

**PL/I**

TECHNICAL REPORT

**ULD  
VERSION  
III**

TR 25.096  
30 June 1969

**CONCRETE SYNTAX OF PL/I**

**G. URSCHLER**

**IBM**

LABORATORY VIENNA



N O T E

This document is not an official PL/I Language Specification. For information concerning the official interpretation the reader is referred to the PL/I Language Specifications, Form No. Y33-6003-1.



IBM LABORATORY VIENNA,  
Austria

CONCRETE SYNTAX OF PL/I

by

G. URSCHLER

ABSTRACT

This report supplements the semantical definition of PL/I given in "Abstract Syntax and Interpretation of PL/I" and the specification of abstract syntax given in "Translation of PL/I into Abstract Syntax" (IBM Laboratory Vienna, TR 25.098 and TR 25.097) by a syntactical definition. The syntactical form of concrete PL/I program text is defined by means of an extended Backus notation, which is described by a meta syntax.

Locator Terms for IBM  
Subject Index

PL/I  
Backus Notation  
Formal Definition  
Syntax, concrete  
21 PROGRAMMING

TR 25.096

30 June 1969



30 June 1969

CONCRETE SYNTAX OF PL/I

PREFACE

This document is an updated version of:

- /1/ ALBER, K., OLIVA, P., URSCHLER, G.: Concrete Syntax of PL/I.  
IBM Laboratory Vienna, Techn. Report TR 25.084, 28 June 1968.

It is part of a series of documents (ULD Version III) presenting the formal definition of syntax and semantics of PL/I:

- /2/ FLECK, M.: Formal Definition of the PL/I Compile Time Facilities (ULD Version III).  
IBM Laboratory Vienna, Techn. Report TR 25.095, 30 June 1969.
- /3/ URSCHLER, G.: Concrete Syntax of PL/I (ULD Version III).  
IBM Laboratory Vienna, Techn. Report TR 25.096, 30 June 1969.
- /4/ URSCHLER, G.: Translation of PL/I into Abstract Text (ULD Version III).  
IBM Laboratory Vienna, Techn. Report TR 25.097, 30 June 1969.
- /5/ WALK, K., ALBER, K., FLECK, M., GOLDMANN, H., LAUER, P., MOSER, E., OLIVA, P., STIGLEITNER, H., ZEISEL, G.: Abstract Syntax and Interpretation of PL/I (ULD Version III).  
IBM Laboratory Vienna, Techn. Report TR 25.098, 30 April 1969
- /6/ ALBER, K., GOLDMANN, H., LAUER, P., LUCAS, P., OLIVA, P., STIGLEITNER, H., WALK, K., ZEISEL, G.: Informal Introduction to the Abstract Syntax and Interpretation of PL/I (ULD Version III).  
IBM Laboratory Vienna, Techn. Report TR 25.099, 30 June 1969.

The method and notation used in these documents are essentially taken over from the first version of a formal definition of PL/I issued by the Vienna Laboratory:

- /7/ PL/I Definition Group of the Vienna Laboratory: Formal Definition of PL/I.  
IBM Laboratory Vienna, Techn. Report TR 25.071, 30 December 1966
- /8/ ALBER, K.: Syntactical Description of PL/I Text and its Translation into Abstract Normal Form.  
IBM Laboratory Vienna, Techn. Report TR 25.074, 14 April 1967.

An outline of the method is given in:

- /9/ LUCAS, P., LAUER, P., STIGLEITNER, H.: Method and Notation for the Formal Definition of Programming Languages.  
IBM Laboratory Vienna, Techn. Report TR 25.087, 28 June 1968.

This document also contains the appropriate references to the relevant literature. The basic ideas and their application to PL/I have been made available through several workshops on the formal definition of PL/I, and presentations and publications inside and outside IBM. The method is demonstrated by application to an appropriately tailored subset of PL/I in:

- /10/ LUCAS, P., WALK, K.: On the Formal Description of PL/I.  
To be published in Annual Review in Automatic Programming - Vol.6.  
Pergamon Press, New York 1969.

The language defined in the present version is PL/I as specified in the PL/I Language Specifications, Form No. Y33-6003-1, with the addition of attention handling, input stream and string scanning, and file variables.

The present document will be made subject to validation by the PL/I Language Department, Hursley.

PRODUCTION

This document was prepared by means of automated text-processing systems. TEXT 360 was used for processing the prose parts. The formatting, indexing, cross-referencing, and updating of formula texts was handled by means of FORMULA 360.

FORMULA 360 is a syntax-controlled formula processing system which was developed in the Vienna Laboratory especially to facilitate the production and maintenance of PL/I Formal Definition documents. The achievements of K.F. KOCH in the overall design and implementation of FORMULA 360 are acknowledged in particular. Essential components of the system are due to G. URSCHLER (syntactical decomposition of formulas) and E. MOSER (formula input checker). H. Hoja and G. Zeisel contributed to the clarification and formulation of the required formatting processes.

Coordination: P. Schwarzenberger, M. Stadler

Technical control: K.F. Koch, E. Moser, M. Stadler

Data transcription: Miss W. Schatzl, Mrs. H. Deim, and sub-contractors

System support: H. Chladek, G. Lehmayr

30 June 1969

CONCRETE SYNTAX OF PL/I

CONTENTS

1. INTRODUCTION . . . . .	1
2. SYNTAX NOTATION . . . . .	1
2.1 Semantics of the Extended Backus Notation . . . . .	1
2.2 The Syntax of the Extended Backus Notation . . . . .	2
2.3 Generation of a Concrete Program Text . . . . .	3
2.3.1 The normal generation process . . . . .	3
2.3.2 Auxiliary rules for additional facilities . . . . .	5
2.3.2.1 Keyword abbreviations . . . . .	5
2.3.2.2 Multiple closure of blocks and groups . . . . .	6
2.3.3 Programs in the 48 character set . . . . .	6
3. CONCRETE SYNTAX . . . . .	1
3.1 Higher Level Productions . . . . .	1
3.1.1 Declarations . . . . .	1
3.1.1.1 Attributes . . . . .	3
3.1.1.2 Formats . . . . .	5
3.1.2 Statements . . . . .	6
3.1.2.1 Block and groups . . . . .	7
3.1.2.2 Flow of control statements . . . . .	8
3.1.2.3 Storage manipulating statements . . . . .	9
3.1.2.4 Condition and attention handling statements . . . . .	9
3.1.2.5 Input and output statements . . . . .	10
3.1.3 Expressions . . . . .	12
3.2 Lower Level Productions . . . . .	13
3.2.1 Identifiers and constants . . . . .	13
3.2.2 Pictures . . . . .	14
3.2.3 Blanks and comments . . . . .	15
3.3 List of PL/I Words: . . . . .	16

APPENDIX: CROSS-REFERENCE INDEX





30 June 1969

CONCRETE SYNTAX OF PL/I

1. INTRODUCTION

This document contains a formal description of the concrete syntax of PL/I.

The syntax is described by giving a generation process for concrete program texts (section 2.3), which is based on a context free production system. To facilitate the description of inserting spaces, the production system is divided into two parts, a higher and a lower level syntax (sections 3.1 and 3.2, respectively).

The production rules are written in extended Backus notation. The syntactical form and the meaning of this notation are explained in chapter 2.

It is the intent of this paper to present a syntax which minimizes the syntactic complexity of PL/I programs. As a consequence, the syntax is rather permissive, in the sense that it allows the production of a great number of non-sensical programs. This design aim has been adopted for the following reasons:

- (1) The syntactic description presented should be easily readable and understandable.
- (2) The syntactic description should be natural in the sense that the syntactic components are the meaningful components of a program. Adding more syntactic restrictions can easily spoil the clean structure of programs.
- (3) Additional syntactic restrictions, in particular context-dependent restrictions, can more easily be handled in the Translator (/4/) or even in the Interpreter (/5/).
- (4) The syntax in its present form is suitable for a test on non-ambiguity.



30 June 1969

CONCRETE SYNTAX OF PL/I

2. SYNTAX NOTATION2.1 SEMANTICS OF THE EXTENDED BACKUS NOTATION

The Backus notation as used in the Algol 60 Report is slightly modified in this paper. The printed brackets are replaced by spaces, so that a production rule gets the general form:

$$V ::= S_1 \mid S_2 \mid \dots \mid S_n$$

(V is to be replaced by one of the alternatives  $S_1$  or  $S_2$  or ... or  $S_n$ )

Note: In this chapter with V variables, with  $S_n$  ( $n \in \{1,2,3,\dots\}$ ) arbitrary strings and with  $T_n$  strings different from the null-string are denoted. Each of these strings may consist of a certain number of not nearer specified syntactical units, denoted by  $U_n$ .

As a further convenience we introduce the extended Backus notation, which has been developed by both the Hursley and the Vienna Laboratories.

