

# Burroughs



# PUBLICATION CHANGE NOTICE

PCN No.: 5001530-003 Date: March 1, 1982  
Publication Title: B 5000/B 6000/B 7000 Series PL/I Language Reference Manual (May 1977)

Other Affected Publications: \_\_\_\_\_

Supersedes: N/A

Description:

Incorporates the Mark 3.3 System Software release.

Replace these pages

Add pages

TOC-1  
TOC-3  
TOC-5  
TOC-7  
TOC-9  
TOC-11  
4-35  
7-13  
7-23  
7-53  
7-55  
9-11  
Index-1  
Index-3  
Index-5  
Index-7  
Index-9  
Index-11  
Index-13

4-36-A  
7-14-A  
7-24-A



## TABLE OF CONTENTS

SECTION 1.	INTRODUCTION . . . . .	1- 1
1.1	PURPOSE . . . . .	1- 1
SECTION 2.	SYNTAX NOTATION. . . . .	2- 1
2.1	NOTATION VARIABLE . . . . .	2- 1
2.2	NOTATION CONSTANT . . . . .	2- 1
2.3	SYNTACTICAL UNIT. . . . .	2- 1
2.4	SYNTAX-LANGUAGE SYMBOLS . . . . .	2- 2
2.4.1	Braces. . . . .	2- 2
2.4.2	The vertical stroke . . . . .	2- 2
2.4.3	Brackets. . . . .	2- 2
2.4.4	The ellipsis. . . . .	2- 2
2.4.5	Syntax rule . . . . .	2- 2
2.4.6	Blanks. . . . .	2- 3
SECTION 3.	PROGRAM ELEMENTS . . . . .	3- 1
3.1	60-CHARACTER SET. . . . .	3- 1
3.2	EXTRALINGUAL CHARACTER SET. . . . .	3- 2
3.3	DELIMITERS. . . . .	3- 2
3.4	OPERATORS . . . . .	3- 2
3.4.1	Arithmetic Operators. . . . .	3- 2
3.4.2	Comparison Operators. . . . .	3- 2
3.4.3	Bit String Operators. . . . .	3- 2
3.4.4	The string operator is: . . . . .	3- 3
3.5	PARENTHESES . . . . .	3- 3
3.6	SEPARATORS AND OTHER DELIMITERS . . . . .	3- 3
3.7	IDENTIFIERS . . . . .	3- 3
3.8	KEYWORDS. . . . .	3- 4
3.8.1	Statement Identifiers . . . . .	3- 4
3.8.2	Attributes. . . . .	3- 4
3.8.3	Separating Keywords . . . . .	3- 4
3.8.4	Built-in Function Names . . . . .	3- 4

3.8.5	Option Keywords . . . . .	3- 5
3.8.6	Condition . . . . .	3- 5
3.9	DATA ELEMENTS . . . . .	3- 5
3.9.1	Problem Data. . . . .	3- 5
3.9.1.1	Arithmetic Data . . . . .	3- 5
3.9.1.2	String Data . . . . .	3- 6
3.9.2	Program-Control Data. . . . .	3- 6
3.9.2.1	Label Data. . . . .	3- 6
3.9.2.2	Entry-Label Data. . . . .	3- 6
3.9.2.3	Task Data . . . . .	3- 6
3.9.2.4	Locator Data. . . . .	3- 6
3.9.2.5	Area Data . . . . .	3- 7
3.10	CONSTANTS . . . . .	3- 7
3.10.1	Real Arithmetic Constants . . . . .	3- 7
3.10.1.1	Decimal-Fixed-Point Constants . . . . .	3- 7
3.10.1.2	Binary Fixed-Point Constants. . . . .	3- 7
3.10.1.3	Decimal Floating-Point Constants. . . . .	3- 7
3.10.1.4	Binary Floating-Point Constants . . . . .	3- 8
3.10.1.5	Precision . . . . .	3- 8
3.11	STRING CONSTANTS. . . . .	3- 8
3.11.1	Character String Constants. . . . .	3- 8
3.11.2	Bit String Constants. . . . .	3- 9
3.12	NAMED CONSTANTS . . . . .	3- 9
3.12.1	Statement Label Constants . . . . .	3- 9
3.12.2	Entry Constants . . . . .	3- 10
3.12.3	File Constants. . . . .	3- 10
3.12.4	Format Label Constants. . . . .	3- 10
3.13	VARIABLES . . . . .	3- 11
3.13.1	Arithmetic Variables. . . . .	3- 11
3.13.2	Character String Variables. . . . .	3- 11
3.13.3	Bit String Variables. . . . .	3- 11
3.13.4	Statement Label Variables . . . . .	3- 11
3.13.5	Entry Variables . . . . .	3- 12

3.13.6	File Variables . . . . .	3- 12
3.13.7	Format Label Variables . . . . .	3- 13
3.14	DATA ORGANIZATION . . . . .	3- 13
3.14.1	Scalar Items . . . . .	3- 13
3.14.1.1	Constants . . . . .	3- 13
3.14.1.2	Scalar Variables . . . . .	3- 14
3.14.2	Data Aggregates . . . . .	3- 14
3.14.2.1	Arrays . . . . .	3- 14
3.14.2.2	Structures . . . . .	3- 14
3.14.2.3	Structure Arrays . . . . .	3- 15
3.14.3	Naming . . . . .	3- 17
3.14.3.1	Simple Names . . . . .	3- 17
3.14.3.2	Subscripted Names . . . . .	3- 17
3.14.3.3	Qualified Names . . . . .	3- 19
3.14.3.4	Subscripted Qualified Names . . . . .	3- 21
3.15	Comments . . . . .	3- 22
SECTION 4.	DATA DESCRIPTION . . . . .	4- 1
4.1	DECLARATIONS . . . . .	4- 1
4.1.1	Explicit Declarations . . . . .	4- 1
4.1.1.1	Label Prefixes . . . . .	4- 1
4.1.1.2	Parameters . . . . .	4- 2
4.1.2	Contextual Declarations . . . . .	4- 2
4.1.3	Implicit Declarations . . . . .	4- 3
4.1.4	Establishment of Declarations . . . . .	4- 3
4.1.5	Scope of Declarations . . . . .	4- 3
4.1.5.1	Scope of External Names . . . . .	4- 4
4.1.5.2	Basic Rule of Use of Names . . . . .	4- 6
4.2	CLASSIFICATION OF ATTRIBUTES . . . . .	4- 7
4.3	ARITHMETIC DATA DESCRIPTIONS . . . . .	4- 7
4.3.1	Mode Attribute . . . . .	4- 7
4.3.2	Base Attributes . . . . .	4- 8
4.3.3	Scale Attributes . . . . .	4- 8
4.3.4	Precision Attribute . . . . .	4- 9

4.4	STRING DATA DESCRIPTIONS . . . . .	4- 10
4.4.1	String Type Attribute . . . . .	4- 10
4.4.2	Length Attribute. . . . .	4- 10
4.4.3	Varying Attribute . . . . .	4- 11
4.5	PICTURE DATA DESCRIPTIONS . . . . .	4- 12
4.5.1	Character Picture Data Description. . . . .	4- 13
4.5.2	Numeric Picture Data Description. . . . .	4- 14
4.5.2.1	Decimal Specifiers. . . . .	4- 16
4.5.2.2	Packed Picture Classification . . . . .	4- 16
4.5.2.3	Zero Suppression Characters . . . . .	4- 17
4.5.2.4	Insertion Characters. . . . .	4- 18
4.5.2.5	Signs and Currency Characters . . . . .	4- 19
4.5.2.6	Credit, Debit and Overpunch Characters. . . . .	4- 21
4.5.2.7	Exponent Characters . . . . .	4- 22
4.5.2.8	Scale Factor. . . . .	4- 22
4.6	PROGRAM CONTROL DATA DESCRIPTION. . . . .	4- 23
4.6.1	Label Attribute . . . . .	4- 23
4.6.2	Format Attribute. . . . .	4- 25
4.6.3	Locator Attributes. . . . .	4- 26
4.6.3.1	Locator Qualification . . . . .	4- 27
4.6.4	In Attribute. . . . .	4- 28
4.6.5	Area Attribute. . . . .	4- 29
4.7	ENTRY DATA DESCRIPTION. . . . .	4- 30
4.7.1	Entry Attribute . . . . .	4- 30
4.7.2	Generic Entry Name. . . . .	4- 32
4.7.3	Builtin Attribute . . . . .	4- 34
4.7.4	Returns Attribute . . . . .	4- 35
4.7.5	Parameter Attribute . . . . .	4- 35
4.7.6	Library Attribute . . . . .	4- 36
4.7.7	Options Attribute . . . . .	4- 36
4.8	FILE DATA DESCRIPTION . . . . .	4-36- A
4.8.1	File Attribute. . . . .	4-36- B
4.8.2	Function Attribute. . . . .	4- 37

