes unless contradicted:
SEQUENTIAL or SEQL	FILE constant RECORD (open time)	INPUT (open time)
STATIC	variable	INTERNAL
STREAM	FILE constant	INPUT (open time)
structure	variable	INTERNAL AUTOMATIC
UNALIGNED	variable	None
UPDATE	FILE constant RECORD (open time)	None
variable	None	INTERNAL AUTOMATIC
VARYING	None	None

†Also effective for any parameter descriptor or returns descriptor.
††Also effective for any parameter descriptor.

STATEMENT PREFIXES

Statement prefixes precede the statement body and are separated from the statement with a colon.

CONDITION PREFIX

A condition prefix can be used with any statement except END, ENTRY, and DECLARE. The condition prefix enables or disables program interrupts that result when certain conditions are raised.

$$\left[\left(\left\{\begin{array}{l}\text{enabled-condition-name}\\ \text{disabled-condition-name}\end{array}\right\},,,\right):\right]\ \ldots$$

ENTRY PREFIX

An entry prefix must be used with each ENTRY or PROCEDURE statement. An entry prefix provides an entry point at which a procedure can be invoked.

entry-name : [entry-name:] ...

FORMAT PREFIX

A format prefix must be used with each FORMAT statement. A format prefix references a FORMAT statement and executes it during edit-directed stream I/O.

format-name : [format-name:] ...

LABEL PREFIX

A label prefix can be used with any statement except ENTRY, PROCEDURE, FORMAT, and the first statement of an on-unit. A label prefix references a source statement.

[label [(subscript,,,)] :] ...

PROGRAM CONTROL STATEMENTS

Program control statements define the structure of a PL/I program and control the execution of program blocks.

BEGIN

The BEGIN statement denotes the start of a begin block. Condition prefixes used on the BEGIN statement enable or disable conditions for the begin block.

[prefix]... BEGIN ;

CALL

The CALL statement invokes a procedure and specifies arguments to be associated with the formal parameters.

[prefix]... CALL subroutine-reference [([argument,,,])] ;

DO

The DO statement executes a do-group until a specified condition occurs. It has three formats.

Noniterative DO

[prefix]... DO ;

DO WHILE

[prefix]... DO WHILE(expression) ;

Indexed DO

[prefix]... DO index = do-specification,,, ;

where do-specification is

start-expression [[TO expression / BY expression]]

[WHILE(expression)]

END

The END statement denotes the end of a procedure block, begin block, or do-group.

```
[label:]... END [closure-name];
```

ENTRY

The ENTRY statement denotes a secondary entry point of a procedure block. It can specify formal parameters and the attributes of the returned value.

```
{entry-name:}... ENTRY [(parameter,,,)]
    [RETURNS [(returns-descriptor)]] ;
```

GOTO

The GOTO statement transfers control to the statement with the specified label prefix.

```
[prefix]... {GOTO  } label-reference ;
            {GO TO }
```

NULL

The null statement has no effect on program execution.

```
[prefix]... ;
```

PROCEDURE

The PROCEDURE statement denotes the beginning of a procedure block. It can include formal parameters to be used in the procedure block, and the attributes of the value to be returned.

```
[condition-prefix]... {entry-name:}... {PROCEDURE}
                                       {PROC     }
   [ [(parameter,,,)] [ ||RETURNS [(returns-descriptor)]|| ] ;
                      [ ||RECURSIVE                      || ]
   [ OPTIONS(MAIN) [RECURSIVE] ]
```

RETURN

The RETURN statement terminates a procedure or on-unit activation and returns control to where the procedure was invoked or condition was raised. It can specify a value to be returned by a procedure.

[prefix]... RETURN [(return-value)] ;

STOP

The STOP statement raises the FINISH condition. The normal termination of the FINISH on-unit is to terminate program execution.

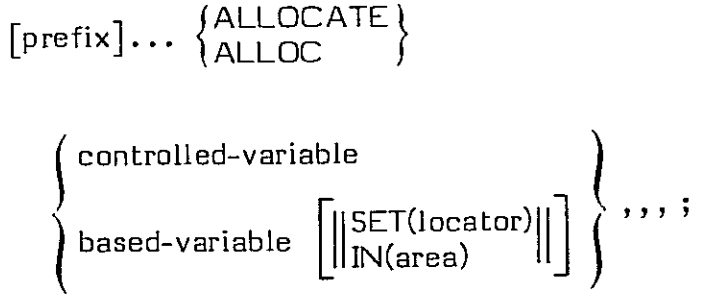
[prefix]... STOP ;

STORAGE CONTROL STATEMENTS

Storage control statements allocate and free storage during program execution.

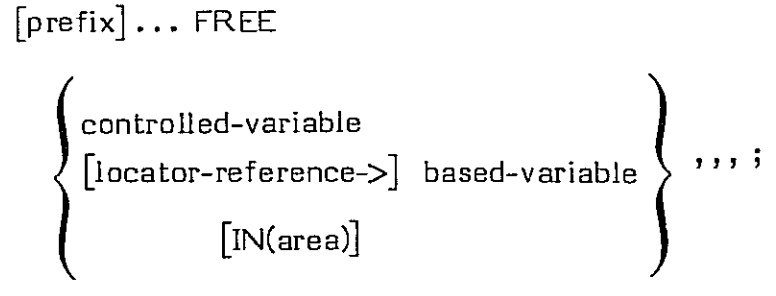
ALLOCATE

The ALLOCATE statement apportions memory for controlled and based variables.



FREE

The FREE statement releases storage that was allocated for controlled or based variables.