This notation uses beyond '::<=' and '|' the metalinguistic signs '[', ']', '[', ']', '.', and '...' with the following meaning:

(1) A production like

$$\text{goto-statement} ::= \text{GOTO reference} ; \mid \text{GO TO reference} ;$$

may be shortened to

$$\text{goto-statement} ::= \{ \text{GOTO} \mid \text{GO TO} \} \text{reference} ;$$

In general

$$V ::= S_1 T_1 S_2 \mid S_1 T_2 S_2 \mid \dots \mid S_1 T_n S_2$$

may be replaced by

$$V ::= S_1 \{ T_1 \mid T_2 \mid \dots \mid T_n \} S_2$$

and vice versa. (Notice that this rule remains valid also for the case  $n=1$ )

(2) A production like

$$\text{return-statement} ::= \text{RETURN} ; \mid \text{RETURN ( expression )} ;$$

may be shortened to

$$\text{return-statement} ::= \text{RETURN} [ \text{ ( expression ) } ] ;$$

In general

$$V ::= S_1 S_2 \mid S_1 T_1 S_2 \mid \dots \mid S_1 T_n S_2$$

may be replaced by

$$V ::= S_1 [ T_1 \mid T_2 \mid \dots \mid T_n ] S_2$$

and vice versa.

- (3) A production like
- $$\text{integer} ::= \text{digit} \mid \text{integer digit}$$
- may be shortened to
- $$\text{integer} ::= \text{digit}\dots$$

In general

$$V ::= U \mid V U \text{ or } V, ::= U \mid U V$$

may be replaced by

$$V ::= U\dots \text{ and vice versa.}$$

Note: The production  $V ::= U\dots$  is often omitted and instead of  $V$  then always  $U\dots$  is written. This signifies for the inverse process of eliminating all instances of  $U\dots$ , that they must be replaced by a newly introduced variable, say  $V$ , and one of the productions  $V ::= U \mid U V$  or  $V ::= U \mid V U$  is to be added to the other production rules.

- (4) A production like
- $$\text{declarationlist} ::= \text{declaration} [ \{ , \text{declaration} \}\dots ]$$
- may be shortened to
- $$\text{declarationlist} ::= [ , \cdot \text{declaration}\dots ]$$

In general

$$V ::= S_1 T_1 [ \{ T_2 T_1 \}\dots ] S_2$$

may be replaced by

$$V ::= S_1 \{ T_2 \cdot T_1\dots \} S_2 \text{ and vice versa.}$$

Note: Instead of  $S_1 [ \{ T_2 \cdot T_1\dots \} ] S_2$  also  $S_1 [ T_2 \cdot T_1\dots ] S_2$  may be written.

## 2.2 THE SYNTAX OF THE EXTENDED BACKUS NOTATION

The expressions "syntactical unit" and "string of syntactical units" (called "unit" and "sequence" respectively in the following) have not been specified in the last section. We now define them together with the general form of the production rules of the concrete PL/I syntax recursively by means of a meta syntax. The meta syntax itself is written in Backus form.

To uphold the Backus notation also formally we use the metalinguistic signs '::<=' and '|' in the meta syntax. Therefore we are obliged to change the notation for similar syntactic or PL/I signs. We adopt the same convention as in chapter 3, that each ambiguous sign is marked on the lower syntactic level by a further underlining. This signifies, for this and only for this section, that the PL/I signs for colon, equal and or-sign get the forms ':', '=', '|' while the or-sign of the PL/I production rules is denoted by '|'.

30 June 1969

CONCRETE SYNTAX OF PL/I

Meta\_Syntax

```

prod-rule ::= not-var ::= definition
definition ::= sequence | sequence | definition
sequence ::= unit | unit sequence
unit ::= not-var | not-const | unit*** |
      [ definition ] | [ definition ] |
      [ not-const * unit*** ] | [ not-const * unit*** ]
not-var ::= sm-letter | sm-letter - not-var | sm-letter not-var
sm-letter ::= a | b | c | d | e | f | g | h | i | j | k | l | m |
            n | o | p | q | r | s | t | u | v | w | x | y | z
not-const ::= PL/I-symb | PL/I-symb not-const
PL/I-symb ::= A | B | C | D | E | F | G | H | I | J | K | L | M |
            N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
            $ | @ | # | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
            blank | _ | = | + | - | * | / | ( | ) | , | . | ; |
            : | & | | | ~ | > | < | ? | % | '

```

Auxiliary rules for the insertion of spaces:

Spaces (e.g., blanks, new lines, new pages) are optional immediately preceding or succeeding ' ::= ' or ' | ' or '{ ' or ' } ' or '[ ' or ' ] ' or ' \* ' or ' \*\*\* ' and between arbitrary adjacent units. A space is mandatory (to avoid ambiguities) between adjacent not-vars and between adjacent not-consts.

2.3 GENERATION OF A CONCRETE PROGRAM TEXT

## 2.3.1 THE NORMAL GENERATION PROCESS

First of all any implementation must provide production rules for the four implementation dependent notation variables external-option, env-option, incorporate-specification, and extralingual-character. Since PL/I has context dependent rules for the insertion of blanks and comments, which cannot be expressed by production rules of the form described in 2.1 and 2.2, the generation of a concrete PL/I program text has to be performed in four steps:

- (1) Starting with the notation variable "program", replacements are to be performed according to the higher level production rules listed in 3.1. This process is to be continued as long as any higher level production rule is applicable.

It ends up with a text consisting of "PL/I words", which are listed in 3.3. In this respect, all those sequences of PL/I symbols which in the production rules are not separated by empty space are assumed to compose words (notation constants) and not to be split up into their single symbols.



30 June 1969.

CONCRETE SYNTAX OF PL/I

### 2.3.2 AUXILIARY RULES FOR ADDITIONAL FACILITIES

PL/I contains two facilities which in the one case would lengthen unnecessarily the production rules and in the other case cannot be expressed by context independent production rules. Both facilities allow a program text to be replaced by a shorter one, without changing the semantical meaning.

#### 2.3.2.1 Keyword abbreviations

The following abbreviations may be inserted instead of the corresponding keywords. This replacement has to be performed before step 3 of the generation process described in 2.3 is performed:

#### keywords:

ATTENTION  
 AUTOMATIC  
 BCOLUMN  
 BEGINVOLUME  
 BINARY  
 BUFFERED  
 CHARACTER  
 COLUMN  
 COMPLEX  
 CONNECTED  
 CONTROLLED  
 CONVERSION  
 DECIMAL  
 DECLARE  
 DEFINED  
 ENDVOLUME  
 ENVIRONMENT  
 EXCLUSIVE  
 EXTERNAL  
 FIXEDOVERFLOW  
 INITIAL  
 INTERNAL  
 IRREDUCIBLE  
 NOCONVERSION  
 NOFIXEDOVERFLOW  
 NOOVERFLOW  
 NOSTRINGRANGE  
 NOSUBSRIPTRANGE  
 NOUNDERFLOW  
 NOZERODIVIDE  
 OVERFLOW  
 PICTURE  
 POINTER  
 POSITION  
 PROCEDURE  
 REDUCIBLE  
 SEQUENTIAL  
 STRINGRANGE  
 STRINGSIZE  
 SUBSRIPTRANGE  
 UNALIGNED  
 UNBUFFERED  
 UNDEFINEDFILE  
 UNDERFLOW  
 VARYING  
 ZERODIVIDE

#### abbreviations:

ATTN  
 AUTO  
 BCOL  
 BOV  
 BIN  
 BUF  
 CHAR  
 COL  
 CPLX  
 CONN  
 CTL  
 CONV  
 DEC  
 DCL  
 DEF  
 EOY  
 ENV  
 EXCL  
 EXT  
 FOFL  
 INIT  
 INT  
 IRRED  
 NOCONV  
 NOFOFL  
 NOOPL  
 NOSTRG  
 NOSUBRG  
 NOUFL  
 NOZDIV  
 OFL  
 PIC  
 PTR  
 POS  
 PROC  
 RED  
 SEQ  
 STRG  
 STRZ  
 SUBRG  
 UNAL  
 UNBUF  
 UNDF  
 UFL  
 VAR  
 ZDIV



2.3.2.2 Multiple closure of blocks and groups

Assume, that all four steps of the generation process described in 2.3.1 including the insertion of abbreviated keywords have been terminated.

Then a part of this program text is called a compound if it could have been generated by means of the following production rule:

```
compound ::= procedure { [ prefixlist ] [ labellist ]
                        { begin-block { group }
```

Before the rightmost semicolon of a compound, i.e., between 'END' and ';', one identifier of the labellist of the compound (i.e., the labellist before its leftmost 'PROCEDURE' or 'BEGIN' or 'DO' keyword) may be inserted.

Provided that a compound actually has such an end, e.g., 'END IDENTIFIER ;' it is allowed to omit an immediately preceding 'END ;' if the inserted identifier does not occur in the labellist of the compound which is closed by the end-clause to be omitted.