4.8.3	Transmission Attribute . . . . .	4- 37
4.8.4	Print Attribute . . . . .	4- 38
4.8.5	Keyed Attribute . . . . .	4- 39
4.8.6	Access Attribute . . . . .	4- 39
4.8.7	Environment Attribute . . . . .	4- 40
4.9	ARRAY DATA DESCRIPTION . . . . .	4- 43
4.9.1	Dimension Attribute . . . . .	4- 43
4.10	STRUCTURE DATA DESCRIPTION . . . . .	4- 44
4.10.1	Structure Attribute . . . . .	4- 44
4.10.2	Member Attribute . . . . .	4- 45
4.10.3	Like Attribute . . . . .	4- 45
4.11	STORAGE CLASS . . . . .	4- 46
4.12	SCOPE ATTRIBUTE . . . . .	4- 49
4.13	DATA ATTRIBUTES . . . . .	4- 50
4.13.1	Alignment Attribute . . . . .	4- 50
4.13.2	Initial Attribute . . . . .	4- 51
4.13.2.1	Initial List . . . . .	4- 51
4.13.2.2	Initial Call . . . . .	4- 53
4.13.3	Variable Attribute . . . . .	4- 54
4.13.4	Defined Attribute . . . . .	4- 54
4.13.4.1	Simple Defining . . . . .	4- 56
4.13.4.2	iSUB Definings . . . . .	4- 57
4.13.4.3	String overlay Defining . . . . .	4- 58
4.14	DEFAULT RULES . . . . .	4- 59
4.14.1	Rejection Rules . . . . .	4- 60
4.14.2	Standard System Default Rules . . . . .	4- 60
SECTION 5.	DATA MANIPULATION . . . . .	5- 1
5.1	EXPRESSIONS . . . . .	5- 1
5.1.1	Scalar Expressions . . . . .	5- 1
5.1.2	Aggregate Expressions . . . . .	5- 1
5.1.2.1	Built-In Functions With Aggregate Arguments . . . . .	5- 1
5.1.2.2	Value of an Aggregate Expression . . . . .	5- 2
5.2	OPERATIONS ON EXPRESSIONS . . . . .	5- 2

5.2.1	Prefix Operations . . . . .	5- 2
5.2.2	Arithmetic Operations . . . . .	5- 2
5.2.2.1	Mixed Characteristics . . . . .	5- 3
5.2.2.2	Results of Arithmetic Operations. . . . .	5- 3
5.2.2.3	Infix Operators and Aggregate Operands. . . . .	5- 5
5.2.2.4	Integer Conversion. . . . .	5- 5
5.2.2.5	Arithmetic Base and Scale Conversion. . . . .	5- 5
5.2.2.6	Floating-Point to Fixed-Point Conversion. . . . .	5- 5
5.2.3	Comparison Operations . . . . .	5- 6
5.2.4	Bit String Operations . . . . .	5- 7
5.2.5	Concatenation Operations. . . . .	5- 8
5.2.6	Type Conversion . . . . .	5- 8
5.2.6.1	Bit String to Character String. . . . .	5- 8
5.2.6.2	Character String to Bit String. . . . .	5- 8
5.2.6.3	Character String to Arithmetic. . . . .	5- 9
5.2.6.4	Bit String to Arithmetic. . . . .	5- 9
5.2.6.5	Arithmetic to Character String. . . . .	5- 9
5.2.6.6	Arithmetic to Bit String. . . . .	5- 9
5.3	EVALUATION OF EXPRESSIONS . . . . .	5- 9
5.3.1	Order of Evaluation of Scalar Expressions . . . . .	5- 9
5.3.2	Order of the Evaluation of Aggregate Expressions. . . . .	5- 10
SECTION 6.	PROGRAM STRUCTURE. . . . .	6- 1
6.1	Statements. . . . .	6- 1
6.1.1	Simple Statements . . . . .	6- 1
6.1.2	Compound Statements . . . . .	6- 1
6.1.3	Label Prefixes. . . . .	6- 2
6.2	GROUPS. . . . .	6- 2
6.3	BLOCKS. . . . .	6- 3
6.4	CONDITION PREFIXES. . . . .	6- 6
6.5	PROGRAMS. . . . .	6- 7
SECTION 7.	STATEMENTS . . . . .	7- 1
7.1	CLASSIFICATION OF STATEMENTS. . . . .	7- 1
7.1.1	Assignment Statement. . . . .	7- 1



7.1.2	Control Statements . . . . .	7- 1
7.1.3	Program Structure Statements . . . . .	7- 1
7.1.4	Data Declaration Statement . . . . .	7- 1
7.1.5	Error Control and Debug Statements . . . . .	7- 1
7.1.6	Input/Output Statements . . . . .	7- 2
7.1.6.1	File Preparation Statements . . . . .	7- 2
7.1.6.2	Record Status Statements . . . . .	7- 2
7.1.6.3	Data Specification Statements . . . . .	7- 2
7.1.6.4	Data Transmission Statements . . . . .	7- 2
7.1.7	Storage Allocation Statements . . . . .	7- 2
7.1.8	System Attribute Statements . . . . .	7- 2
7.1.9	Null Statement . . . . .	7- 3
7.2	SEQUENCE OF CONTROL . . . . .	7- 3
7.3	LIST OF STATEMENTS BY CLASSIFICATION . . . . .	7- 4
7.3.1	The Assignment Statement . . . . .	7- 4
7.3.1.1	Scalar Assignments . . . . .	7- 5
7.3.1.2	Aggregate Assignments . . . . .	7- 6
7.3.2	Control Statements . . . . .	7- 9
7.3.2.1	The CALL Statement . . . . .	7- 9
7.3.2.2	The DELAY Statement . . . . .	7- 10
7.3.2.3	The DO Statement . . . . .	7- 10
7.3.2.4	The EXIT Statement . . . . .	7- 14
7.3.2.4A	The FREEZE Statement . . . . .	7- 14
7.3.2.5	The GO TO Statement . . . . .	7- 14
7.3.2.6	The IF Statement . . . . .	7- 16
7.3.2.7	The RETURN Statement . . . . .	7- 17
7.3.2.8	The SORT Statement . . . . .	7- 18
7.3.2.9	The STOP Statement . . . . .	7- 20
7.3.2.10	The WAIT Statement . . . . .	7- 21
7.3.3	Program Structure Statements . . . . .	7- 21
7.3.3.1	The BEGIN Statement . . . . .	7- 21
7.3.3.2	The END Statement . . . . .	7- 22
7.3.3.3	The ENTRY Statement . . . . .	7- 22

7.3.3.4	The PROCEDURE Statement . . . . .	7- 23
7.3.4	Data Declaration Statements . . . . .	7-24- A
7.3.4.1	The DECLARE Statement . . . . .	7- 25
7.3.4.1.1	Declaration of Structures . . . . .	7- 26
7.3.4.1.2	Factoring in DECLARE Statements . . . . .	7- 26
7.3.4.1.3	Multiple Declarations . . . . .	7- 27
7.3.4.2	DEFAULT Statement . . . . .	7- 27
7.3.5	Error Control and Debug Statements . . . . .	7- 30
7.3.5.1	The DUMP Statement . . . . .	7- 30
7.3.5.2	The ON Statement . . . . .	7- 30
7.3.5.3	The REVERT Statement . . . . .	7- 31
7.3.5.4	The SIGNAL Statement . . . . .	7- 33
7.3.6	Input/Output Statements . . . . .	7- 34
7.3.6.1	The CLOSE Statement . . . . .	7- 34
7.3.6.2	The DELETE Statement . . . . .	7- 35
7.3.6.3	The DISPLAY Statement . . . . .	7- 36
7.3.6.4	The FORMAT Statement . . . . .	7- 36
7.3.6.5	The GET Statement . . . . .	7- 37
7.3.6.6	The LOCATE Statement . . . . .	7- 39
7.3.6.7	The OPEN Statement . . . . .	7- 40
7.3.6.8	The PUT Statement . . . . .	7- 43
7.3.6.9	The READ Statement . . . . .	7- 44
7.3.6.10	The REWRITE Statement . . . . .	7-46- A
7.3.6.11	The WRITE Statement . . . . .	7-46- A
7.3.7	Storage Allocation Statements . . . . .	7- 47
7.3.7.1	The ALLOCATE Statement . . . . .	7- 47
7.3.7.2	The FREE Statement . . . . .	7- 51
7.3.8	System Attribute Statements . . . . .	7- 53
7.3.8.1	The System File Attribute Assignment Statement . . . . .	7- 53
7.3.8.2	The System File Attribute Reference Statement . . . . .	7- 53
7.3.8.3	The System Task Attribute Assignment Statement . . . . .	7- 54
7.3.8.4	The System Task Attribute Reference Statement . . . . .	7- 54
7.3.8.5	The System Library Attribute Assignment Statement . . . . .	7- 54