CONDITIONAL STATEMENTS

Conditional statements test for specified circumstances and determine the flow of control based on the results of those tests.

IF

The IF statement tests for a specified condition and directs the flow of control along one of two paths.

[prefix]... IF expression

THEN executable-unit-1

[ELSE executable-unit-2]

where each executable-unit is

{
begin-block
do-group
ALLOCATE-statement
assignment-statement
CALL-statement
CLOSE-statement
DELETE-statement
FREE-statement
GET-statement
GOTO-statement
IF-statement
LOCATE-statement
null-statement
ON-statement
OPEN-statement
PUT-statement
READ-statement
RETURN-statement
REVERT-statement
REWRITE-statement
SIGNAL-statement
STOP-statement
WRITE-statement
}

The ON statement establishes an on-unit for the specified condition. The on-unit specifies the action to be taken when the condition is raised.

[prefix]... ON condition [SNAP] { on-unit / SYSTEM ; }

where on-unit is

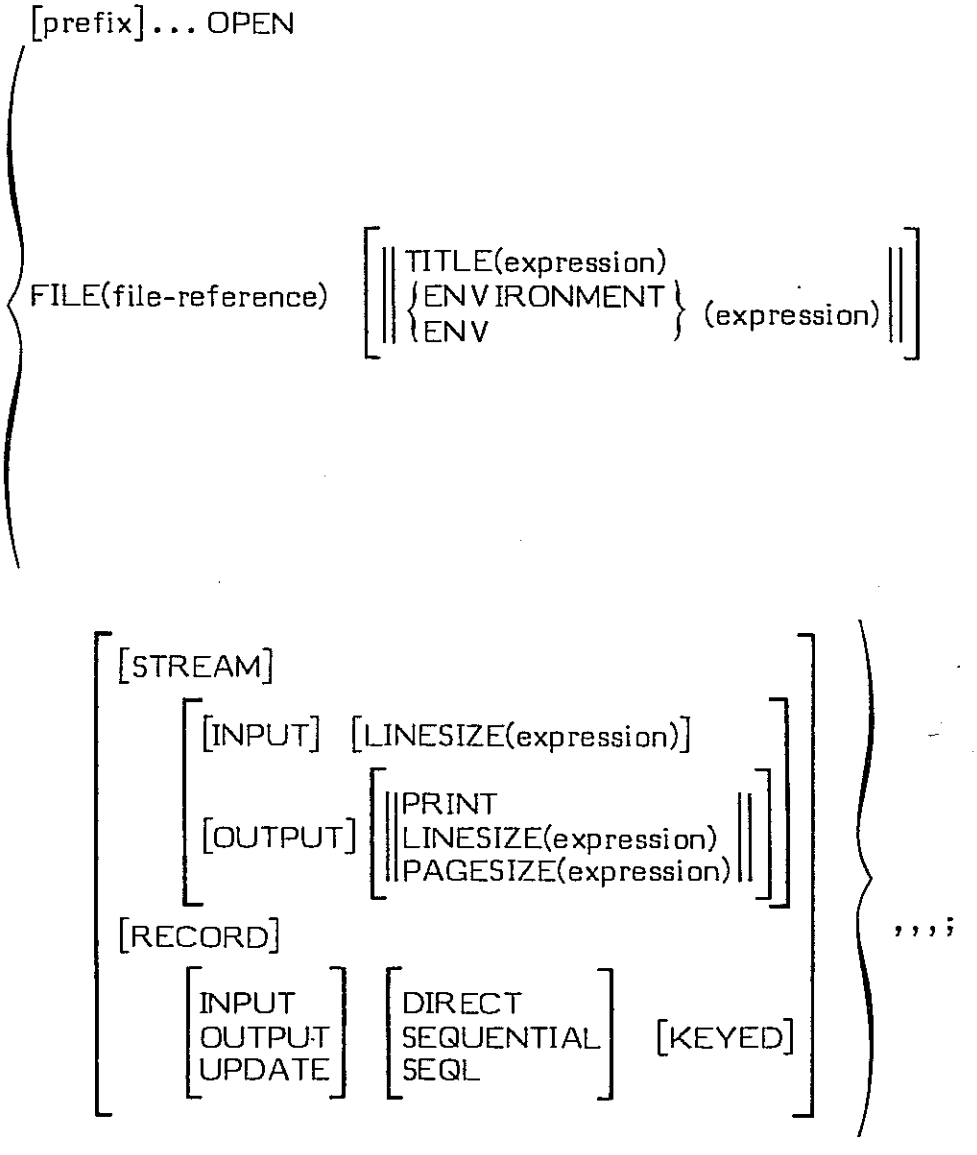
{ begin-block
ALLOCATE-statement
assignment-statement
CALL-statement
CLOSE-statement
DELETE-statement
FREE-statement
GET-statement
GOTO-statement
LOCATE-statement
null-statement
OPEN-statement
PUT-statement
READ-statement
RETURN-statement
REVERT-statement
REWRITE-statement
SIGNAL-statement
STOP-statement
WRITE-statement }

I/O STATEMENTS

The input/output statements cause data to be transferred between central memory and peripheral devices. Record I/O statements transfer complete records to and from CYBER Record Manager (CRM) files. A CRM file is a local file known to the record manager and the operating system. Stream I/O statements transfer strings of data. Edit-directed stream I/O formats data; list-directed stream I/O does not format data.

OPEN

The OPEN statement opens a file for I/O operations.