2.3.3 PROGRAMS IN THE 48 CHARACTER SET

It is possible to write PL/I programs in the following 48 character set:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
$ 0 1 2 3 4 5 6 7 8 9 blank = + - * / ( ) . , ' "
```

If the program shall be written in this character set, in addition to the processes described in 2.3.1 and 2.3.2 the following rules have to be obeyed:

- (1) From the production rules for "letter", "alphanumeric-character", "string-character" and "comment-symbol" the following 12 symbols have to be deleted:

```
@ # _ ; : & | - > < ? %
```

- (2) The following 13 PL/I words have to be handled as notation variables and to be replaced by means of the (higher level) production rules:

```
; ::= ,. >= ::= GE
~ ::= NOT <= ::= LE
& ::= AND -> ::= NG
! ::= OR ~< ::= NL
> ::= GT ~- ::= NE
< ::= LT || ::= CAT
-> ::= PT
```

For the insertion of spaces the word ',.' is handled as a delimiter and the other 12 words resulting from these replacements as non-delimiters.

- (3) The 12 sequences of letters

```
NOT, AND, OR, GT, LT, GE, LE, NG, NL, NE, CAT, PT
```

are "reserved words", i.e., no identifier must finally be replaced by any of these sequences.

- (4) In the final text, each colon ':' is to be replaced:

- (a) when immediately following a dot '.' by means of the production rule