7.3.8.6	The System Library Attribute Reference Statement . . . . .	7- 55
7.3.9	The Null-Statement . . . . .	7- 55
SECTION 8.	INPUT/OUTPUT . . . . .	8- 1
8.1	FILE ATTRIBUTES . . . . .	8- 1
8.1.1	Merging of Attributes . . . . .	8- 2
8.1.2	Valid Combinations for File Attributes . . . . .	8- 3
8.2	Opening a File . . . . .	8- 3
8.2.1	Explicit Opening . . . . .	8- 4
8.2.2	Implicit Opening . . . . .	8- 4
8.3	Closing a File . . . . .	8- 4
8.4	Stream Transmission . . . . .	8- 4
8.4.1	Statements . . . . .	8- 4
8.4.2	Modes of Stream Transmission . . . . .	8- 6
8.4.2.1	List-Directed Transmission . . . . .	8- 6
8.4.2.2	Data-Directed Transmission . . . . .	8- 6
8.4.2.3	Edit-Directed Transmissions . . . . .	8- 6
8.4.3	Data Lists . . . . .	8- 7
8.4.3.1	Repetitive Specification . . . . .	8- 7
8.4.3.2	Transmission of Data List Elements . . . . .	8- 9
8.4.4	List-Directed Data Specification . . . . .	8- 10
8.4.4.1	List-Directed Input Format . . . . .	8- 10
8.4.4.2	List-Directed Output Format . . . . .	8- 11
8.4.4.2.1	Coded Arithmetic Data . . . . .	8- 12
8.4.4.2.2	Numeric Character Data . . . . .	8- 14
8.4.4.2.3	Character String Data . . . . .	8- 14
8.4.4.2.4	Bit String Data . . . . .	8- 14
8.4.5	Data-Directed Data Specification . . . . .	8- 14
8.4.5.1	Data-Directed Data in the Stream . . . . .	8- 15
8.4.5.1.1	Data Directed Input . . . . .	8- 15
8.4.5.1.2	Data Directed Output . . . . .	8- 17
8.4.5.2	Length of Data Fields in Data-Directed Output . . . . .	8- 18
8.4.6	Edit-Directed Data Specification . . . . .	8- 19
8.4.6.1	Format Lists . . . . .	8- 20

8.4.6.2	Data Format Items . . . . .	8- 20
8.4.6.2.1	Fixed-Point Format Items . . . . .	8- 21
8.4.6.2.2	Floating-Point Format Items . . . . .	8- 22
8.4.6.2.3	Numeric Picture Format Item . . . . .	8- 23
8.4.6.2.4	Bit String Format Items . . . . .	8- 24
8.4.6.2.5	Character String Format Items . . . . .	8- 24
8.4.6.3	Control Format Items . . . . .	8- 25
8.4.6.3.1	Spacing Format Item . . . . .	8- 25
8.4.6.3.2	Positioning Format Items . . . . .	8- 25
8.4.6.3.3	Printing Format Items . . . . .	8- 26
8.4.6.3.4	Remote Format Item . . . . .	8- 26
8.5	RECORD TRANSMISSION . . . . .	8- 27
8.5.1	Record Transmission Operations . . . . .	8- 28
SECTION 9.	PROCEDURES . . . . .	9- 1
9.1	PARAMETERS . . . . .	9- 1
9.2	REFERENCES . . . . .	9- 2
9.2.1	Procedure References . . . . .	9- 2
9.2.1.1	Function References . . . . .	9- 2
9.2.1.2	Subroutine References . . . . .	9- 3
9.3	Procedure Reference Arguments . . . . .	9- 3
9.3.1	Evaluation of Argument Subscripts . . . . .	9- 4
9.3.2	Use of Dummy Arguments . . . . .	9- 4
9.3.3	Entry Names as Arguments . . . . .	9- 5
9.4	USE OF THE ENTRY ATTRIBUTE . . . . .	9- 7
9.5	CORRESPONDENCE OF PARAMETERS AND ARGUMENTS . . . . .	9- 8
9.6	ALLOCATION OF PARAMETERS . . . . .	9- 10
9.7	BUILT-IN FUNCTIONS . . . . .	9- 10
9.8	LIBRARIES . . . . .	9- 11
9.8.1	Introduction . . . . .	9- 11
9.8.2	Creating Libraries . . . . .	9- 11
9.8.3	Referencing Libraries . . . . .	9- 11
9.8.4	Changing Library Attributes . . . . .	9- 11
9.8.5	Entry Points and their Parameters . . . . .	9- 11

SECTION 10.	DYNAMIC PROGRAM STRUCTURE . . . . .	10- 1
10.1	PROLOGUES . . . . .	10- 1
10.2	ACTIVATION AND TERMINATION OF BLOCKS. . . . .	10- 2
10.2.1	Dynamic Descent . . . . .	10- 2
10.2.2	Dynamic Encompassing. . . . .	10- 3
10.2.3	The Environment of a Block Activation . . . . .	10- 3
10.2.4	The Environment of a Label Constant . . . . .	10- 4
10.3	GENERATION OF A VARIABLE. . . . .	10- 4
10.4	ALLOCATION OF DATA AND STORAGE CLASSES. . . . .	10- 5
10.4.1	Definitions and Rules . . . . .	10- 5
10.4.2	Storage Classes . . . . .	10- 5
10.4.2.1	The Static Storage Class. . . . .	10- 6
10.4.2.2	The Automatic Storage Class . . . . .	10- 6
10.4.2.3	The Controlled Storage Class. . . . .	10- 6
10.4.2.4	The Based Storage Class . . . . .	10- 8
10.5	INTERRUPT OPERATIONS. . . . .	10- 8
10.5.1	Purpose of the Condition Prefix . . . . .	10- 9
10.5.2	Scope of the Condition Prefix . . . . .	10- 9
10.5.3	Use of The ON Statement . . . . .	10- 10
10.5.4	System Interrupt Action . . . . .	10- 11
10.5.5	Use of the Revert Statement . . . . .	10- 13
10.5.6	Programmer-Named Conditions . . . . .	10- 14
10.5.7	Condition Built-In Functions and Pseudo Variables . . . . .	10- 14
SECTION 11.	COMPILE-TIME FACILITIES . . . . .	11- 1
11.1	PREPROCESSOR INPUT. . . . .	11- 1
11.1.1	Effects of Compile-Time Statements. . . . .	11- 1
11.2	PREPROCESSOR OUTPUT . . . . .	11- 2
11.2.1	Rescanning And Replacement of Compile-Time Identifiers. . . . .	11- 2
11.3	COMPILE-TIME VARIABLES. . . . .	11- 3
11.4	COMPILE-TIME EXPRESSIONS. . . . .	11- 3
11.5	COMPILE-TIME PROCEDURES . . . . .	11- 4
11.5.1	Declaring Compile-Time Procedures . . . . .	11- 5
11.5.2	Execution of Compile-Time Procedures. . . . .	11- 5

11.6	COMPILE-TIME BUILTIN-FUNCTIONS . . . . .	11- 6
11.7	COMPILE-TIME STATEMENTS . . . . .	11- 6
11.7.1	Activate and Deactivate Statements . . . . .	11- 6
11.7.2	Assignment Statement . . . . .	11- 7
11.7.3	Declare Statement . . . . .	11- 8
11.7.4	Do-Group . . . . .	11- 8
11.7.5	GO TO Statement . . . . .	11- 9
11.7.6	If Statement . . . . .	11- 9
11.7.7	Include Statement . . . . .	11- 10
11.7.8	Null Statement . . . . .	11- 11
11.7.9	Put Statement . . . . .	11- 11
APPENDIX 1.	BUILTIN FUNCTIONS . . . . .	A1- 1
APPENDIX 2.	CONDITIONS . . . . .	A2- 1
APPENDIX 3.	KEYWORD ABBREVIATIONS . . . . .	A3- 1
APPENDIX 4.	48-CHARACTER SET . . . . .	A4- 1
APPENDIX 5.	ERROR MESSAGES . . . . .	A5- 1
APPENDIX 6.	CONVERSION FROM IBM . . . . .	A6- 1
APPENDIX 7.	COMPILER CONTROL IMAGES . . . . .	A7- 1

**Assumption:**

The default <scope-attribute> is INTERNAL for any item with the BUILTIN attribute.