CLOSE

The CLOSE statement closes a file.

```
[prefix]... CLOSE { FILE(file-reference)
                  [ {ENVIRONMENT} (expression) ] } ,,, ;
                  [ {ENV        }              ]
```

READ

The READ statement is a record I/O statement that transfers a record from a CRM file to central memory.

```
[prefix]... READ FILE(file-reference)
    {INTO(read-target)   } [KEY(expression)   ]
    {SET(pointer)        } [KEYTO(key-target) ] ;
    {IGNORE(expression)  }
```

REWRITE

The REWRITE statement is a record I/O statement that replaces an existing record in a CRM file.

```
[prefix]... REWRITE FILE(file-reference)
    [FROM(source) [KEY(expression)]] ;
```

WRITE

The WRITE statement is a record I/O statement that transfers a record from central memory to a CRM file.

```
[prefix]... WRITE FILE(file-reference)
    FROM(source) [KEYFROM(expression)] ;
```

DELETE

The DELETE statement deletes a record from a CRM file.

```
[prefix]... DELETE FILE(file-reference)
    KEY(expression) ;
```

LOCATE

The LOCATE statement is a record I/O statement that allocates a based variable for use as an output buffer, and can transmit a record to a CYBER Record Manager file.

```
[prefix]... LOCATE based-variable

    FILE(file-reference) [ | SET(pointer)      | ] ;
                         [ | KEYFROM(expression)| ]
```

FORMAT

The FORMAT statement is used during edit-directed stream I/O to control the format of the transmitted data.

```
[condition-prefix]...{format-name:}...

    FORMAT(format-specification,,,) ;
```

where format-specification is

```
{integer     }
{(expression)}(format-specification,,,)

                                  A [(width)]
                                  B [(width)]
                                  {COLUMN} [(new-column)]
                                  {COL   }
                                  E (width,fd [,sd])
                                  F (width [,fd [,scale]])
[integer     ]                    LINE [(new-line)]
[(expression)]                    P 'picture-specification'
                                  PAGE
                                  R(remote-format)
                                  SKIP [(line-count)]
                                  X [(space-count)]
```

The GET statement is a stream I/O statement which transfers data from a peripheral device to central memory.

File Positioning

```
[prefix]... GET [COPY [(file-reference)]]
    [FILE(file-reference)] SKIP [(line-count)] ;
```

Stream Input

```
[prefix]... GET [COPY [(file-reference)]]
   [ [FILE(file-reference)] [SKIP [(line-count)]]
     STRING(input-source)                          ]
   { LIST(input-target,,,)                         }
   { EDIT{(input-target,,,)                        } ;
   {          (format-specification,,,)}...        }
```

where input-target is

```
{ target-variable                  }
{ pseudovariable                   }
{ (input-target,,, embedded-do)    }
```

where embedded-do is

```
DO index = { start-expression
           [ [ TO expression ] ] [WHILE(expression)] } ,,,
           [ [ BY expression ] ]
```

where format-specification is

```
{ { integer      } (format-specification,,,) }
{ { (expression) }                           }
{                                            }
{ [ integer      ]  { A (width)                                    } }
{ [ (expression) ]  { B (width)                                    } }
{                   { { COLUMN } [(new-column)]                    } }
{                   { { COL    }                                   } }
{                   { E (width,fd[,sd])                            } }
{                   { F (width [,fd [,scale]] )                    } }
{                   { P 'picture-specification'                    } }
{                   { R (remote-format)                            } }
{                   { SKIP [(line-count)]                          } }
{                   { X [(space-count)]                            } }
```

The PUT statement is a stream I/O statement that transfers data from central memory to a peripheral device.

File Positioning

```
[prefix]... PUT [FILE(file-reference)]

  { SKIP [(line-count)]              }
  { LINE [(new-line)]                } ;
  { PAGE [LINE [(new-line)]]         }
```

Stream Output

```
                  [ [FILE(file-reference)]                        ]
                  [                                               ]
[prefix]... PUT   [      [ SKIP (line-count)                  ]   ]
                  [      [ [PAGE] [LINE [(new-line)]]         ]   ]
                  [                                               ]
                  [ STRING(output-target)                         ]

      { LIST(output-source,,,)                        }
      { EDIT {(output-source,,,)                      }   ;
      {       (format-specification,,,)} ...          }
```