```
: ::= space . .
```

- (b) else by means of the production rule

```
: ::= . .
```

30 June 1969

CONCRETE SYNTAX OF PL/I

3. CONCRETE SYNTAX3.1 HIGHER LEVEL PRODUCTIONS

- (1) program ::=  
    procedure\*\*\*
- (2) procedure ::=  
    [ prefixlist ] labellist PROCEDURE [ parameterlist ] [ procedure-optionslist ]  
    ; sentencelist
- (3) parameterlist ::=  
    ( [ , • identifier\*\*\* ] )
- (4) procedure-optionslist ::=  
    [ options-attribute ] returns-attribute [ ORDER | REORDER | RECURSIVE ]\*\*\*
- (5) sentencelist ::=  
    [ sentence\*\*\* ] end-clause
- (6) end-clause ::=  
    [ prefixlist ] [ labellist ] END ;
- (7) sentence ::=  
    procedure | entry | declaration-sentence | format-sentence | statement
- (8) entry ::=  
    labellist ENTRY [ parameterlist ] [ returns-attribute ] ;

3.1.1 DECLARATIONS

- (9) declaration-sentence ::=  
    [ labellist ] { declare-sentence | default-sentence }
- (10) declare-sentence ::=  
    DECLARE declarationlist ;

- (11) declarationlist ::=  
       { , • declaration\*\*\* }
- (12) declaration ::=  
       [ integer ] { identifier |  
       ( declarationlist ) } [ dimension-attribute ] [ attribute\*\*\* ]
- (13) default-sentence ::=  
       default-option-1 | default-option-2
- (14) default-option-1 ::=  
       DEFAULT ALL [ attribute-spec ] ;
- (15) default-option-2 ::=  
       DEFAULT { , • default-spec\*\*\* } ;
- (16) default-spec ::=  
       simple-default-spec | factored-default-spec
- (17) simple-default-spec ::=  
       range-spec [ attribute-spec ]
- (18) range-spec ::=  
       identifier-range-spec | DESCRIPTORS
- (19) identifier-range-spec ::=  
       RANGE ( [ [ , • { identifier | letter : letter }\*\*\* ] | \* ] )
- (20) attribute-spec ::=  
       [ dimension-attribute ] [ attribute | value-clause ]\*\*\* | SYSTEM
- (21) value-clause ::=  
       VALUE ( { , • value-spec\*\*\* } )
- (22) factored-default-spec ::=  
       ( { , • default-spec\*\*\* } ) [ attribute-spec ]
- (23) value-spec ::=  
       precision-spec | string-attribute | area-attribute

30 June 1969

CONCRETE SYNTAX OF PL/I

(24) precision-spec ::=  
     arithmetic-attribute\*\*\* |  
     ( [ , \* arithmetic-attribute\*\*\* ] ) arithmetic-attribute

### 3.1.1.1 Attributes

(25) options-attribute ::=  
     OPTIONS ( [ , \* external-option\*\*\* ] )

(26) returns-attribute ::=  
     RETURNS ( [ data-attribute | entry-name-attribute | FILE ]\*\*\* )

(27) dimension-attribute ::=  
     ( [ , \* bound-pair\*\*\* ] )

(28) bound-pair ::=  
     [ refer-expression : ] refer-expression | \*

(29) refer-expression ::=  
     expression [ REFER ( unsubscripted-reference ) ]

(30) attribute ::=  
     data-attribute | non-data-attribute | entry-name-attribute |  
     file-name-attribute | scope-attribute | like-attribute

(31) data-attribute ::=  
     arithmetic-attribute | string-attribute | VARYING | picture-attribute |  
     area-attribute | label-attribute | POINTER | offset-attribute | TASK |  
     EVENT | storage-class-attribute | defined-attribute | based-attribute |  
     UNALIGNED | ALIGNED | SECONDARY | CONNECTED | VARIABLE | initial-attribute

(32) arithmetic-attribute ::=  
     { REAL | COMPLEX | DECIMAL | BINARY | FLOAT |  
     FIXED } [ ( integer [ , signed-integer ] ) ]

(33) signed-integer ::=  
     [ + | - ] integer

(34) string-attribute ::=  
     { BIT | CHARACTER } [ ( { refer-expression | \* } ) ]

(35) picture-attribute ::=  
     PICTURE [ picture-specification ]

- (36) area-attribute ::=  
 AREA [ ( ( refer-expression | \* ) ) ]
- (37) label-attribute ::=  
 LABEL [ ( [ , • identifier\*\*\* ] ) ]
- (38) offset-attribute ::=  
 OFFSET [ ( reference ) ]
- (39) storage-class-attribute ::=  
 AUTOMATIC | STATIC | CONTROLLED
- (40) defined-attribute ::=  
 DEFINED basic-reference | POSITION ( expression )
- (41) based-attribute ::=  
 BASED [ ( reference ) ]
- (42) initial-attribute ::=  
 INITIAL { initial-call | initial-itemlist }
- (43) initial-call ::=  
 CALL reference
- (44) initial-itemlist ::=  
 ( [ , • initial-item\*\*\* ] )
- (45) initial-item ::=  
 initial-iteration | initial-constant | simple-string-constant | reference |  
 ( expression ) | \*
- (46) initial-iteration ::=  
 ( expression ) { initial-constant | initial-itemlist | reference }
- (47) initial-constant ::=  
 replicated-string-constant | arithmetic-init-constant | sterling-constant
- (48) arithmetic-init-constant ::=  
 [ + ] - ] real-constant [ [ + | - ] imaginary-constant ] |  
 [ + ] - ] imaginary-constant

30 June 1969 .

CONCRETE SYNTAX OF PL/I

- (49) non-data-attribute ::=  
       BUILTIN | generic-attribute | attention-attribute
- (50) entry-name-attribute ::=  
       ENTRY [ ( descriptorlist ) ] | returns-attribute | REDUCIBLE | IRREDUCIBLE
- (51) descriptorlist ::=  
       descriptor [ , descriptorlist ]
- (52) descriptor ::=  
       [ integer ] [ dimension-attribute ] [ attribute\*\*\* ] | \*
- (53) file-name-attribute ::=  
       FILE | file-attribute | ENVIRONMENT ( env-option )
- (54) file-attribute ::=  
       BITSTREAM | STREAM | RECORD | INPUT | OUTPUT | UPDATE | SEQUENTIAL | DIRECT |  
       BUFFERED | UNBUFFERED | KEYED | PRINT | BACKWARDS | EXCLUSIVE | TRANSIENT
- (55) generic-attribute ::=  
       GENERIC ( [ , \* generic-element\*\*\* ] )
- (56) generic-element ::=  
       reference WHEN ( descriptorlist )
- (57) scope-attribute ::=  
       INTERNAL | EXTERNAL
- (58) like-attribute ::=  
       LIKE unsubscripted-reference
- (59) attention-attribute ::=  
       ATTENTION ENVIRONMENT ( env-option )

3.1.1.2 Formats

- (60) format-sentence ::=  
       [ prefixlist ] labellist FORMAT formatlist ;

- (61) formatlist ::=  
       ( [ , • format\*\*\* ] )
- (62) format ::=  
       format-iteration | format-item
- (63) format-iteration ::=  
       [ integer | ( expression ) ] [ format-item | formatlist ]
- (64) format-item ::=  
       data-format | control-format | remote-format
- (65) data-format ::=  
       real-format | complex-format | string-format | picture-format
- (66) real-format ::=  
       { E | F } ( expression [ , expression [ , expression ] ] )
- (67) complex-format ::=  
       C ( [ real-format | picture-format ] [ , real-format | , picture-format ] )
- (68) string-format ::=  
       [ A | B | BB ] [ ( expression ) ]
- (69) picture-format ::=  
       [ BP | P ] picture-specification
- (70) control-format ::=  
       [ BCOLUMN | BX | COLUMN | LINE | PAGE | SKIP | X ] [ ( expression ) ]
- (71) remote-format ::=  
       R ( reference )

## 3.1.2 STATEMENTS

- (72) statement ::=  
       [ prefixlist ] [ labellist ] ( if-statement | unconditional-statement )
- (73) prefixlist ::=  
       { ( [ , • prefix-element\*\*\* ] ) : }\*\*\*

30 June 1969

CONCRETE SYNTAX OF PL/I

- (74) prefix-element ::=  
 prefix | no-prefix | check-condition | no-check-condition
- (75) prefix ::=  
 CONVERSION | FIXEDOVERFLOW | OVERFLOW | STRINGRANGE | STRINGSIZE |  
 SUBSCRIPTRANGE | UNDERFLOW | ZERODIVIDE
- (76) no-prefix ::=  
 NOCONVERSION | NOFIXEDOVERFLOW | NOOVERFLOW | NOSIZE | NOSTRINGSIZE |  
 NOSTRINGRANGE | NOSUBSCRIPTRANGE | NOUNDERFLOW | NOZERODIVIDE
- (77) labellist ::=  
 [ basic-reference : ]\*\*\*
- (78) unconditional-statement ::=  
 begin-block | group | goto-statement | call-statement | incorporate-statement |  
 fetch-statement | release-statement | return-statement | wait-statement |  
 delay-statement | exit-statement | stop-statement | assignment-statement |  
 allocate-statement | free-statement | on-statement | revert-statement |  