**4.7.4 Returns Attribute**

The <returns-attribute> may be specified in a DECLARE statement for an entry name that is used as a function reference within the scope of the declaration.

**Syntax**

```
<returns-attribute> ::= RETURNS (<attribute-list>);
```

**Semantics**

The <returns-attribute> specifies the attributes of the function value returned when the entry name is invoked as a function.

The attributes in the <attribute-list> must agree with the attributes specified explicitly or by default in the PROCEDURE statement or ENTRY statement to which the entry name is prefixed.

The <length-attribute> specifications are evaluated on entry to the block containing the RETURNS attribute specification. Such evaluated <returns-attribute>s form part of the environment of blocks contained within the block declaring the attribute and dynamically descendant from the block.

**Restrictions:**

For an internal function, the <returns-attribute> can be specified only in a DECLARE statement that is internal to the same block as the function procedure.

Only arithmetic, string, locator, picture, aligned and unaligned attributes can be specified in the attribute list.

**Assumptions:**

If an entry name has no <returns-attribute> specification, a RETURNS attribute is assumed.

If the entry name begins with any of the letters I thru N, the defaults are REAL, FIXED, BINARY with default precision. If the entry name begins with any other letter, the defaults are REAL, FLOAT, DECIMAL with default precision.

**4.7.5 Parameter Attribute**

The <parameter-attribute> denotes a data item that is a parameter to an entry point of a procedure.

**Syntax**

```
<parameter-attribute> ::= [PARAMETER]
```

**Semantics**

This attribute is explicitly specified for a variable by the appearance of the name of that variable in a <parameter-list> and does not need to be specified by the programmer.

**4.7.6 Library Attribute**

A library attribute specifies that the identifier being declared is a <library-identifier>.

**Syntax**

```
<library-attribute> ::=
    LIBRARY [( <system-library-attribute-list> )]
<system-library-attribute-list> ::=
    <system-library-attribute>
    [, <system-library-attribute> ] ...
<system-library-attribute> ::=
    INTNAME = <char-string-constant>
    TITLE = <char-string-constant>
    LIBPARAMETER = <char-string-constant>
```

**Semantics**

The TITLE, INTNAME and LIBPARAMETER attributes are EBCDIC string-type attributes. The INTNAME is the internal identifier for the library. The TITLE is the code file name of the library. The LIBPARAMETER transmits information from the user program to the selection procedures of library programs that provide dynamic entry point linkage.

The character strings specified for system library attributes must be constants. Specification of these attributes is optional; by default, the identifier being declared is used for the TITLE and the INTNAME. If the INTNAME is specified but the TITLE is not, the string specified for the INTNAME is used for the TITLE.

**4.7.7 Options Attribute**

The OPTIONS attribute defines an entry as an external library entry point.

**Syntax**

```
<options-attribute> ::=
    ENVIRONMENT/OPTIONS (LIBRARY=<library-identifier>
    [,ACTUALNAME=<char-string-constant>])
```



### Semantics

Use of an OPTIONS list that includes LIBRARY or ACTUALNAME implies the ENTRY and CONSTANT attributes for the identifier being declared. LIBRARY and ACTUALNAME are mutually exclusive with file attributes; this use implies that a file is being declared. The library must be named; the declared <library identifier> is used by default. The specification of an ACTUALNAME is optional.

### Example

In this example, calling the entry DO-THIS from the PL/I program causes the MCP to link the program to a library called YOUR-LIBRARY with an entry point called DO-THAT.

```

DECLARE LIBI LIBRARY
  (TITLE = "MATH/LIBRARY");

DECLARE MY LIBRARY LIBRARY
  (INTNAME = "YOUR-LIBRARY");

DECLARE DO-THIS ENTRY (BINARY FIXED, CHAR(*))
  RETURNS (CHARACTER(100) VARYING)
  OPTIONS (LIBRARY = MY-LIBRARY,
          ACTUALNAME = "DO-THAT");

```

## 4.8 FILE DATA DESCRIPTION

### Syntax

```

<file> ::=
  <attribute> |
  <function-attribute> |
  <transmission-attribute> |
  <print-attribute> |
  <key-attribute> |
  <access-attribute> |
  <environment-attribute>

```

**4.8.1 File Attribute**

The <file-attribute> specifies that the identifier being declared is a file name.

**Syntax**

<file-attribute> ::= FILE

**Assumptions:**

The <file-attribute> can be implied by any of the other file description attributes.

An identifier may be contextually declared with the <file-attribute> through its appearance in the <file-option> of any data transmission statement or in an ON statement for any data transmission condition.

The <scope-attribute> EXTERNAL is assumed for files that are not parameters.

represented by <expression5> through <expression8>, the second expansion would look like this:

```

NEXT:  E5 = <expression5>;
      .
      .
      V=E5;
LABEL3: IF ... THEN GO TO NEXT1;
      IF (<expression8> THEN;
      ELSE GO TO NEXT1;
      statement 1
      .
      .
      statement m
LABEL4: V=V+E7;
      GO TO LABEL3;
NEXT1: statement n

```

2. If the <while-expression> is omitted, the IF statement immediately preceding statement 1 in the expansion is omitted.
3. If TO <expression2> is omitted, the statement E2 = <expression2> and the IF statement identified by LABEL2 are omitted.
4. If both TO <expression2> and BY <expression3> are omitted, all statements involving E2 and E3, and the statement GO TO LABEL2 are omitted.
5. Although the above expansions show a specific order in which the BY and TO-clauses are evaluated, no specific ordering is defined by the language.

The <while-expression> specifies that before each associated execution of the do-group, the expression is evaluated and, if necessary, converted to give a bit-string value. If any bit in the resulting string has the value '1', the iteration continues. If all bits have the value '0', the iterations associated with the current specification are terminated.

In the <iteration-specification> list, <expression1> represents the starting value of the control variable, <expression3> represents the increment to be added to the control variable after each iteration of the statements in the do-group. <expression2> represents the terminating value of the control variable. Iteration terminates as soon as the value of the control variable passes its terminating value. When the last specification is completed, control passes to the statement following the do-group.

Control may, under any circumstances, be transferred into a do-group from outside the do-group if iteration is not specified, i.e., no <while-expression> or <iteration-expression> exists. If the do-group is iterative and a GO TO statement transfers control to a statement inside the group, the results are undefined unless the GO TO is an abnormal return from a block that has been activated from within the do-group.

The effect of allocating or freeing the control variable is undefined.

**Examples**

- 1 DO INDEX = Z WHILE (A>B), 5 TO 10 WHILE (A=B), 100;
2. DO I = 1 TO 9, 11 TO 20;
3. DO WHILE (P);
4. DO;
5. DO WHILE (TAX-DEDUCT < ESTTAX \* 4);
6. DO I = 1 TO 100 BY 3;
7. DO INDEX = V REPEAT(Z) WHILE(X<100);

**7.3.2.4 The EXIT Statement****Syntax**

```
<exit-statement> ::= EXIT;
```

**Semantics**

The EXIT statement causes immediate termination of the task that contains the statement and all tasks attached by this task. The execution of the EXIT statement in a major task is equivalent to a STOP statement.

**7.3.2.4A The FREEZE Statement****Syntax**

```
<freeze-statement> ::= FREEZE OPTIONS(<freeze-option>);
<freeze-option> ::= PERMANENT/TEMPORARY
```

**Semantics**

A program must be "frozen" at run time using the FREEZE statement in order to become a library. The FREEZE statement must occur in the outer block of the program.

**7.3.2.5 The GO TO Statement**

The GO TO statement causes control of a program to be transferred to the specified statement within the program.

**Syntax**

```
<go-to-statement> ::=
  {{GO TO}|GOTO} <statement-label-name>;
```

**Semantics**

If a label variable is specified, the GO TO statement has the effect of a multi-way switch. The value of the label variable is the label of the statement to which control is transferred when the GO TO statement is executed.

Since the label variable may have different values at each execution of the GO TO statement, control may not always pass to the same statement. (Example 2 illustrates a GO TO statement used as a multi-way switch.)

A GO TO statement cannot pass control to an inactive block or to another task.

If control is transferred into a do-group that specifies iteration, the results are undefined unless the GO TO specifies an abnormal return from a block that has been activated from within the do-group.