where output-source is

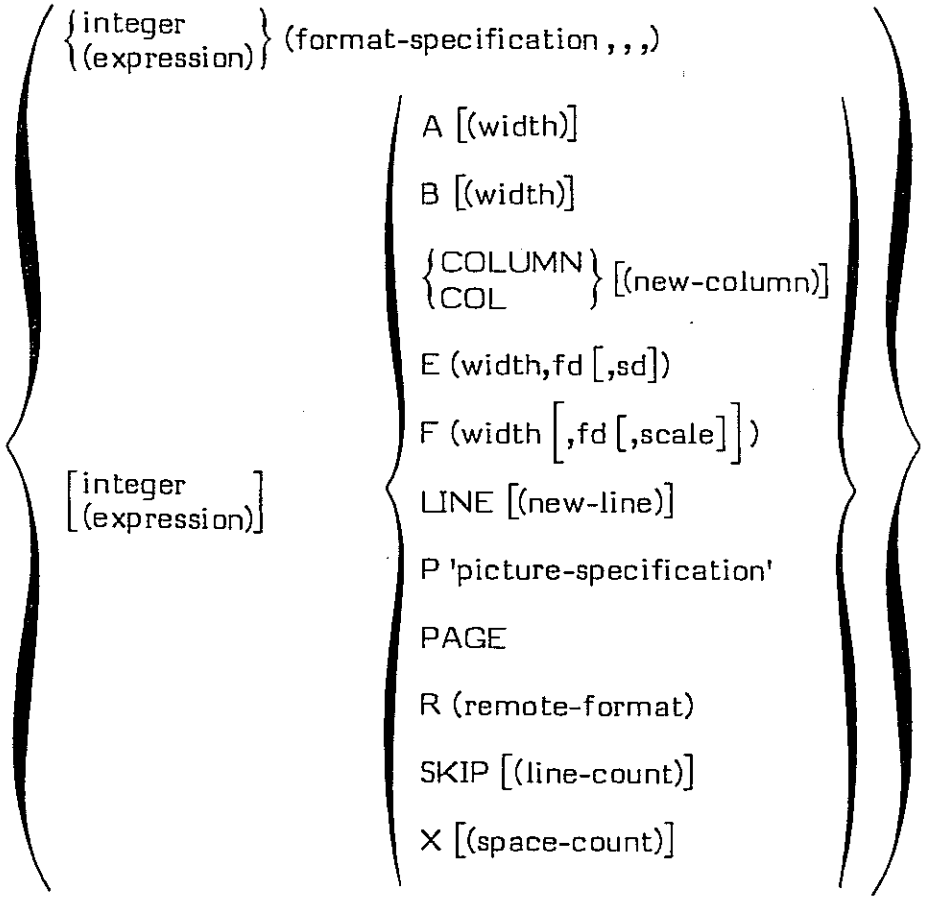
```
  { expression                              }
  { (output-source,,, embedded-do)          }
```

where embedded-do is

```
  DO index = { start-expression

    [ | TO expression | ]  [WHILE(expression)] } ,,,
    [ | BY expression | ]
```

where format-specification is



PICTURE SPECIFICATION CODES

Picture specifications are required for variables declared with the PICTURE attribute, and for input/output statements that use the P format item. A picture specification consists of a group of picture specification codes that describe the type and form of a pictured data item. A picture can be specified for a pictured character, pictured numeric fixed point, or pictured numeric floating point item.

PICTURED CHARACTER

A pictured character item has a character value that can contain characters, digits, or blanks.

Pictured character must include at least one A or X code and is

$$\left\{\begin{matrix}9\\A\\X\end{matrix}\right\} \ldots$$

The picture codes that can be used for pictured character items are shown in table 3.

TABLE 3. CODES FOR PICTURED CHARACTER

Category	Code	Use
Character codes	9	Digit or blank
	A	Letter A-Z or blank
	X	Any character

PICTURED NUMERIC FIXED POINT

A pictured numeric fixed point item has a character value that represents an edited fixed point decimal value. The character value contains the decimal digits and edited characters.

Pictured numeric fixed must include at least one of the codes 9 Z * Y T I or R, must include no more than one sign code S + - T I R CR or DB, can include insertion codes / , . or B at any position, and is

$$\left\{\begin{matrix}\text{nondrifting-field}\\ \text{drifting-field}\end{matrix}\right\} \quad \left[\text{F(} \left[\begin{matrix}+\\ -\end{matrix}\right] \text{ integer)}\right]$$

where nondrifting-field is

$$\left\{\begin{matrix}\left[\$\right] \bullet \left[\begin{matrix}\text{S}\\ +\\ -\end{matrix}\right] \bullet \text{digits}\\ \left[\$\right] \bullet \text{digits} \left\{\begin{matrix}\text{CR}\\ \text{DB}\end{matrix}\right\}\end{matrix}\right\}$$

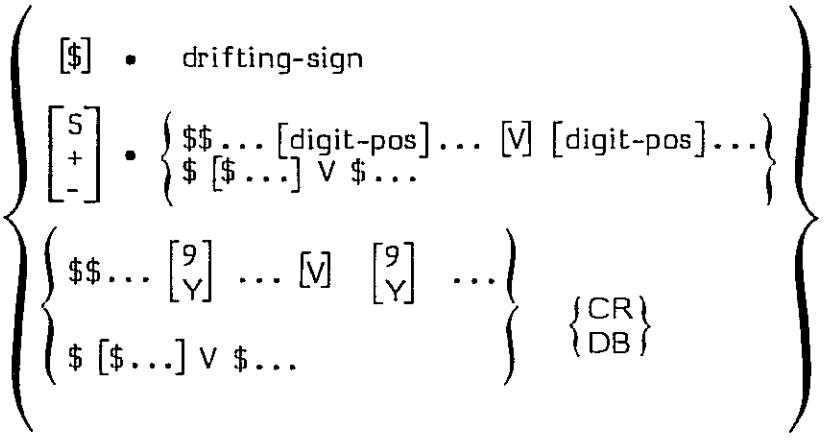
where digits is

$$\left\{\begin{matrix}\left[\begin{matrix}\text{Z}\ldots\\ *\ldots\end{matrix}\right] \text{[digit-pos]}\ldots \text{[V]} \text{[digit-pos]}\ldots\\ \left[\begin{matrix}\text{Z}\ldots\\ *\ldots\end{matrix}\right] \begin{matrix}\text{V Z}\ldots\\ \text{V }*\ldots\end{matrix}\end{matrix}\right\}$$

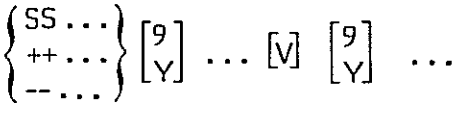
where digit-pos is

$$\left\{\begin{matrix}9\\ \text{Y}\\ \text{T}\\ \text{I}\\ \text{R}\end{matrix}\right\}$$

where drifting-field is



where drifting-sign is



The picture codes that can be used with pictured numeric fixed point items are shown in table 4.