 signal-statement | enable-statement | disable-statement | access-statement |  
 open-statement | close-statement | stream-io-statement | record-io-statement |  
 display-statement | null-statement
- (79) null-statement ::=  
 ;

### 3.1.2.1 Block and groups

- (80) begin-block ::=  
 BEGIN [ [ options-attribute | ORDER | REORDER ]\*\*\* ] ; sentencelist
- (81) group ::=  
 simple-group | iterated-group
- (82) simple-group ::=  
 DO ; sentencelist
- (83) iterated-group ::=  
 DO [ do-specification | WHILE ( expression ) ] ; sentencelist
- (84) do-specification ::=  
 reference = ( , • specification\*\*\* )



(85) specification ::=  
       expression [ BY expression [ TO expression ] |  
       TO expression [ BY expression ] ] [ WHILE ( expression ) ]

### 3.1.2.2 Flow of control statements

(86) if-statement ::=  
       if-clause statement ; if-clause balanced-statement ELSE statement

(87) if-clause ::=  
       IF expression THEN

(88) balanced-statement ::=  
       [ prefixlist ] [ labellist ] { if-clause balanced-statement ELSE  
       balanced-statement | unconditional-statement }

(89) goto-statement ::=  
       { GOTO | GO TO } reference ;

(90) call-statement ::=  
       CALL reference [ call-optionslist ] ;

(91) call-optionslist ::=  
       [ TASK [ ( reference ) ] | PRIORITY ( expression ) | EVENT ( reference ) ]\*\*\*

(92) return-statement ::=  
       RETURN [ ( expression ) ] ;

(93) incorporate-statement ::=  
       INCORPORATE ( incorporate-specification )

(94) fetch-statement ::=  
       FETCH [ , \* reference\*\*\* ]

(95) release-statement ::=  
       RELEASE [ , \* reference\*\*\* ]

(96) wait-statement ::=  
       WAIT ( [ , \* reference\*\*\* ] ) [ ( expression ) ] ;

30 June 1969

CONCRETE SYNTAX OF PL/I

(97) delay-statement ::=  
       DELAY ( expression ) ;

(98) exit-statement ::=  
       EXIT ;

(99) stop-statement ::=  
       STOP ;

### 3.1.2.3 Storage manipulating statements

(100) assignment-statement ::=  
       [ , \* reference\*\*\* ] = expression [ , BY NAME ] ;

(101) allocate-statement ::=  
       ALLOCATE [ , \* { based-allocate-item | controlled-allocate-item }\*\*\* ] ;

(102) based-allocate-item ::=  
       identifier { SET ( reference ) [ IN ( reference ) ] |  
       IN ( reference ) [ SET ( reference ) ] }

(103) controlled-allocate-item ::=  
       [ integer ] identifier [ dimension-attribute ] [ { string-attribute |  
       area-attribute | initial-attribute }\*\*\* ]

(104) free-statement ::=  
       FREE [ , \* { reference [ IN ( reference ) ] }\*\*\* ] ;

### 3.1.2.4 Condition and attention handling statements

(105) on-statement ::=  
       ON condition [ SNAP ] { unconditional-statement | SYSTEM ; }

(106) revert-statement ::=  
       REVERT condition ;

(107) signal-statement ::=  
       SIGNAL condition ;

- (108) condition ::=  
 prefix | check-condition | AREA | named-io-condition | ERROR | FINISH |  
 programmer-named-condition | attention-condition
- (109) check-condition ::=  
 CHECK ( { , \* unsubscripted-reference\*\*\* } )
- (110) no-check-condition ::=  
 NOCHECK ( { , \* unsubscripted-reference\*\*\* } )
- (111) named-io-condition ::=  
 io-condition ( reference )
- (112) io-condition ::=  
 BEGINVOLUME | ENDFILE | ENDPAGE | ENDVOLUME | KEY | NAME | PENDING | RECORD |  
 TRANSMIT | UNDEFINEDFILE
- (113) programmer-named-condition ::=  
 CONDITION ( identifier )
- (114) attention-condition ::=  
 ATTENTION ( { , \* identifier\*\*\* } )
- (115) access-statement ::=  
 ACCESS ATTENTION [ ( { , \* identifier\*\*\* } ) ] [ ELSE statement | ; ]
- (116) enable-statement ::=  
 ENABLE { , \* { attention-condition [ ACCESS | ASYNC |  
 EVENT ( reference ) ]\*\*\* }\*\*\* }
- (117) disable-statement ::=  
 DISABLE attention-condition

### 3.1.2.5 Input and output statements

- (118) open-statement ::=  
 OPEN { , \* open-optionslist\*\*\* } ;
- (119) open-optionslist ::=  
 { file-attribute | FILE ( reference ) | BLINESIZE ( expression ) |  
 LINESIZE ( expression ) | PAGESIZE ( expression ) | TITLE ( expression ) |  
 ENVIRONMENT ( env-option ) | VOLUME }\*\*\*

30 June 1969

CONCRETE SYNTAX OF PL/I

- (120) close-statement ::=  
 CLOSE [ , \* close-optionslist\*\*\* ] ;
- (121) close-optionslist ::=  
 { FILE ( reference ) | ENVIRONMENT ( env-option ) | VOLUME }\*\*\*
- (122) stream-io-statement ::=  
 { GET | PUT } stream-optionslist ;
- (123) stream-optionslist ::=  
 { FILE ( reference ) | BITSTRING ( expression ) | STRING ( expression ) |  
 data-specification | COPY | SKIP [ ( expression ) ] | PAGE |  
 LINE ( expression ) }\*\*\*
- (124) data-specification ::=  
 data-directed | edit-directed | list-directed
- (125) data-directed ::=  
 DATA [ ( datalist ) ]
- (126) edit-directed ::=  
 EDIT [ ( datalist ) formatlist ]\*\*\*
- (127) list-directed ::=  
 LIST ( datalist )
- (128) datalist ::=  
 { , \* datalist-element\*\*\* }
- (129) datalist-element ::=  
 ( datalist DO do-specification ) | expression
- (130) record-io-statement ::=  
 { READ | WRITE | REWRITE | LOCATE identifier | DELETE |  
 UNLOCK } record-optionslist ;
- (131) record-optionslist ::=  
 { FILE ( reference ) | EVENT ( reference ) | FROM ( reference ) |  
 IGNORE ( expression ) | INTO ( reference ) | KEY ( expression ) |  
 KEYTO ( reference ) | KEYFROM ( expression ) | SET ( reference ) | NOLOCK }\*\*\*

(132) display-statement ::=

DISPLAY ( expression ) [ REPLY ( reference ) [ EVENT ( reference ) ] |  
EVENT ( reference ) REPLY ( reference ) ] ;

### 3.1.3 EXPRESSIONS

(133) expression ::=

expression-six | expression | expression-six

(134) expression-six ::=

expression-five | expression-six & expression-five

(135) expression-five ::=

expression-four | expression-five comparison-operator expression-four

(136) comparison-operator ::=

> | >= | = | < | <= | ~> | ~|= | ~<

(137) expression-four ::=

expression-three | expression-four || expression-three

(138) expression-three ::=

expression-two | expression-three { + | - } expression-two

(139) expression-two ::=

expression-one | expression-two { \* | / } expression-one

(140) expression-one ::=

primitive-expression | [ + | - | ~ ] expression-one |  
primitive-expression \*\* expression-one

(141) primitive-expression ::=

( expression ) | reference | constant | isub

(142) reference ::=

[ reference -> ] basic-reference

(143) basic-reference ::=

[ { identifier [ subscriptlist ] . }\*\*\* ] identifier [ subscriptlist\*\*\* ]

30 June 1969

CONCRETE SYNTAX OF PL/I

- (144) subscriptlist ::=  
 ( [ , \* [ expression | \* ]\*\*\* ] )
- (145) unsubscripted-reference ::=  
 [ . \* identifier\*\*\* ]
- (146) constant ::=  
 real-constant | imaginary-constant | sterling-constant |  
 simple-string-constant | replicated-string-constant
- (147) replicated-string-constant ::=  
 ( integer ) simple-string-constant

### 3.2 LOWER LEVEL PRODUCTIONS

#### 3.2.1 IDENTIFIERS AND CONSTANTS

- (148) identifier ::=  
 letter [ alphaameric-character\*\*\* ]
- (149) letter ::=  
 A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |  
 P | Q | R | S | T | U | V | W | X | Y | Z | \$ | @ | #
- (150) alphaameric-character ::=  
 letter | digit | \_
- (151) digit ::=  
 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- (152) isub ::=  
 integer SUB
- (153) integer ::=  
 digit\*\*\*
- (154) real-constant ::=  
 { fixed-constant | float-constant } [ B ]
- (155) fixed-constant ::=  
 integer [ . ] | [ integer ] . integer

- (156) float-constant ::=  