If a label variable is specified, the GO TO statement has the  
 effect of a multi-way switch. The value of the label variable is  
 the label of the statement to which control is transferred when the  
 GO TO statement is executed.

Since the label variable may have different values at each  
 execution of the GO TO statement, control may not always pass to  
 the same statement. [Example 1 illustrates a GO TO statement used  
 as a multi-way switch.]

A GO TO statement cannot pass control to an iterative block or to  
 another task.

If control is transferred into a do-group the specific iteration  
 the transfer is made to unless the GO TO specifies an abnormal  
 transfer from a block that has been covered from within the  
 do-group.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

entry names. If no <data-specification-attributes> are specified, arithmetic defaults are applied separately to each name.

Consider the Statement:

```
A:I: ENTRY;
```

This statement is equivalent to:

```
A: ENTRY;
I: ENTRY;
```

An ENTRY statement cannot be internal to a begin block or to a do-group that specifies iteration.

A condition prefix cannot be prefixed to an ENTRY statement.

#### 7.3.3.4 The PROCEDURE Statement

Syntax

```
<procedure-statement> ::=
  <entry-name>: ...PROCEDURE
    [( <parameter> [,<parameter>]...)]
    [OPTIONS (<option-list>)]
    [RECURSIVE]
    [RETURNS (<data-specification-attributes>)];

<option-list> ::= <option> [,<option>]...

<option> ::= MAIN / DOUBLE / SORTINPUT / SORTOUTPUT /
  EXPORT (<entry-name>[AS<char-string-constant>]
  [,<entry-name>[AS<char-string-constant>]]...)
```

Semantics

The OPTIONS, RECURSIVE, and RETURNS options may appear in any order.

The PROCEDURE statement has the following functions:

1. Identifies a portion of program text as a procedure.
2. Defines the primary entry point to a procedure.
3. Specifies the parameters for the primary entry point.
4. Defines any special attributes of the procedure.
5. Specifies the attributes of the value that is returned if the procedure is invoked as a function at the primary entry point.

The <parameter>s are names that specify the parameters of the entry point. When the procedure is invoked, a relationship is established between the arguments of the invocation and the parameters of the invoked entry point. (See Section 9.5, Correspondence of Parameter Arguments).

OPTIONS specifies a list of options. OPTIONS may be specified for any procedure. DOUBLE allows the use of double precision for

arithmetic calculations, and SORTINPUT and SORTOUTPUT are discussed in Section 7.3.2.8, The Sort Statement.

The EXPORT option may only be used in the main (that is, outer block) <procedure-statement>. This option specifies which entry points can be used by other programs that call the program as a library. The <entry-names> used in the EXPORT option must be declared in the outer block of the program.

RECURSIVE specifies that the procedure may be invoked recursively (i.e., it may invoke itself). RECURSIVE does not have to be specified as all procedures are recursive by default.

The <data-specification-attributes> permitted for the RETURNS attribute of a PROCEDURE statement are the arithmetic, string, AREA, OFFSET, and POINTER attributes. These attributes specify the characteristics of the value returned by the procedure when invoked as a function at the primary entry point. (This rule applies to each entry name prefixed to the PROCEDURE statement.) The value specified in the RETURN statement of the invoked procedure is converted to the specified <data-specification-attributes>.

If a PROCEDURE statement has more than one entry name, the first name is interpreted as the only label on the statement. Each subsequent entry name is interpreted as a separate ENTRY statement having an identical parameter list and the same <data-specification-attributes> as written in the PROCEDURE. This equivalence is true only after multiple closure has been resolved. Defaults for the <data-specification-attributes> are applied separately for each such ENTRY statement and for the resulting PROCEDURE statement.

#### Examples

1. A:I: PROCEDURE;

Is effectively the same as:

```
A: PROCEDURE;
I: ENTRY;
```

The equivalence applies only after multiple closure has been resolved.

```
2. B: PROCEDURE;
   DCL A ENTRY RETURNS(FIXED);
   .
   .
   D=A(X,Y);
   END B;
A: PROCEDURE (B,C) RETURNS(FIXED);
   .
   .
   RETURN (B*C + SIN (P));
   END A;
```

If procedure A is invoked as a function as it is in procedure B, then when control is returned to B, the expression (B\*C + SIN (P)); is evaluated, converted to fixed point, and the value assigned to D in procedure B.



3. PROGRAM: PROCEDURE OPTIONS(MAIN, EXPORT(E1, E2 AS "E"));

In the following example, E1 and E2 are library entry points.

```
E1: PROCEDURE
      :
      END E1;
```

```
A: PROCEDURE
      :
E2: ENTRY
      :
      END A;
```

#### 7.3.4 Data Declaration Statements

PROGRAM: (PROGRAM INFORMATION, EXCEPT TO AS \*E\*)  
In the following example: 01 and 02 are library entry points  
01 PROGRAM  
02 END  
PROGRAM  
03 ENTRY  
04 END  
7.3.4 Data Definition Statements

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

### 7.3.8 System Attribute Statements

#### 7.3.8.1 The System File Attribute Assignment Statement

The <system-file-attribute-assignment-statement> is used to set the value of a system file attribute for a file outside of a file declaration or an OPEN statement.

##### Syntax

```
<system-file-attribute-assignment-statement> ::=
  <system-file-attribute> (<file-designator>) =
  <scalar-expression>;
```

##### Semantics

Attempting to set a read-only attribute will generate a syntax error of level three (3) at compile-time and a diagnostic error at run-time.

Multiple assignments are not allowed.

If a file attribute has mnemonics, they must be used by specifying a string constant or a string variable which has been assigned the correct value. If a constant is used, a compile-time check is made to see if it is a valid mnemonic for the specified file attribute. If not an error of level three is given. If a variable is used, it is checked at run-time, and if it is an invalid mnemonic, a non-fatal run-time error is given.

#### 7.3.8.2 The System File Attribute Reference Statement

The <system-file-attribute-reference-statement> is used to reference the value of a system file attribute.

##### Syntax

```
<system-file-attribute-reference-statement> ::=
  {<scalar-variable> | <pseudo-variable>} =
  <attribute-name> (<internal-file-name>
  [,<scalar-expression> [,<scalar-expression>]]);
```

##### Semantics

The two <scalar-expression>s are interpreted as follows: if KIND = PORT or REMOTE, the first <scalar-expression> identifies the STATION number, and the second <scalar-expression> is not allowed. If KIND = DISK or PACK, one <scalar-expression> refers to row number; two <scalar-expression>s refer to row number followed by copy number.

An attempt to read a write-only attribute generates a syntax error of level three at compile-time and a diagnostic error at run-time.

### 7.3.8.3 The System Task Attribute Assignment Statement

The `<system-task-attribute-assignment-statement>` is used to set the value of a system attribute for a task.

#### Syntax

```

<system-task-attribute-assignment-statement> ::=
  <system-task-attribute> (<task-designator>) =
    <scalar-expression>;
<task-designator> ::= <task-variable>

```

#### Semantics

Attempting to set a read-only attribute will generate a syntax error of level three (3) at compile-time and a diagnostic at run-time.

Multiple assignments are not allowed.

If a task attribute has mnemonics, they may be used, but mnemonics are not required.

### 7.3.8.4 The System Task Attribute Reference Statement

The `<system-task-attribute-reference-statement>` is used to reference the value of a system task attribute.

#### Syntax

```

<system-task-attribute-reference-statement> ::=
  {<scalar-variable> | <pseudo-variable>} =
  <system-task-attribute> (<task-designator>);

```

#### Semantics

Attempting to read a write-only attribute will generate a syntax error of level three (3) at compile time and a diagnostic error at run time.

### 7.3.8.5 The System Library Attribute Assignment Statement

The `<system-library-attribute-assignment-statement>` sets the value of library attributes outside of a library declaration.

#### Syntax

```

<system-library-attribute-assignment-statement> ::=
  <system-library-attribute> (<library-identifier>) =
  <scalar-expression>;

```

**Semantics**

Attempting to set a read-only attribute generates a syntax error at compile time and a diagnostic error at run time.

Multiple assignments are not allowed.

**7.3.8.6 The System Library Attribute Reference Statement**

The <system-library-attribute-reference-statement> is used to reference the value of a system library attribute.

**Syntax**

```
<system-library-attribute-reference-statement> ::=
  <scalar-variable>/<pseudo-variable> =
  <system-library-attribute>(<library-identifier>);
```

**Semantics**

The values specified for the system library attributes are subject to some constraints:

1. The TITLE must be a valid file title as judged by the system; "\*", usercode, and ON PACKNAME specifications are allowed.
2. The INTNAME must be a single identifier.
3. The LIBPARAMETER may be any string and is limited only by the conventions of the particular library.