PICTURED NUMERIC FLOATING POINT

A pictured numeric floating point item has a character value that represents an edited floating point decimal value. The character value contains the decimal digits and the edited characters.

Pictured numeric floating point is

where mantissa must include at least one of the codes 9 Z * Y T I or R, must include no more than one sign code S + - T I or R, can include insertion codes / , . or B at any position, and is

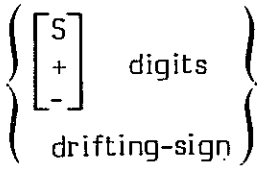


TABLE 4. CODES FOR PICTURED NUMERIC FIXED POINT

Category	Code	Use
Digit code	9	Digit
	Z	Leading zero suppression digit
	*	Leading zero replacement digit
	Y	Zero suppression digit
Decimal point code	V	Assumed decimal point
Sign position codes	S	Sign + or - (can drift)
	+	Sign if + (can drift)
	-	Sign if - (can drift)
Signed digit codes	T	Overpunch digit with + or -
	I	Overpunch digit if +
	R	Overpunch digit if -
Sign suffix codes	CR	Credit indicator
	DB	Debit indicator
Currency code	$	Dollar sign (can drift)
Insertion codes	/	Inserted slash
	,	Inserted comma
	.	Inserted period
	B	Inserted blank
Scaling factor code	F	Scaling factor for arithmetic value

and where exponent must include at least one of the codes 9 Z * Y T I or R, must include no more than one sign code S + - T I or R, can include insertion codes / , . or B at any position, and is

$$\left[\begin{matrix}S\\+\\-\end{matrix}\right]\left\{\begin{matrix}\text{digit-pos}\ldots\\\left\{\begin{matrix}Z\\*\end{matrix}\right\}\ldots\ [\text{digit-pos}]\ldots\end{matrix}\right\}$$

The picture codes that can be used for the mantissa and exponent fields of pictured numeric floating point items are shown in table 5.

TABLE 5. CODES FOR PICTURED NUMERIC FLOATING POINT

Category	Code	Use
Mantissa		
Digit codes	9	Digit
	Z	Leading zero suppression digit
	*	Leading zero replacement digit
	Y	Zero suppression digit
Decimal point code	V	Assumed decimal point
Sign position codes	S	Sign + or - (can drift)
	+	Sign if + (can drift)
	-	Sign if - (can drift)
Signed digit codes	T	Overpunch digit with + or -
	I	Overpunch digit if +
	R	Overpunch digit if -
Insertion codes	/	Inserted slash
	,	Inserted comma
	.	Inserted period
	B	Inserted blank

TABLE 5. CODES FOR PICTURED NUMERIC FLOATING POINT (Contd)

Category	Code	Use
Exponent		
Exponent codes	E	Start exponent field (supply E)
	K	Start exponent field (suppress E)
Digit codes	9	Digit
	Z	Leading zero suppression digit
	*	Leading zero replacement digit
	Y	Zero suppression digit
Sign position codes	S	Sign + or -
	+	Sign if +
	-	Sign if -
Signed digit codes	T	Overpunch digit with + or -
	I	Overpunch digit if +
	R	Overpunch digit if -
Insertion codes	/	Inserted slash
	,	Inserted comma
	.	Inserted period
	B	Inserted blank

BUILTIN FUNCTIONS

PL/I provides utility routines to perform commonly occurring operations. Builtin function names require explicit declaration only if the same name is used in another context in a containing program block. The builtin functions are summarized below. Builtin function names can be abbreviated by concatenating the underlined characters.

Syntax	Value Returned
ABS(x)	Absolute value of x
ACOS(x)	Arccosine of x in radians
ADD(x,y,p[,q])	Sum of x and y to precision (p,q)
ADDR(x)	Address of x
AFTER(strg1,strg2)	Portion of strg2 that occurs after strg1
ALLOCATION(x)	Number of generations allocated for x
ASIN(x)	Arcsine of x in radians
ATAN(x)	Arctangent of x in radians
ATAND(x)	Arctangent of x in degrees
ATANH(x)	Inverse hyperbolic tangent of x in radians
BEFORE(strg1,strg2)	Portion of strg1 that occurs before strg2
BINARY(x[,p[,q]])	Binary representation of x to precision (p,q)
BIT(x[,length])	Bit string representation of x of specified length

Syntax	Value Returned
BOOL(x,y,op)	Result of operation op performed on x and y
CEIL(x)	Smallest integer not less than x
CHARACTER(x[,length])	Character string representation of x of specified length
COLLATE[()]	All characters in collating sequence in order
COPY(strg,count)	strg copied count times
COS(x)	Cosine of x, where x is in radians
COSD(x)	Cosine of x, where x is in degrees
COSH(x)	Hyperbolic cosine of x, where x is in radians
DATE[()]	6-character string representing current date
DECAT(strg1,strg2,op)	Result of operation op performed on strg1 and strg2
DECIMAL(x[,p[,q]])	Decimal representation of x to precision (p,q)
DIMENSION(x,n)	Span of dimension n of array x
DIVIDE(x,y,p[,q])	Result of x divided by y to precision (p,q)
EMPTY[()]	An empty area variable
ERF(x)	Error function of x
ERFC(x)	Complementary error function of x
EXP(x)	Exponential of x