       fixed-constant E [ + | - ] integer
- (157) imaginary-constant ::=  
       real-constant I
- (158) simple-string-constant ::=  
       bit-string | character-string
- (159) bit-string ::=  
       ' [ bit\*\*\* ] ' B
- (160) bit ::=  
       0 | 1
- (161) character-string ::=  
       ' [ string-character\*\*\* ] '
- (162) string-character ::=  
       alphanumeric-character | BLANK | '' | = | + | - | \* | / |  
       ( | ) | , | . | ; | : | & | \_ | ~ | > | < | ? | % | extralingual-character
- (163) sterling-constant ::=  
       integer . integer . fixed-constant L

### 3.2.2 PICTURES

- (164) picture-specification ::=  
       ' picture-string [ F ( [ + | - ] integer ) ] '
- (165) picture-string ::=  
       [ ( integer ) ] picture-character [ picture-string ]
- (166) picture-character ::=  
       A | B | C | D | E | G | H | I | K | M | P | R | S | T | V |  
       X | Y | Z | \$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | + | - | \* | / | , | .

30 June 1969 .

CONCRETE SYNTAX OF PL/I

## 3.2.3 BLANKS AND COMMENTS

(167) space ::=

{ BLANK | comment }\*\*\*

(168) comment ::=

/ \* [ [ [ \*\*\* ] comment-symbol | / ]\*\*\* ] \*\*\* /

(169) comment-symbol ::=

alphanumeric-character | BLANK | ' | = | + | - | ( | ) | , |  
 . | ; | : | & | \_ | ~ | > | < | ? | % | extralingual-character



3.3 LIST OF PL/I WORDS:

.	CHECK	IGNORE	RANGE
<	CLOSE	imaginary-constant	READ
<=	COLUMN	IN	REAL
(	COMPLEX	INCORPORATE	real-constant
+	CONDITION	INITIAL	RECORD
	CONNECTED	INPUT	RECURSIVE
	CONTROLLED	integer	REDUCIBLE
&	CONVERSION	INTERNAL	REFER
*	COPY	INTO	REORDER
**	DATA	IRREDUCIBLE	RELEASE
)	DECIMAL	isub	REPLY
;	DECLARE	KEY	RETURN
-	DEFAULT	KEYED	RETURNS
-<	DEFINED	KEYFROM	REVERT
->	DELAY	KEYTO	REWRITE
-=	DELETE	LABEL	SECONDARY
-	DESCRIPTORS	LIKE	SEQUENTIAL
->	DIRECT	LINE	SET
/	DISABLE	LINESIZE	SIGNAL
,	DISPLAY	LIST	simple-string-constant
>	DO	LOCATE	SIZE
>=	E	NAME	SKIP
:	EDIT	NOCHECK	SNAP
=	ELSE	NOCONVERSION	STATIC
A	ENABLE	NOFIXEDOVERFLOW	sterling-constant
ACCESS	END	NOLOCK	STOP
ALIGNED	ENDFILE	NOOVERFLOW	STREAM
ALL	ENDPAGE	NOSIZE	STRING
ALLOCATE	ENDVOLUME	NOSTRINGRANGE	STRINGRANGE
AREA	ENTRY	NOSTRINGSIZE	STRINGSIZE
ASYN	ENVIRONMENT	NOSUBCRIPTRANGE	SUBSCRIPTRANGE
ATTENTION	ERROR	NOUNDERFLOW	SYSTEM
AUTOMATIC	EVENT	NOZERODIVIDE	TASK
B	EXCLUSIVE	OFFSET	THEN
BB	EXIT	ON	TITLE
BACKWARDS	EXTERNAL	OPEN	TO
BASED	F	OPTIONS	TRANSIENT
BCOLUMN	FETCH	ORDER	TRANSMIT
BEGIN	FILE	OUTPUT	UNALIGNED
BEGINVOLUME	FINISH	OVERFLOW	UNBUFFERED
BINARY	FIXED	P	UNDEFINEDFILE
BIT	FIXEDOVERFLOW	PAGE	UNDERFLOW
BITSTREAM	FLOAT	PAGESIZE	UNLOCK
BITSTRING	FORMAT	PENDING	UPDATE
BLINESIZE	FREE	PICTURE	VALUE
BP	FROM	picture-specification	VARIABLE
BUFFERED	GENERIC	POINTER	VARYING
BUILTIN	GET	POSITION	VOLUME
BX	GO	PRINT	WAIT
BY	GOTO	PRIORITY	WHEN
C	IDENT	PROCEDURE	WHILE
CALL	identifier	PUT	WRITE
CHARACTER	IP	R	X
			ZERODIVIDE

30 June 1969

CONCRETE SYNTAX OF PL/I

APPENDIX: CROSS-REFERENCE INDEX

This index lists all non-terminals and terminals of the productions of chapter 3.

A reference is given by the form 3-YY(ZZ), where YY is the page number within the chapter 3. For non-terminals the defining formula is indicated by an underlining.

CONCRETE SYNTAX OF PL/I

30 June 1969

- . . . . . 3-12 (143), 3-13 (145), 3-13 (155), 3-14 (162), 3-14 (163), 3-14 (166), 3-15 (169)
- < . . . . . 3-12 (136), 3-14 (162), 3-15 (169)
- <= . . . . . 3-12 (136)
- ( . . . . . 3-1 (3), 3-2 (12), 3-2 (19), 3-2 (21), 3-2 (22), 3-3 (24), 3-3 (25), 3-3 (26), 3-3 (27), 3-3 (29),  
 3-3 (32), 3-3 (34), 3-4 (36), 3-4 (37), 3-4 (38), 3-4 (40), 3-4 (41), 3-4 (44), 3-4 (45), 3-4 (46),  
 3-5 (50), 3-5 (53), 3-5 (55), 3-5 (56), 3-5 (59), 3-6 (61), 3-6 (63), 3-6 (66), 3-6 (67), 3-6 (68),  
 3-6 (70), 3-6 (71), 3-6 (73), 3-7 (83), 3-8 (85), 3-8 (91), 3-8 (92), 3-8 (93), 3-8 (96), 3-9 (97),  
 3-9 (102), 3-9 (104), 3-10 (109), 3-10 (110), 3-10 (111), 3-10 (113), 3-10 (114), 3-10 (115),  
 3-10 (116), 3-10 (119), 3-11 (121), 3-11 (123), 3-11 (125), 3-11 (126), 3-11 (127), 3-11 (129),  
 3-11 (131), 3-12 (132), 3-12 (141), 3-13 (144), 3-13 (147), 3-14 (162), 3-14 (164), 3-14 (165),  
 3-15 (169)
- + . . . . . 3-3 (33), 3-4 (48), 3-12 (138), 3-12 (140), 3-14 (156), 3-14 (162), 3-14 (164), 3-14 (166),  
 3-15 (169)
- ! . . . . . 3-12 (133), 3-14 (162), 3-15 (169)
- !! . . . . . 3-12 (137)
- & . . . . . 3-12 (134), 3-14 (162), 3-15 (169)
- \$ . . . . . 3-13 (149), 3-14 (166)
- \* . . . . . 3-2 (19), 3-3 (28), 3-3 (34), 3-4 (36), 3-4 (45), 3-5 (52), 3-12 (139), 3-13 (144), 3-14 (162),  
 3-14 (166), 3-15 (168)
- \*\* . . . . . 3-12 (140)
- ) . . . . . 3-1 (3), 3-2 (12), 3-2 (19), 3-2 (21), 3-2 (22), 3-3 (24), 3-3 (25), 3-3 (26), 3-3 (27), 3-3 (29),  
 3-3 (32), 3-3 (34), 3-4 (36), 3-4 (37), 3-4 (38), 3-4 (40), 3-4 (41), 3-4 (44), 3-4 (45), 3-4 (46),  
 3-5 (50), 3-5 (53), 3-5 (55), 3-5 (56), 3-5 (59), 3-6 (61), 3-6 (63), 3-6 (66), 3-6 (67), 3-6 (68),  
 3-6 (70), 3-6 (71), 3-6 (73), 3-7 (83), 3-8 (85), 3-8 (91), 3-8 (92), 3-8 (93), 3-8 (96), 3-9 (97),  
 3-9 (102), 3-9 (104), 3-10 (109), 3-10 (110), 3-10 (111), 3-10 (113), 3-10 (114), 3-10 (115),  
 3-10 (116), 3-10 (119), 3-11 (121), 3-11 (123), 3-11 (125), 3-11 (126), 3-11 (127), 3-11 (129),  
 3-11 (131), 3-12 (132), 3-12 (141), 3-13 (144), 3-13 (147), 3-14 (162), 3-14 (164), 3-14 (165),  
 3-15 (169)
- ; . . . . . 3-1 (2), 3-1 (6), 3-1 (8), 3-1 (10), 3-2 (14), 3-2 (15), 3-5 (60), 3-7 (79), 3-7 (80), 3-7 (82),  
 3-7 (83), 3-8 (89), 3-8 (90), 3-8 (92), 3-8 (96), 3-9 (97), 3-9 (98), 3-9 (99), 3-9 (100), 3-9 (101),  
 3-9 (104), 3-9 (105), 3-9 (106), 3-9 (107), 3-10 (115), 3-10 (118), 3-11 (120), 3-11 (122),  
 3-11 (130), 3-12 (132), 3-14 (162), 3-15 (169)
- ~ . . . . . 3-12 (140), 3-14 (162), 3-15 (169)
- ~< . . . . . 3-12 (136)
- ~> . . . . . 3-12 (136)
- ~= . . . . . 3-12 (136)
- . . . . . 3-3 (33), 3-4 (48), 3-12 (138), 3-12 (140), 3-14 (156), 3-14 (162), 3-14 (164), 3-14 (166),  
 3-15 (169)
- > . . . . . 3-12 (142)
- / . . . . . 3-12 (139), 3-14 (162), 3-14 (166), 3-15 (168)
- , . . . . . 3-1 (3), 3-2 (11), 3-2 (15), 3-2 (19), 3-2 (21), 3-2 (22), 3-3 (24), 3-3 (25), 3-3 (27), 3-3 (32),  
 3-4 (37), 3-4 (44), 3-5 (51), 3-5 (55), 3-6 (61), 3-6 (66), 3-6 (67), 3-6 (73), 3-7 (84), 3-8 (94),  
 3-8 (95), 3-8 (96), 3-9 (100), 3-9 (101), 3-9 (104), 3-10 (109), 3-10 (110), 3-10 (114), 3-10 (115),  
 3-10 (116), 3-10 (118), 3-11 (120), 3-11 (128), 3-13 (144), 3-14 (162), 3-14 (166), 3-15 (169)
- ^ . . . . . 3-14 (162), 3-15 (169)

30 June 1969

CONCRETE SYNTAX OF PL/I

\_ . . . . . 3-13 (150)  
 > . . . . . 3-12 (136), 3-14 (162), 3-15 (169)  
 >= . . . . . 3-12 (136)  
 ? . . . . . 3-14 (162), 3-15 (169)  
 : . . . . . 3-2 (19), 3-3 (28), 3-6 (73), 3-7 (77), 3-14 (162), 3-15 (169)  
 # . . . . . 3-13 (149)  
 @ . . . . . 3-13 (149)  
 ' . . . . . 3-14 (159), 3-14 (161), 3-14 (164), 3-15 (169)  
 '' . . . . . 3-14 (162)  
 = . . . . . 3-7 (84), 3-9 (100), 3-12 (136), 3-14 (162), 3-15 (169)  
 A . . . . . 3-6 (68), 3-13 (149), 3-14 (166)  
 ACCESS . . . . . 3-10 (115), 3-10 (116)  
 access-statement . . . . . 3-10 (115), 3-7 (78)  
 ALIGNED . . . . . 3-3 (31)  
 ALL . . . . . 3-2 (14)  
 ALLOCATE . . . . . 3-9 (101)  
 allocate-statement . . . . . 3-9 (101), 3-7 (78)  
 alphameric-character . . . . . 3-13 (150), 3-13 (148), 3-14 (162), 3-15 (169)  
 AREA . . . . . 3-4 (36), 3-10 (108)  
 area-attribute . . . . . 3-4 (36), 3-2 (23), 3-3 (31), 3-9 (103)  
 arithmetic-attribute . . . . . 3-3 (32), 3-3 (24), 3-3 (31)  
 arithmetic-init-constant . . . . . 3-4 (48), 3-4 (47)  
 assignment-statement . . . . . 3-9 (100), 3-7 (78)  