Attempting to read a write-only attribute generates a syntax error at compile time and a diagnostic error at run time.

**7.3.9 The Null-Statement**

The <null-statement> is an empty statement. The <null-statement> causes no action and does not affect sequential operations in any way.

**Syntax**

```
<null-statement> ::= ;
```

**Example**

```

.
.
ON OVERFLOW;
.
.
```

The <on-unit> is a <null-statement>.

THIS PAGE IS INTENTIONALLY LEFT BLANK FOR FORMATTING PURPOSES

## 9.8 LIBRARIES

### 9.8.1 Introduction

A library is a program which provides a set of procedural "entry points" which can be called on by "user programs". Any user can create and use libraries. (A detailed discussion of the usage of libraries is found in Chapter 20 of the B 5000/B 6000/B 7000 Series System Software Operational Guide, Volume 2. Familiarity with this material is assumed in the discussion that follows.)

### 9.8.2 Creating Libraries

The OPTION list of the main (that is, outer block) PROCEDURE statement must include the EXPORT option. The EXPORT option specifies which entry points the library makes available to the programs that call on the library. (Refer to Section 7.3.3.4, The PROCEDURE Statement, for a more detailed discussion of the EXPORT option.)

To become a library, a program must be "frozen" at run time using the FREEZE statement. (Refer to Section 7.3.2.4A, The FREEZE Statement, for a more detailed discussion of this statement.)

### 9.8.3 Referencing Libraries

To call a library, the user program declares library names and specifies which procedures within the program call a given library. Libraries must be explicitly declared using the LIBRARY attribute in a DECLARE statement. (Refer to Section 4.7.6, The LIBRARY Attribute, for a more detailed discussion of this attribute.)

An entry point name is associated with a library by adding the OPTIONS attribute to an entry point declaration. (Refer to Section 4.7.7, The OPTIONS Attribute, for a more detailed discussion of this attribute.)

### 9.8.4 Changing Library Attributes

Library attributes may also be set and tested dynamically using the <system-library-attribute-reference-statement>. (Refer to Section 7.3.8.6, The System Library Attribute Reference Statement, for a more detailed discussion of this statement.)

### 9.8.5 Entry Points and their Parameters

Parameters must be passed according to certain rules. When a PL/I program calls a library procedure written in another language, the library procedure expects to receive parameters by name or by reference and not by value. When a PL/I library is called by a program written in another language, parameters must not be passed by name.

Parameters to a library entry point may be any of the following types:

**Binary Fixed (p,q),**  
...q = 0, p <= 39  
...q = 0, 39 < p <= 78  
...q neq 0, p <= 39  
...q neq 0, 39 < p <= 78

**Decimal Fixed(p, q),**  
...q = 0, p <= 11  
...q = 0, 11 < p <= 23  
...q neq 0, q <= 11  
...q neq 0, 11 < p <= 23

**Binary Float(p),**  
...p <= 39  
...39 < p <= 78

**Decimal Float(p),**  
...p <= 11  
...11 < p <= 23

**Character(n)**  
**Character(\*)**  
**Character(n) Varying**  
**Character(\*) Varying**

**Bit(n), n <= 48**

**Transaction record**  
**Transaction record array**



## INDEX

ABS Arithmetic Built-In Function	A1- 14
<access-attribute>	4-36- A, 4- 39
<activate-statement>	11- 6
ACTIVATE statement	11- 2, 11- 6, 11- 7
ACTIVATION AND TERMINATION OF BLOCKS	10- 2
ADD Arithmetic Built-In Function	A1- 14
ADDR Based Storage Function	A1- 33
AFTER String Built-In Function	A1- 3
<alignment-attribute>	4- 50, 4- 51
ALL Array Manipulation Function	A1- 28
<allocate-statement>	7- 47
ALLOCATE statement	4- 27, 4- 47, 7- 15, 7- 47, 7- 48, 7- 52, 10- 6, 10- 8
<allocation>	7- 47
ALLOCATION Built-In Function	A1- 34
alphabetic characters	3- 1
alphanumeric character	3- 1
alternative attributes	7- 29
ANY Array Manipulation Function	A1- 28
<area>	7- 25
AREA	3- 6, 4- 2, 4- 29, 4- 30, 4- 32, 4- 60, 4- 61, 7- 5, 7- 24, 7- 48, 7- 49, 7- 52, A1- 38, A2- 6, A2- 9
<area-attribute>	4- 23, 4- 27, 4- 29
Area Data	3- 7
<area-expression>	4- 29
<area-reference>	4- 26, 4- 27, 4- 28
<argument>	7- 9, 9- 2
<argument-list>	4- 53
<arithmetic>	4- 7
ARITHMETIC BUILT-IN FUNCTIONS	A1- 14
<arithmetic-constant>	8- 10
Arithmetic Data	3- 5, 8- 12
Arithmetic Operators	3- 2
<arithmetic-type>	4- 60
Arithmetic Variables	3- 11
<array>	4- 7, 4- 43
<array-assignment>	7- 4, 7- 5, 7- 6, 7- 8
ARRAY MANIPULATION BUILT-IN FUNCTIONS	A1- 27
<array-variable>	7- 4
<assignment-statement>	7- 4
<asterisk>	4- 11, 4- 29, 4- 43, 4- 48
ATAN Mathematical Built-In Function	A1- 21
ATAND Mathematical Built-In Function	A1- 22
ATTRIB	A7- 2, A7- 4, A7- 13
<attribute>	4-36- A, 7- 25, 7- 27
<attribute-expression>	7- 27
<attribute-expression1>	7- 27
<attribute-expression2>	7- 27
<attribute-expression3>	7- 27
<attribute-expression1>	7- 27
<attribute-keyword>	7- 27
<attribute-list>	7- 27
<attribute-name>	4- 32, 4- 35, 11- 8
<attribute-set>	7- 53
AUTOMATIC storage class attribute	4- 60, 7- 26, 7- 27, 7- 29
AVAILABLE	10- 6
BASE	3- 5, 5- 3, 7- 30, A1- 14, A1- 15, A1- 17, A1- 19, A1- 20, A1- 29, 7- 40, 7- 42
<base attribute>	A1- 30
<base-attribute>	4- 8
<base-item>	4- 7, 4- 8, 4- 13
	4- 54

Based Allocation	7- 48
<based-attribute>	4- 49
BASED attribute	4- 11, 4- 44, 4- 47, 4- 51, 7- 25, 7- 48, 7- 50, 10- 8
<based-locator-variable>	4- 27
BASED STORAGE BUILT-IN FUNCTIONS	A1- 32
<based-variable>	4- 27, 7- 51, 7- 52
BASED variable	7- 39, 7- 40, 7- 52, 10- 8
BEFORE String Built-in Function	A1- 3
<begin-block>	6- 4
<begin-option-list>	7- 21
<begin-statement>	6- 4, 7- 21
BEGIN statement	6- 4, 7- 21, 7-21- A, 10- 2, 10- 9, A7- 29
BINARY	3- 5, 4- 8, 4- 9, 4- 32, 4- 35, 4-36- A, 4- 41, 4- 59, 4- 60, 4- 61, 5- 9, 7- 28, 7- 30, 9- 6, A1- 1, A1- 9, A1- 15, A3- 1
BINARY Arithmetic Built-In Function	A1- 15
binary digit	3- 1
<binary-fixed-point-constant>	3- 7
<binary-floating-point-constant>	3- 7
BIT	4- 3, 4- 5, 4- 10, 4- 32, 4- 53, 4- 60, 4- 61, 7- 19, 7- 20, 7- 46, 7-46- C, A1- 1, A1- 4, A1- 9
BIT String Built-In Function	A1- 4
<bit-string-format-item>	8- 24
<bit-string-operation>	5- 7
<block>	10- 14
<Boolean-valued-system-file-attribute>	7- 40
<bound>	4- 43
Braces	2- 2
Brackets	2- 2, A7- 11
Built-in Function Names	3- 4
<builtin-attribute>	4- 30, 4- 34
BY NAME option	7- 6, 7- 7
<call-statement>	7- 9, 9- 2
CALL statement	4- 2, 7- 3, 7- 9, 9- 3, 9- 7
CARDCOL	A7- 3, A7- 4
CEIL Arithmetic Built-In Function	A1- 15
CHAR String Built-In Function	A1- 5
CHARACTER	1- 1, 3- 1, 3- 2, 4- 3, 4- 10, 4- 32, 4-36- A, 4- 58, 4- 60, 4- 61, 7- 25, 7- 50, 8- 9, 10- 7, 11- 2, 11- 3, 11- 4, 11- 6, 11- 8, A1- 8, A3- 1, A4- 1
<character-picture-element>	4- 13
<character-picture-item>	4- 13
<character-picture-specification>	4- 12, 4- 13, 8- 24, 8- 25
<character-string>	7- 40
Character String Constants	3- 8
Character String Data	8- 14
<character-string-format-item>	8- 24
Character String Variables	3- 11
CHARACTERISTICS	A1- 14
CHECK	A2- 1, A7- 2, A7- 6
CLASSIFICATION OF CONDITIONS	A2- 1
<close-option>	7- 34
<close-statement>	7- 34
CLOSE statement	7- 2, 7- 34, 8- 4
Closing a File	8- 4
CODE	7- 30, A7- 2, A7- 6
Coded real fixed-point binary data	8- 13
Coded real floating-point binary data	8- 13
Coded real floating-point decimal data	8- 13
COLLATE String Built-In Function	A1- 5
COLUMN format item	8- 26
COL1	A7- 3, A7- 7, A7- 24
COL2	A7- 3, A7- 9, A7- 25
<comment>	3- 22