Syntax	Value Returned
FIXED(x,p[,q])	Fixed point representation of x to precision (p,q)
FLOAT(x,p)	Floating point representation of x to precision p
FLOOR(x)	Largest integer less than x
HBOUND(x,n)	Upper bound of dimension n of array x
HIGH(length)	String of the specified length of the highest character in the collating sequence
INDEX(strg1,strg2)	Portion of strg2 in strg1
LBOUND(x,n)	Lower bound of dimension n of array x
LENGTH(strg)	Length of strg
LINENO(file-ref)	Current line number of file file-ref
LOG(x)	Natural logarithm of x
LOG10(x)	Common logarithm of x
LOG2(x)	Logarithm base 2 of x
LOW(length)	String of the specified length of the lowest character in the collating sequence
MAX(x1,x2,,,)	Highest value of x1, x2, ...
MIN(x1,x2,,,)	Lowest value of x1, x2, ...
MOD(x,y)	x(modulo y)

Syntax	Value Returned
MULTIPLY(x,y,p[,q])	Product of x and y to precision (p,q)
NULL[()]	A null pointer
OFFSET(pointer,area)	Offset value corresponding to the given pointer value
ONCHAR[()]	Character that caused a condition to be raised
ONCODE[()]	ONCODE value for the raised condition
ONFILE[()]	Name of the file constant for which the CONVERSION condition was raised
ONKEY[()]	Key for the record for which the I/O condition was raised
ONLOC[()]	Name of the entry point of the procedure in which a condition was raised
ONSOURCE[()]	Character string which caused the CONVERSION condition to be raised
PAGENO(file-ref)	Current page number of the file file-ref
POINTER(offset,area)	Pointer value corresponding to the given offset value
PRECISION(x,p[,q])	x to precision (p,q)
REVERSE(strg)	Character or bit string strg in reverse order
ROUND(x,n)	x rounded at digit n
SIGN(x)	-1, 0, or 1 indicating if x is negative, zero, or positive.

Syntax	Value Returned
SIN(x)	Sine of x, where x is in radians
SIND(x)	Sine of x, where x is in degrees
SINH(x)	Hyperbolic sine of x, where x is in radians
SQRT(x)	Square root of x
SUBSTR(strg,pos[,length])	Substring of strg of length specified beginning at pos
SUBTRACT(x,y,p[,q])	x minus y to precision (p,q)
TAN(x)	Tangent of x, where x is in radians
TAND(x)	Tangent of x, where x is in degrees
TANH(x)	Hyperbolic tangent of x, where x is in radians
TIME[()]	6-character string representing the current time
TRANSLATE(strg,rep[,pat])	strg with occurrences of rep replaced with pat
TRUNC(x)	x with fractional part truncated
UNSPEC(x)	Internal bit representation of x
VALID(pic-var)	Indicates if pic-var is valid according to its picture
VERIFY(strg1,strg2)	Position of character in strg1 which differs from the corresponding position in strg2

COMPILATION AND EXECUTION

The PL/I compiler is called using the PLI control statement. The control statement has three possible formats:

```
PLI(p1,p2,...,pn) comments
PLI,p1,p2,...,pn. comments
PLI. comments
```

Parameters and comments are optional. Parameters can be written in any order. The possible parameters are:

B (Binary Output)

omitted	Write binary output to file LGO.
B	Write binary output to file BIN.
B=lfn	Write binary output to local file lfn.
B=0	Suppress binary output.

BL (Burstable Listing)

omitted	Error and attribute lists begin anywhere on page.
BL	Error and attribute lists begin on new page.

COL (Source Columns)

omitted	Source program appears in columns 1 through 72.
COL	Source program appears in columns 2 through 72.
COL=m/n[/p]	Source program appears in columns m through n. Carriage control is in column p.

Carriage control characters are:

blank	single space
0	double space
-	triple space
1	new page

DB (Debug Options)

omitted	Same as DB=0.
DB	Same as DB=B.
DB=B	Loadable binary output produced regardless of errors.
DB=0	Loadable binary output produced for external procedures without errors of severity C or F.

E (Error File)

omitted	Error information written to file OUTPUT.
E	Error information written to file ERRS.
E=lfn	Error information written to local file lfn.
E=0	Error information suppressed.

EL (Error Level)

omitted	Same as EL=W.
EL	Same as EL=F.
EL=I	All diagnostics, including informational, are listed.
EL=T	Trivial errors and those of severity W, F, and C are listed.
EL=W	Warning error diagnostics and those of severity F and C are listed.
EL=F	Fatal errors and those of severity C are listed.
EL=C	Compiler errors only are listed.

ET (Error Termination)

omitted	Same as ET=0.
ET	Same as ET=F.
ET=I	Abort job step if errors of severity I or greater occur.
ET=T	Abort job step if errors of severity T or greater occur.
ET=W	Abort job step if errors of severity W or greater occur.
ET=F	Abort job step if errors of severity F or greater occur.
ET=C	Abort job step if errors of severity C occur.
ET=0	Do not abort job step.

GO (Load and Execute)

omitted	Same as GO=0.
GO	Binary output is loaded and executed after compilation.
GO=0	Binary output is not loaded.

I (Input File)

omitted	Source code is read from file INPUT.
I=lfn	Source code is read from local file lfn.

INRULE

omitted	Same as INRULE=0.
INRULE	Nonstandard default attributes are used.
INRULE=0	Standard default attributes are used.

L (Listing File)

omitted	Listing written to file OUTPUT.
L	Listing written to file LIST.
L=lfn	Listing written to local file lfn.
L=0	Listing suppressed.