 ASYNC . . . . . 3-10 (116)  
 ATTENTION . . . . . 3-5 (59), 3-10 (114), 3-10 (115)  
 attention-attribute . . . . . 3-5 (59), 3-5 (49)  
 attention-condition . . . . . 3-10 (114), 3-10 (108), 3-10 (116), 3-10 (117)  
 attribute . . . . . 3-3 (30), 3-2 (12), 3-2 (20), 3-5 (52)  
 attribute-spec . . . . . 3-2 (20), 3-2 (14), 3-2 (17), 3-2 (22)  
 AUTOMATIC . . . . . 3-4 (39)  
 B . . . . . 3-6 (68), 3-13 (149), 3-13 (154), 3-14 (159), 3-14 (166)  
 BACKWARDS . . . . . 3-5 (54)  
 balanced-statement . . . . . 3-8 (88), 3-8 (86), 3-8 (88)  
 BASED . . . . . 3-4 (41)

based-allocate-item . . . . .	. <u>3-9(102)</u> , 3-9(101)
based-attribute . . . . .	. <u>3-4(41)</u> , 3-3(31)
basic-reference . . . . .	. <u>3-12(143)</u> , 3-4(40), 3-7(77), 3-12(142)
BB . . . . .	. 3-6(68)
BCOLUMN . . . . .	. 3-6(70)
BEGIN . . . . .	. 3-7(80)
begin-block . . . . .	. <u>3-7(80)</u> , 3-7(78)
BEGINVOLUME . . . . .	. 3-10(112)
BINARY . . . . .	. 3-3(32)
bit . . . . .	. <u>3-14(160)</u> , 3-14(159)
BIT . . . . .	. 3-3(34)
bit-string . . . . .	. <u>3-14(159)</u> , 3-14(158)
BITSTREAM . . . . .	. 3-5(54)
BITSTRING . . . . .	. 3-11(123)
BLANK . . . . .	. 3-14(162), 3-15(167), 3-15(169)
BLINESIZE . . . . .	. 3-10(119)
bound-pair . . . . .	. <u>3-3(28)</u> , 3-3(27)
BP . . . . .	. 3-6(69)
BUFFERED . . . . .	. 3-5(54)
BUILTIN . . . . .	. 3-5(49)
BX . . . . .	. 3-6(70)
BY . . . . .	. 3-8(85), 3-9(100)
C . . . . .	. 3-6(67), 3-13(149), 3-14(166)
CALL . . . . .	. 3-4(43), 3-8(90)
call-optionslist . . . . .	. <u>3-8(91)</u> , 3-8(90)
call-statement . . . . .	. <u>3-8(90)</u> , 3-7(78)
CHARACTER . . . . .	. 3-3(34)
character-string . . . . .	. <u>3-14(161)</u> , 3-14(158)
CHECK . . . . .	. 3-10(109)
check-condition . . . . .	. <u>3-10(109)</u> , 3-7(74), 3-10(108)
CLOSE . . . . .	. 3-11(120)
close-optionslist . . . . .	. <u>3-11(121)</u> , 3-11(120)
close-statement . . . . .	. <u>3-11(120)</u> , 3-7(78)
COLUMN . . . . .	. 3-6(70)

30 June 1969

## CONCRETE SYNTAX OF PL/I

comment . . . . .	. <u>3-15(168)</u> , 3-15(167)
comment-symbol . . . . .	. <u>3-15(169)</u> , 3-15(168)
comparison-operator . . . . .	. <u>3-12(136)</u> , 3-12(135)
COMPLEX . . . . .	3-3(32)
complex-format . . . . .	. <u>3-6(67)</u> , 3-6(65)
condition . . . . .	<u>3-10(108)</u> , 3-9(105), 3-9(106), 3-9(107)
CONDITION . . . . .	3-10(113)
CONNECTED . . . . .	3-3(31)
constant . . . . .	. <u>3-13(146)</u> , 3-12(141)
control-format . . . . .	. <u>3-6(70)</u> , 3-6(64)
CONTROLLED . . . . .	3-4(39)
controlled-allocate-item . . . . .	. <u>3-9(103)</u> , 3-9(101)
CONVERSION . . . . .	3-7(75)
COPY . . . . .	3-11(123)
D . . . . .	3-13(149), 3-14(166)
DATA . . . . .	3-11(125)
data-attribute . . . . .	. <u>3-3(31)</u> , 3-3(26), 3-3(30)
data-directed . . . . .	. <u>3-11(125)</u> , 3-11(124)
data-format . . . . .	. <u>3-6(65)</u> , 3-6(64)
data-specification . . . . .	. <u>3-11(124)</u> , 3-11(123)
datalist . . . . .	. <u>3-11(128)</u> , 3-11(125), 3-11(126), 3-11(127), 3-11(129)
datalist-element . . . . .	. <u>3-11(129)</u> , 3-11(128)
DECIMAL . . . . .	3-3(32)
declaration . . . . .	. <u>3-2(12)</u> , 3-2(11)
declaration-sentence . . . . .	. <u>3-1(9)</u> , 3-1(7)
declarationlist . . . . .	. <u>3-2(11)</u> , 3-1(10), 3-2(12)
DECLARE . . . . .	3-1(10)
declare-sentence . . . . .	<u>3-1(10)</u> , 3-1(9)
DEFAULT . . . . .	3-2(14), 3-2(15)
default-option-1 . . . . .	. <u>3-2(14)</u> , 3-2(13)
default-option-2 . . . . .	. <u>3-2(15)</u> , 3-2(13)
default-sentence . . . . .	<u>3-2(13)</u> , 3-1(9)
default-spec . . . . .	. <u>3-2(16)</u> , 3-2(15), 3-2(22)
DEFINED . . . . .	3-4(40)

## CONCRETE SYNTAX OF PL/I

30 June 1969

defined-attribute . . . . .	.3-4(40), 3-3(31)
DELAY . . . . .	.3-9(97)
delay-statement . . . . .	.3-9(97), 3-7(78)
DELETE . . . . .	.3-11(130)
descriptor . . . . .	.3-5(52), 3-5(51)
descriptorlist . . . . .	.3-5(51), 3-5(50), 3-5(51), 3-5(56)
DESCRIPTORS . . . . .	.3-2(18)
digit . . . . .	.3-13(151), 3-13(150), 3-13(153)
dimension-attribute . . . . .	.3-3(27), 3-2(12), 3-2(20), 3-5(52), 3-9(103)
DIRECT . . . . .	.3-5(54)
DISABLE . . . . .	.3-10(117)
disable-statement . . . . .	.3-10(117), 3-7(78)
DISPLAY . . . . .	.3-12(132)
display-statement . . . . .	.3-12(132), 3-7(78)
DO . . . . .	.3-7(82), 3-7(83), 3-11(129)
do-specification . . . . .	.3-7(84), 3-7(83), 3-11(129)
E . . . . .	.3-6(66), 3-13(149), 3-14(156), 3-14(166)
EDIT . . . . .	.3-11(126)
edit-directed . . . . .	.3-11(126), 3-11(124)
ELSE . . . . .	.3-8(86), 3-8(88), 3-10(115)
ENABLE . . . . .	.3-10(116)
enable-statement . . . . .	.3-10(116), 3-7(78)
END . . . . .	3-1(6)
end-clause . . . . .	.3-1(6), 3-1(5)
ENDFILE . . . . .	.3-10(112)
ENDPAGE . . . . .	.3-10(112)
ENDVOLUME . . . . .	.3-10(112)
entry . . . . .	.3-1(8), 3-1(7)
ENTRY . . . . .	3-1(8), 3-5(50)
entry-name-attribute . . . . .	.3-5(50), 3-3(26), 3-3(30)
env-option . . . . .	.3-5(53), 3-5(59), 3-10(119), 3-11(121)
ENVIRONMENT . . . . .	.3-5(53), 3-5(59), 3-10(119), 3-11(121)
ERROR . . . . .	.3-10(108)
EVENT . . . . .	.3-3(31), 3-9(91), 3-10(116), 3-11(131), 3-12(132)

30 June 1969

CONCRETE SYNTAX OF PL/I

EXCLUSIVE . . . . .	3-5 (54)
EXIT . . . . .	3-9 (98)
exit-statement . . . . .	<u>3-9 (98)</u> , 3-7 (78)
expression . . . . .	<u>3-12 (133)</u> , 3-3 (29), 3-4 (40), 3-4 (45), 3-4 (46), 3-6 (63), 3-6 (66), 3-6 (68), 3-6 (70), 3-7 (83), 3-8 (85), 3-8 (87), 3-8 (91), 3-8 (92), 3-8 (96), 3-9 (97), 3-9 (100), 3-10 (119), 3-11 (123), 3-11 (129), 3-11 (131), 3-12 (132), 3-12 (133), 3-12 (141), 3-13 (144)
expression-five . . . . .	<u>3-12 (135)</u> , 3-12 (134), 3-12 (135)
expression-four . . . . .	<u>3-12 (137)</u> , 3-12 (135), 3-12 (137)
expression-one . . . . .	<u>3-12 (140)</u> , 3-12 (139), 3-12 (140)
expression-six . . . . .	<u>3-12 (134)</u> , 3-12 (133), 3-12 (134)
expression-three . . . . .	<u>3-12 (138)</u> , 3-12 (137), 3-12 (138)
expression-two . . . . .	<u>3-12 (139)</u> , 3-12 (138), 3-12 (139)
EXTERNAL . . . . .	3-5 (57)
external-option . . . . .	3-3 (25)
extralingual-character . . . . .	3-14 (162), 3-15 (169)
F . . . . .	3-6 (66), 3-13 (149), 3-14 (164)
factored-default-spec . . . . .	<u>3-2 (22)</u> , 3-2 (16)
FETCH . . . . .	3-8 (94)
fetch-statement . . . . .	<u>3-8 (94)</u> , 3-7 (78)
FILE . . . . .	3-3 (26), 3-5 (53), 3-10 (119), 3-11 (121), 3-11 (123), 3-11 (131)
file-attribute . . . . .	<u>3-5 (54)</u> , 3-5 (53), 3-10 (119)
file-name-attribute . . . . .	<u>3-5 (53)</u> , 3-3 (30)
FINISH . . . . .	3-10 (108)
FIXED . . . . .	3-3 (32)
fixed-constant . . . . .	<u>3-13 (155)</u> , 3-13 (154), 3-14 (156), 3-14 (163)
FIXEDOVERFLOW . . . . .	3-7 (75)
FLOAT . . . . .	3-3 (32)
float-constant . . . . .	<u>3-14 (156)</u> , 3-13 (154)
format . . . . .	<u>3-6 (62)</u> , 3-6 (61)
FORMAT . . . . .	3-5 (60)
format-item . . . . .	<u>3-6 (64)</u> , 3-6 (62), 3-6 (63)
format-iteration . . . . .	<u>3-6 (63)</u> , 3-6 (62)
format-sentence . . . . .	<u>3-5 (60)</u> , 3-1 (7)
formatlist . . . . .	<u>3-6 (61)</u> , 3-5 (60), 3-6 (63), 3-11 (126)
FREE . . . . .	3-9 (104)



## CONCRETE SYNTAX OF PL/I

30 June 1969

free-statement . . . . .	<u>3-9(104)</u> , 3-7(78)
FROM . . . . .	3-11(131)
G . . . . .	3-13(149), 3-14(166)
GENERIC . . . . .	3-5(55)
generic-attribute . . . . .	<u>3-5(55)</u> , 3-5(49)
generic-element . . . . .	<u>3-5(56)</u> , 3-5(55)
GET . . . . .	3-11(122)
GO . . . . .	3-8(89)
GOTO . . . . .	3-8(89)
goto-statement . . . . .	<u>3-8(89)</u> , 3-7(78)
group . . . . .	<u>3-7(81)</u> , 3-7(78)
H . . . . .	3-13(149), 3-14(166)
I . . . . .	3-13(149), 3-14(157), 3-14(166)
identifier . . . . .	<u>3-13(148)</u> , 3-1(3), 3-2(12), 3-2(19), 3-4(37), 3-9(102), 3-9(103), 3-10(113), 3-10(114), 3-10(115), 3-11(130), 3-12(143), 3-13(145)