<comment-string>	3- 22
Comments	3- 22
<comparison-operation>	5- 6
Comparison Operators	3- 2
<compile-time assignment statement>	11- 7
<compile-time declare-statement>	11- 8
<compile-time do-group>	11- 9
<compile-time expression>	11- 4
<compile-time-expression>	11- 7, 11- 10
Compile-time expressions	11- 3
<compile-time GO TO statement>	11- 9
<compile-time IF statement>	11- 10
<compile-time null-statement>	11- 11
<compile-time procedure>	11- 4
compile-time procedure	11- 2, 11- 3, 11- 4, 11- 5, 11- 8, 11- 9
<compile-time PUT statement>	11- 11
COMPILE-TIME STATEMENTS	11- 1, 11- 6
<compile-time variable>	11- 7
<compile-time- variable>	11- 11
<compile-time-variable>	11- 11
compile-time variable	11- 3, 11- 6, 11- 7, 11- 9
<compound statement>	6- 1
COMPUTATIONAL BUILT-IN FUNCTIONS	A1- 3
<concatenation-operation>	5- 8
<condition>	7- 30, 7- 33
Condition	3- 5, 6- 6, 10- 9, 10- 14, A1- 1, A1- 30, A1- 31, A1- 32
CONDITION BUILT-IN FUNCTIONS	A1- 30
<condition-list>	7- 31
<condition-name>	6- 7, 10- 11
CONDITION-NAME	4- 2
<condition-prefix>	6- 7
<conditional compound statement>	6- 1
<config>	A1- 9
<constant>	4- 52, 8- 15
CONSTANT attribute	4- 3
<constant expression>	7- 18
<constant-expression>	4- 40
Constants	3- 7, 3- 8, 3- 9, 3- 10, 3- 13, 8- 10, 9- 3
CONSTANTS	3- 7, 3- 8, 3- 9
Contextual declarations	4- 2
Control Format Items	8- 25
Controlled Allocation	7- 47
<controlled-variable>	7- 51, 7- 52
CONTROLS	A7- 2, A7- 10
CONVERSION condition	4- 13, 8- 22, 10- 15, A1- 30, A1- 31, A1- 32, A1- 37
COPY String Built-In Function	A1- 5
COS Mathematical Built-In Function	A1- 22
COSD Mathematical Built-In Function	A1- 22
COSH Mathematical Built-In Function	A1- 22
COUNT Built-In Function	A1- 35
COUNTS option	A7- 30
<credit>	4- 14, 4- 21
<currency-character>	4- 19
<D digits>	8- 23
<data>	4- 7
Data Aggregates	3- 14
<data-attribute>	4- 50
<data-character>	3- 22
<data-description-attributes>	4- 7
<data-directed-data-in-stream>	8- 15
<data-directed-data-specification>	8- 14
Data Directed Output	8- 17
Data-directed transmission	8- 6

DATA ELEMENTS	3- 5
Data format items	8- 20
<data-list>	8- 5, 8- 7, 8- 10, 8- 11, 8- 14, 8- 15, 8- 17, 8- 19
DATA ORGANIZATION	3- 13
<data-specification>	7- 37, 7- 43, 8- 5
<data-specification-attributes>	7- 22, 7- 23, 7- 24
<data-type>	4- 60
DATAFIELD	10- 14, 10- 16, A1- 2, A1- 30, A2- 4
DATAFIELD Condition Built-In Function	A1- 30
DATE Built-In Function	A1- 35
<deactivate-statement>	11- 6
DEACTIVATE statement	11- 7
<debit>	4- 14, 4- 21
DECAT String Built-In Function	A1- 6
DECIMAL	3- 5, 4- 8, 4- 9, 4- 32, 4- 35, 4- 59, 4- 60, 4- 61, 5- 9, 7- 28, 7- 30, 11- 3, 11- 7, 11- 8, A1- 1, A1- 9, A1- 16, A3- 1
DECIMAL Arithmetic Built-In Function	A1- 16
Decimal digits	3- 1
<decimal-fixed-point-constant>	3- 7
<decimal-floating-point-constant>	3- 7
<decimal-specifier>	4- 14, 4- 16
<declaration>	7- 26
<declaration-list>	7- 26
DECLARATIONS	4- 1
<declare-statement>	7- 25, 7- 26
DECLARE statement	3- 15, 4- 1, 4- 2, 4- 35, 4- 46, 4- 47, 7- 25, 7- 26, 7- 47, 8- 2, 8- 4, 9- 1, 9- 7, 9- 11, 11- 2, 11- 3, 11- 5, 11- 7, 11- 8
<declared identifiers>	7- 26
<default-set>	7- 27
<default statement>	7- 27
DEFAULT statement	4- 60, 7- 27, 7- 28, 7- 29, 7- 30
DEFINED	4- 30, 4- 54, 4- 55, 4- 56, 4- 57, 4- 58, 4- 60, 4- 61, 7- 20, 7- 25, 7- 45, 9- 1, A3- 1
<defined-attribute>	4- 50, 4- 54
DEFINED attribute	4- 30, 4- 56, 4- 57, 7- 45, 9- 1
<delay-statement>	7- 10
DELAY statement	7- 10
<delete-statement>	7- 35
DELETE statement	4- 38, 4- 42, 7- 2, 7- 35, 7- 36, 8- 28
<description-list>	4- 32, 4- 34
<digit>	2- 1, 2- 2
DIM Array Manipulation Function	A1- 28
<dimension-attribute>	4- 43, 4- 45
DIMENSION attribute	4- 32, 9- 8
DIRECT file	7- 35, 8- 28
<display-statement>	7- 36
DISPLAY statement	7- 2, 7- 36
DIVIDE Arithmetic Built-In Function	A1- 16
<do-group>	6- 3
<do-statement>	2- 1, 6- 3, 7- 10
DO statement	3- 3, 6- 3, 7- 3, 7- 10, 7- 11, 7- 12, A1- 36
<dump-option>	7- 30
<dump-statement>	7- 30
DUMP statement	7- 30, A7- 11, A7- 12
<edit-directed-data-specification>	8- 19
Edit-directed transmission	8- 6
<element>	8- 7, 8- 8
<elementary-arithmetic-operation>	5- 3
ellipsis	2- 2
EMPTY built-in function	4- 29
<end-statement>	7- 22
END statement	6- 3, 6- 4, 7- 3, 7- 17, 7- 22, 10- 2, 11- 1, 11- 4
ENDFILE(<filename>)	A2- 3

ENDPAGE(<filename>) . . . . . A2- 3

<entry> . . . . . 4- 7, 4- 30

<entry-attribute> . . . . . 4- 30, 4- 31

ENTRY attribute . . . . . 4- 30, 4- 31, 4- 32, 7- 26, 9- 6, 9- 7, 9- 8, 9- 9

<entry constant> . . . . . 7- 18

Entry Constants . . . . . 3- 10

<entry-expression> . . . . . 4- 34

<entry-expressions> . . . . . 4- 32

Entry-Label Data . . . . . 3- 6

<entry-name> . . . . . 4- 53, 7- 9, 7- 22, 7- 23, 9- 2

entry name . . . . . 4- 5, 4- 24, 4- 31, 4- 32, 4- 34, 4- 35, 4- 51, 6- 6, 7- 22, 7- 24, 9- 1, 9- 2, 9- 4, 9- 5, 9- 6, 9- 7, 9- 9, 10- 4, 11- 2, 11- 5, 11- 6