LO (Listing Options)

omitted	Same as LO=S/A.
LO	Same as LO=S/A/R.
LO=A	Attributes for each identifier listed.
LO=O	Object code listed.
LO=R	Reference list generated.
LO=S	Source program listed.
LO=0	Only error messages are listed.

Multiple options can be selected by separating options with slashes.

PD (Print Density)

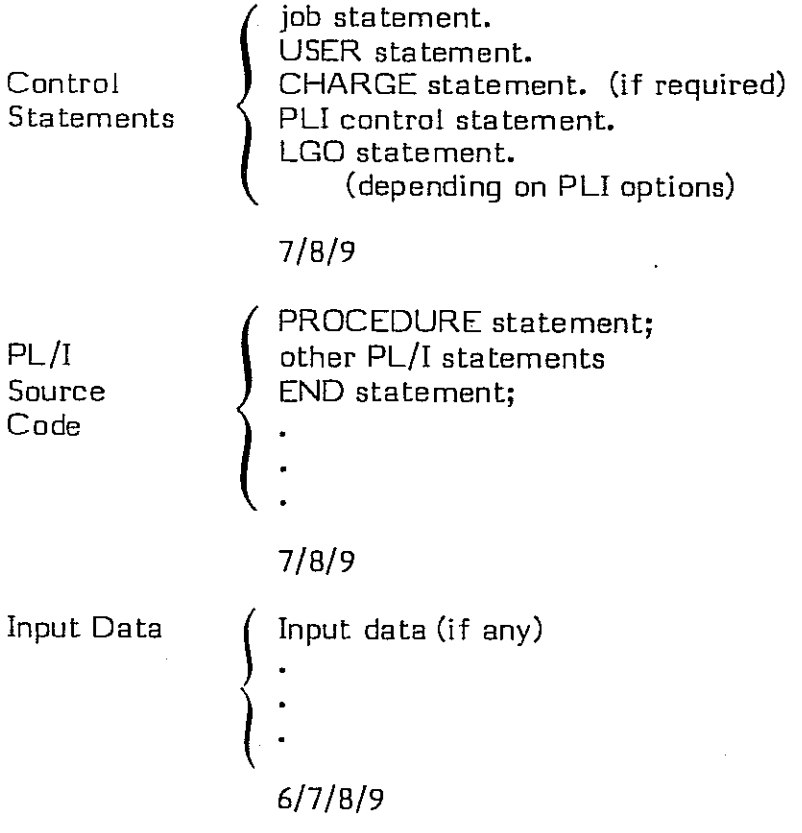
omitted	Same as PD=6.
PD	Same as PD=8.
PD=6	Listing printed 6 lines per inch.
PD=8	Listing printed 8 lines per inch.

PS (Page Size)

omitted	Page size is 60 lines if print density is 6, or 80 lines if print density is 8.
PS=n	Page size is n lines.

JOB STRUCTURE

NOS and NOS/BE PL/I jobs consist of three components: control statements, source code, and input data. The components are separated by end-of-record marks (multipunched 7/8/9 cards). The job is terminated with an end-of-file mark (a multipunched 6/7/8/9 card). The following is an example of a NOS job:



The NOS/BE job structure is the same, except the USER and CHARGE statements are omitted. An ACCOUNT statement might be required between the job statement and PLI control statement, depending on the installation.

LIST OF KEYWORDS

Keywords are identifiers that have special meaning to the compiler when used in the appropriate context. PL/I keywords are not reserved words. The following is a list of PL/I keywords and their functions. Keywords can be abbreviated by concatenating the underlined characters.

A	Format item
ABS	Builtin function
ACOS	Builtin function
ADD	Builtin function
ADDR	Builtin function
AFTER	Builtin function
ALIGNED	Attribute
ALLOCATE	Statement
ALLOCATION	Builtin function
AREA	Attribute
AREA	Condition
ASIN	Builtin function
ATAN	Builtin function
ATAND	Builtin function
ATANH	Builtin function
AUTOMATIC	Attribute
B	Format item
B1,B2,B3,B4	Format items
BASED	Attribute
BEFORE	Builtin function
BEGIN	Statement
BINARY	Attribute
BINARY	Builtin function
BIT	Attribute
BIT	Builtin function
BOOL	Builtin function
BUILTIN	Attribute
BY	Option
BY NAME	Option
C	Format item
CALL	Statement
CEIL	Builtin function
CHARACTER	Attribute
CHARACTER	Builtin function
CLOSE	Statement
COLLATE	Builtin function
COLUMN	Format item
COMPLEX	Attribute
COMPLEX	Builtin function
CONDITION	Attribute
CONDITION	Condition
CONJG	Builtin function
CONSTANT	Attribute
CONTROLLED	Attribute

CONVERSION	Condition
CONVERSION	Enabled-condition name
COPY	Builtin function
COPY	Option
COS	Builtin function
COSD	Builtin function
COSH	Builtin function
DATA	Option
DATE	Builtin function
DECAT	Builtin function
DECIMAL	Attribute
DECIMAL	Builtin function
DECLARE	Statement
DEFAULT	Statement
DEFINED	Attribute
DELETE	Statement
DIMENSION	Attribute
DIMENSION	Builtin function
DIRECT	Attribute
DIRECT	Option
DIVIDE	Builtin function
DO	Option
DO	Statement
DOT	Builtin function
E	Format item
EDIT	Option
ELSE	Option
EMPTY	Builtin function
END	Statement
ENFILE	Condition
ENDPAGE	Condition
ENTRY	Attribute
ENTRY	Statement
ENVIRONMENT	Attribute
ENVIRONMENT	Option
ERF	Builtin function
ERFC	Builtin function
ERROR	Condition
EVERY	Builtin function
EXP	Builtin function
EXTERNAL	Attribute
F	Format item
FILE	Attribute
FILE	Option
FINISH	Condition
FIXED	Attribute
FIXED	Builtin function
FIXEDOVERFLOW	Condition
FIXEDOVERFLOW	Enabled-condition name
FLOAT	Attribute
FLOAT	Builtin function
FLOOR	Builtin function
FORMAT	Attribute
FORMAT	Statement