identifier-range-spec . . . . .	<u>3-2(19)</u> , 3-2(18)
IF . . . . .	3-8(87)
if-clause . . . . .	<u>3-8(87)</u> , 3-8(86), 3-8(88)
if-statement . . . . .	<u>3-8(86)</u> , 3-6(72)
IGNORE . . . . .	3-11(131)
imaginary-constant . . . . .	<u>3-14(157)</u> , 3-4(48), 3-13(146)
IN . . . . .	3-9(102), 3-9(104)
INCORPORATE . . . . .	3-8(93)
incorporate-specification . . . . .	3-8(93)
incorporate-statement . . . . .	<u>3-8(93)</u> , 3-7(78)
INITIAL . . . . .	3-4(42)
initial-attribute . . . . .	<u>3-4(42)</u> , 3-3(31), 3-9(103)
initial-call . . . . .	<u>3-4(43)</u> , 3-4(42)
initial-constant . . . . .	<u>3-4(47)</u> , 3-4(45), 3-4(46)
initial-item . . . . .	<u>3-4(45)</u> , 3-4(44)
initial-itemlist . . . . .	<u>3-4(44)</u> , 3-4(42), 3-4(46)
initial-iteration . . . . .	<u>3-4(46)</u> , 3-4(45)
INPUT . . . . .	3-5(54)
integer . . . . .	<u>3-13(153)</u> , 3-2(12), 3-3(32), 3-3(33), 3-5(52), 3-6(63), 3-9(103), 3-13(147), 3-13(152), 3-13(155), 3-14(156), 3-14(163), 3-14(164), 3-14(165)



non-data-attribute . . . . . 3-5(49), 3-3(30)

NOOVERFLOW . . . . . 3-7(76)

NOSIZE . . . . . 3-7(76)

NOSTRINGRANGE . . . . . 3-7(76)

NOSTRINGSIZE . . . . . 3-7(76)

NOSUBSCRIPTRANGE . . . . . 3-7(76)

NOUNDERFLOW . . . . . 3-7(76)

NOZERODIVIDE . . . . . 3-7(76)

null-statement . . . . . 3-7(79), 3-7(78)

O . . . . . 3-13(149)

OFFSET . . . . . 3-4(38)

offset-attribute . . . . . 3-4(38), 3-3(31)

ON . . . . . 3-9(105)

on-statement . . . . . 3-9(105), 3-7(78)

OPEN . . . . . 3-10(118)

open-optionslist . . . . . 3-10(119), 3-10(118)

open-statement . . . . . 3-10(118), 3-7(78)

OPTIONS . . . . . 3-3(25)

options-attribute . . . . . 3-3(25), 3-1(4), 3-7(80)

ORDER . . . . . 3-1(4), 3-7(80)

OUTPUT . . . . . 3-5(54)

OVERFLOW . . . . . 3-7(75)

P . . . . . 3-6(69), 3-13(149), 3-14(166)

PAGE . . . . . 3-6(70), 3-11(123)

PAGESIZE . . . . . 3-10(119)

parameterlist . . . . . 3-1(3), 3-1(2), 3-1(8)

PENDING . . . . . 3-10(112)

PICTURE . . . . . 3-3(35)

picture-attribute . . . . . 3-3(35), 3-3(31)

picture-character . . . . . 3-14(166), 3-14(165)

picture-format . . . . . 3-6(69), 3-6(65), 3-6(67)

picture-specification . . . . . 3-14(164), 3-3(35), 3-6(69)

picture-string . . . . . 3-14(165), 3-14(164), 3-14(165)

POINTER . . . . . 3-3(31)

30 June 1969

CONCRETE SYNTAX OF PL/I

POSITION . . . . . 3-4 (40)

precision-spec . . . . . 3-3(24), 3-2 (23)

prefix . . . . . 3-7(75), 3-7 (74), 3-10 (108)

prefix-element . . . . . 3-7(74), 3-6 (73)

prefixlist . . . . . 3-6(73), 3-1 (2), 3-1 (6), 3-5 (60), 3-6 (72), 3-8 (88)

primitive-expression . . . . . 3-12(141), 3-12 (140)

PRINT . . . . . 3-5 (54)

PRIORITY . . . . . 3-8 (91)

procedure . . . . . 3-1(2), 3-1 (1), 3-1 (7)

PROCEDURE . . . . . 3-1 (2)

procedure-optionslist . . . . . 3-1(4), 3-1 (2)

program . . . . . 3-1(1)

programmer-named-condition . . . . . 3-10(113), 3-10 (108)

PUT . . . . . 3-11 (122)

Q . . . . . 3-13 (149)

R . . . . . 3-6 (71), 3-13 (149), 3-14 (166)

RANGE . . . . . 3-2 (19)

range-spec . . . . . 3-2(18), 3-2 (17)

READ . . . . . 3-11 (130)

REAL . . . . . 3-3 (32)

real-constant . . . . . 3-13(154), 3-4 (48), 3-13 (146), 3-14 (157)

real-format . . . . . 3-6(66), 3-6 (65), 3-6 (67)

RECORD . . . . . 3-5 (54), 3-10 (112)

record-io-statement . . . . . 3-11(130), 3-7 (78)

record-optionslist . . . . . 3-11(131), 3-11 (130)

RECURSIVE . . . . . 3-1 (4)

REDUCIBLE . . . . . 3-5 (50)

REFER . . . . . 3-3 (29)

refer-expression . . . . . 3-3(29), 3-3 (28), 3-3 (34), 3-4 (36)

reference . . . . . 3-12(142), 3-4 (38), 3-4 (41), 3-4 (43), 3-4 (45), 3-4 (46), 3-5 (56), 3-6 (71), 3-7 (84),  
 3-8 (89), 3-8 (90), 3-8 (91), 3-8 (94), 3-8 (95), 3-8 (96), 3-9 (100), 3-9 (102), 3-9 (104),  
 3-10 (111), 3-10 (116), 3-10 (119), 3-11 (121), 3-11 (123), 3-11 (131), 3-12 (132), 3-12 (141),  
 3-12 (142)

RELEASE . . . . . 3-8 (95)

release-statement . . . . . 3-8(95), 3-7 (78)

remote-format . . . . .	.3-6(71), 3-6(64)
REORDER . . . . .	3-1(4), 3-7(80)
replicated-string-constant . . . . .	.3-13(147), 3-4(47), 3-13(146)
REPLY . . . . .	3-12(132)
RETURN . . . . .	3-8(92)
return-statement . . . . .	.3-8(92), 3-7(78)
RETURNS . . . . .	3-3(26)
returns-attribute . . . . .	.3-3(26), 3-1(4), 3-1(8), 3-5(50)
REVERT . . . . .	3-9(106)
revert-statement . . . . .	.3-9(106), 3-7(78)
REWRITE . . . . .	3-11(130)
S . . . . .	3-13(149), 3-14(166)
scope-attribute . . . . .	.3-5(57), 3-3(30)
SECONDARY . . . . .	3-3(31)
sentence . . . . .	.3-1(7), 3-1(5)
sentencelist . . . . .	.3-1(5), 3-1(2), 3-7(80), 3-7(82), 3-7(83)
SEQUENTIAL . . . . .	3-5(54)
SET . . . . .	3-9(102), 3-11(131)
SIGNAL . . . . .	3-9(107)
signal-statement . . . . .	.3-9(107), 3-7(78)
signed-integer . . . . .	.3-3(33), 3-3(32)
simple-default-spec . . . . .	.3-2(17), 3-2(16)
simple-group . . . . .	.3-7(82), 3-7(81)
simple-string-constant . . . . .	.3-14(158), 3-4(45), 3-13(146), 3-13(147)
SKIP . . . . .	3-6(70), 3-11(123)
SNAP . . . . .	3-9(105)
space . . . . .	.3-15(167)
specification . . . . .	.3-8(85), 3-7(84)
statement . . . . .	.3-6(72), 3-1(7), 3-8(86), 3-10(115)
STATIC . . . . .	3-4(39)
sterling-constant . . . . .	.3-14(163), 3-4(47), 3-13(146)
STOP . . . . .	3-9(99)
stop-statement . . . . .	.3-9(99), 3-7(78)
storage-class-attribute . . . . .	.3-4(39), 3-3(31)

30 June 1969

CONCRETE SYNTAX OF PL/I

STREAM . . . . .	.3-5 (54)
stream-io-statement . . . . .	. <u>3-11(122)</u> , 3-7 (78)
stream-optionslist . . . . .	. <u>3-11(123)</u> , 3-11 (122)
STRING . . . . .	.3-11 (123)
string-attribute . . . . .	. <u>3-3(34)</u> , 3-2 (23), 3-3 (31), 3-9 (103)
string-character . . . . .	. <u>3-14(162)</u> , 3-14 (161)
string-format . . . . .	. <u>3-6(68)</u> , 3-6 (65)
STRINGRANGE . . . . .	.3-7 (75)
STRINGSIZE . . . . .	.3-7 (75)
SUB . . . . .	.3-13 (152)
subscriptlist . . . . .	. <u>3-13(144)</u> , 3-12 (143)
SUBSCRIPTRANGE . . . . .	.3-7 (75)
SYSTEM . . . . .	.3-2 (20), 3-9 (105)
T . . . . .	.3-13 (149), 3-14 (166)
TASK . . . . .	.3-3 (31), 3-8 (91)
THEN . . . . .	.3-8 (87)
TITLE . . . . .	.3-10 (119)
TO . . . . .	.3-8 (85), 3-8 (89)
TRANSIENT . . . . .	.3-5 (54)
TRANSMIT . . . . .	.3-10 (112)
U . . . . .	.3-13 (149)
UNALIGNED . . . . .	.3-3 (31)
UNBUFFERED . . . . .	.3-5 (54)
unconditional-statement . . . . .	. <u>3-7(78)</u> , 3-6 (72), 3-8 (88), 3-9 (105)
UNDEFINEDFILE . . . . .	.3-10 (112)
UNDERFLOW . . . . .	.3-7 (75)
UNLOCK . . . . .	.3-11 (130)
unsubscripted-reference . . . . .	. <u>3-13(145)</u> , 3-3 (29), 3-5 (58), 3-10 (109), 3-10 (110)
UPDATE . . . . .	.3-5 (54)
V . . . . .	.3-13 (149), 3-14 (166)
VALUE . . . . .	.3-2 (21)
value-clause . . . . .	. <u>3-2(21)</u> , 3-2 (20)
value-spec . . . . .	. <u>3-2(23)</u> , 3-2 (21)
VARIABLE . . . . .	.3-3 (31)

## CONCRETE SYNTAX OF PL/I

30 June 1969

VARYING . . . . .	3-3 (31)
VOLUME . . . . .	3-10 (119), 3-11 (121)
W . . . . .	3-13 (149)
WAIT . . . . .	3-8 (96)
wait-statement . . . . .	3-8 (96), 3-7 (78)
WHEN . . . . .	3-5 (56)
WHILE . . . . .	3-7 (83), 3-8 (85)
WRITE . . . . .	3-11 (130)
X . . . . .	3-6 (70), 3-13 (149), 3-14 (166)
Y . . . . .	3-13 (149), 3-14 (166)
Z . . . . .	3-13 (149), 3-14 (166)
ZERODIVIDE . . . . .	3-7 (75)
0 . . . . .	3-13 (151), 3-14 (160)
1 . . . . .	3-13 (151), 3-14 (160), 3-14 (166)
2 . . . . .	3-13 (151), 3-14 (166)
3 . . . . .	3-13 (151), 3-14 (166)
4 . . . . .	3-13 (151), 3-14 (166)
5 . . . . .	3-13 (151), 3-14 (166)
6 . . . . .	3-13 (151), 3-14 (166)
7 . . . . .	3-13 (151), 3-14 (166)
8 . . . . .	3-13 (151), 3-14 (166)
9 . . . . .	3-13 (151), 3-14 (166)