FREE	Statement
FROM	Option
GENERIC	Attribute
GET	Statement
GO TO	Statement
HBOUND	Builtin function
HIGH	Builtin function
IF	Statement
IGNORE	Option
IMAG	Builtin function
IN	Option
%INCLUDE	Preprocessor statement
INDEX	Builtin function
INITIAL	Attribute
INPUT	Attribute
INPUT	Option
INTERNAL	Attribute
INTO	Option
KEY	Condition
KEY	Option
KEYED	Attribute
KEYED	Option
KEYFROM	Option
KEYTO	Option
LABEL	Attribute
LBOUND	Builtin function
LENGTH	Builtin function
LIKE	Attribute
LINE	Format item
LINE	Option
LINENO	Builtin function
LINESIZE	Option
LIST	Option
LOCAL	Attribute
LOCATE	Statement
LOG	Builtin function
LOG10	Builtin function
LOG2	Builtin function
LOW	Builtin function
MAIN	Option
MAX	Builtin function
MEMBER	Attribute
MIN	Builtin function
MOD	Builtin function
MULTIPLY	Builtin function
NAME	Condition
NOCONVERSION	Disabled-condition name
NOFIXEDOVERFLOW	Disabled-condition name
NONE	Option
NONVARYING	Attribute
NOOVERFLOW	Disabled-condition name
NOSIZE	Disabled-condition name
NOSTRINGRANGE	Disabled-condition name
NOSTRINGSIZE	Disabled-condition name

NOSUBSCRIPTRANGE	Disabled-condition name
NOUNDERFLOW	Disabled-condition name
NOZERODIVIDE	Disabled-condition name
NULL	Builtin function
OFFSET	Attribute
OFFSET	Builtin function
ON	Statement
ONCHAR	Builtin function
ONCHAR	Pseudovariable
ONCODE	Builtin function
ONFIELD	Builtin function
ONFILE	Builtin function
ONKEY	Builtin function
ONLOC	Builtin function
ONSOURCE	Builtin function
ONSOURCE	Pseudovariable
OPEN	Statement
OPTIONS	Option
OUTPUT	Attribute
OUTPUT	Option
OVERFLOW	Condition
OVERFLOW	Enabled-condition name
P	Format item
PAGE	Format item
PAGE	Option
PAGENO	Builtin function
PAGENO	Pseudovariable
PAGESIZE	Option
PARAMETER	Attribute
PICTURE	Attribute
POINTER	Attribute
POSITION	Attribute
PRECISION	Attribute
PRECISION	Builtin function
PRINT	Attribute
PRINT	Option
PROCEDURE	Statement
PROD	Builtin function
PUT	Statement
R	Format item
RANGE	Option
READ	Statement
REAL	Attribute
REAL	Builtin function
RECORD	Attribute
RECORD	Condition
RECORD	Option
RECURSIVE	Option
REFER	Option
REPEAT	Option
RETURN	Statement
RETURNS	Attribute
RETURNS	Option
REVERSE	Builtin function

REVERT	Statement
REWRITE	Statement
ROUND	Builtin function
SEQUENTIAL	Attribute
SEQUENTIAL	Option
SET	Option
SIGN	Builtin function
SIGNAL	Statement
SIN	Builtin function
SIND	Builtin function
SINH	Builtin function
SIZE	Condition
SIZE	Enabled-condition name
SKIP	Format item
SKIP	Option
SNAP	Option
SOME	Builtin function
SQRT	Builtin function
STATIC	Attribute
STOP	Statement
STORAGE	Condition
STREAM	Attribute
STREAM	Option
STRING	Builtin function
STRING	Pseudovariable
STRING	Option
STRINGRANGE	Enabled-condition name
STRINGSIZE	Condition
STRINGSIZE	Enabled-condition name
STRUCTURE	Attribute
SUBSCRIPTRANGE	Condition
SUBSCRIPTRANGE	Enabled-condition name
SUBSTR	Builtin function
SUBSTR	Pseudovariable
SUBTRACT	Builtin function
SUM	Builtin function
SYSTEM	Option
TAB	Format item
TAB	Option
TAN	Builtin function
TAND	Builtin function
TANH	Builtin function
THEN	Option
TIME	Builtin function
TITLE	Option
TO	Option
TRANSLATE	Builtin function
TRANSMIT	Condition
TRUNC	Builtin function
UNALIGNED	Attribute
UNDEFINEDFILE	Condition
UNDERFLOW	Condition
UNDERFLOW	Enabled-condition name
UNSPEC	Builtin function

UNSPEC	Pseudovariable
UPDATE	Attribute
UPDATE	Option
VALID	Builtin function
VARIABLE	Attribute
VARYING	Attribute
VERIFY	Builtin function
WHEN	Option
WHILE	Option
WRITE	Statement
X	Format item
ZERODIVIDE	Condition
ZERODIVIDE	Enabled-condition name

CORPORATE HEADQUARTERS, 8100 34th AVE. SO.
MINNEAPOLIS, MINN, 55440

SALES OFFICES AND SERVICE CENTERS
IN MAJOR CITIES THROUGHOUT THE WORLD