<entry-statement> . . . . . 7- 22

ENTRY statement . . . . . 4- 1, 4- 2, 4- 32, 4- 35, 6- 4, 7- 1, 7- 3, 7- 22, 7- 23, 7- 24, 7- 29, 9- 1

<entry-type> . . . . . 4- 60

Entry Variables . . . . . 3- 12

ENVIRONMENT . . . . . 4- 36, 4- 40, 4- 42, 4- 61, 7- 18, 7- 34, 7- 35, 7- 40, 7- 41, 7- 42, 8- 1, 8- 3, A3- 1

<environment-attribute> . . . . . 4-36- A, 4- 40

ERF Mathematical Built-In Function . . . . . A1- 23

ERRLIST . . . . . A7- 2, A7- 12

ERRORS . . . . . A7- 2, A7- 12

<event-valued-attribute> . . . . . 7- 21

EXCEPT String Built-In Function . . . . . A1- 7

EXEC . . . . . A7- 3, A7- 13

<exit-statement> . . . . . 7- 14

EXIT statement . . . . . 7- 3, 7- 14, 10- 2

Explicit Declarations . . . . . 4- 1

<exponent> . . . . . 8- 22, 8- 23

<exponent-character> . . . . . 4- 14, 4- 22

<expression> . . . . . 4- 11, 4- 29, 4- 43, 7- 11, 7- 46, 8- 5, 9- 3, A1- 19, A1- 20

<expression 1> . . . . . 7- 10

<expression 2> . . . . . 7- 10

<expression 3> . . . . . 7- 10

<expression 4> . . . . . 7- 10

<expression1> . . . . . 7- 11, 7- 12, 7- 13, 8- 8

<expression2> . . . . . 7- 12, 7- 13, 8- 8

<expression3> . . . . . 7- 11, 7- 12, 7- 13, 8- 8

<expression4> . . . . . 8- 8

<expression5> . . . . . 7- 12, 7- 13

<expression8> . . . . . 7- 13

EXTERN . . . . . 4- 50, A7- 2, A7- 13

EXTERNAL attribute . . . . . 4- 1, 4- 4, 4- 5, 4- 49, A7- 13

<factor> . . . . . A1- 6, A1- 10

<file> . . . . . 4- 7, 4-36- A

<file-attribute> . . . . . 4-36- B

<file-attribute-list> . . . . . 8- 1

File Constants . . . . . 3- 10

<file designator> . . . . . 7- 40

<file-designator> . . . . . 7- 53

<file-exp> . . . . . 4- 42

<file expression> . . . . . 7- 18

<file name> . . . . . 8- 5

<file-name> . . . . . 2- 1, 7- 34, 7- 35, 7- 37, 7- 39, 7- 40, 7- 41, 7- 43, 7- 44, 7- 45, 7- 46, 7-46- A, 7-46- B, A1- 35, A1- 36

FILE option . . . . . 4- 2, 7- 34, 7- 41

<file-title> . . . . . 7- 40, 11- 10

<file-type> . . . . . 4- 60

File Variables . . . . . 3- 12

<filename> . . . . . A2- 3, A2- 4, A2- 5

FIXED	2- 2, 3- 6, 4- 5, 4- 6, 4- 8, 4- 9, 4- 26, 4- 32, 4- 35, 4-36- A, 4- 59, 4- 60, 4- 61, 5- 9, 7- 17, 7- 24, 7- 27, 7- 30, 8- 9, 8- 13, 8- 17, 9- 1, 9- 4, 9- 6, 9- 8, 9- 9, 11- 2, 11- 3, 11- 4, 11- 7, 11- 8, A1- 1, A1- 16, A1- 17
FIXED Arithmetic Built-in Function	A1- 16
<fixed-point-format-item>	8- 21
<fixed-point-picture>	4- 14
FIXEDOVERFLOW	6- 7, 10- 9, 10- 10, A2- 1, A2- 2, A3- 1
FLEVEL	A7- 3, A7- 14
FLOAT	2- 2, 3- 4, 3- 6, 4- 3, 4- 5, 4- 8, 4- 32, 4- 35, 4- 60, 4- 61, 5- 9, 7- 17, 7- 27, 7- 28, 7- 30, 8- 9, 9- 1, 9- 5, 9- 6, 9- 7, 9- 8, A1- 1, A1- 17, A6- 4
FLOAT Arithmetic Built-in Function	A1- 17
<floating-point-format-item>	8- 22
<floating-point-picture>	4- 14
<format-attribute>	4- 23, 4- 25
<format-designator>	4- 25
<format-item>	8- 20
Format Label Constants	3- 10
Format Label Variables	3- 13
<format-list>	7- 36, 7- 37, 8- 5, 8- 19, 8- 20
<format-list-item>	8- 20
<format-statement>	7- 36
FORMAT statement	7- 2, 7- 36, 7- 37, 8- 27
<free-statement>	7- 51
FREE statement	4- 29, 7- 47, 7- 51, 7- 52, 10- 6, 10- 8
FROM	4- 42, 7-46- A, 7-46- B, 7-46- C, 7- 47, 8- 28, A1- 1, A1- 7, A1- 8, A2- 8, A6- 1
<fully-prefixed-statement>	6- 7
<function-attribute>	4-36- A, 4- 37
<function-reference>	9- 2
GENERIC	4- 32, 4- 34, 7- 9, 9- 9
<generic-attribute>	4- 30, 4- 32
<generic-attribute-list>	4- 32
<generic-attributes>	4- 32
<generic-element>	4- 32
<get-statement>	7- 37
GET statement	7- 37, 7- 41, 8- 5, 8- 15, A2- 8
<go-to-statement>	7- 14
GO TO statement	7- 3, 7- 13, 7- 14, 7-14- A, 7- 15, 7- 37, 11- 4, 11- 9, 11- 10
<group>	6- 3
HBOUND Array Manipulation Function	A1- 29
<identifier>	2- 2, 3- 17, 3- 19, 3- 21, 6- 2, 7- 18, 7- 25, 7- 27, 7- 28, 7- 47, 10- 14, 11- 4, 11- 6, 11- 8, A2- 7
<identifiers>	6- 2
IDENTIFIERS	3- 3
<if-statement>	7- 16
IF statement	7- 13, 7- 16, 10- 9, 11- 9, 11- 10
IGNORE option	7- 45
Implicit Opening	8- 3, 8- 4
<in-attribute>	4- 23, 4- 28, 4- 29, 4- 30
<in-option>	4- 29, 4- 30
IN option	7- 48, 7- 49, 7- 52
INCLCOL	A7- 3, A7- 14
<include-file-name>	11- 10
<include-statement>	11- 10
INCLUDE statement	11- 6, 11- 10
INDEX option	7- 45, 7-46- B
INDEX String Built-In Function	A1- 8
<initial attribute>	11- 8

<initial-attribute>	4- 50, 4- 51, 11- 8
INITIAL attribute	4- 24, 4- 30, 4- 47, 4- 51, 4- 53, 7- 39
<initial-call>	4- 30, 4- 51, 4- 53, 11- 8
<initial-list>	4- 51, 4- 52, 4- 53
INPUT attribute	4- 37
<input-file-option>	8- 5
<input option>	7- 18
<input-option>	7- 18
INPUT/OUTPUT CONDITIONS	A2- 3
<input-string-option>	8- 5
<insertion-character>	4- 14, 4- 18
<integer-constant>	4- 9, 4- 13, 4- 23, 4- 54, 7- 25, 8- 20
INTERNAL attribute	4- 1, 8- 6
<internal-file-name>	7- 53
IOTIME Built-In Function	A1- 35
<item>	4- 52
<iteration-expression>	7- 10, 7- 11, 7- 12, 7- 13
<iteration factor>	8- 20
<iteration-factor>	4- 52, 8- 20
<iteration-specification>	4- 52, 7- 10, 7- 11, 7- 12, 7- 13
<key-attribute>	4-36- A, 4- 39
KEY(<filename>)	A2- 5
<key option>	7- 18
<key-option>	7- 18
KEY option	7- 35, 7- 45, 8- 28
KEYED attribute	4- 39, 7- 45, 8- 28
KEYFROM	4- 39, 4- 42, 7- 39, 7- 45, 7-46- A, 7-46- B, 7-46- C, 8- 28, A2- 9, A2- 10
KEYFROM option	7- 45, 7-46- B
KEYLENGTH	4- 40, 4- 41, 4- 42
KEYORDER	4- 40, 4- 42
KEYSPERENTRY	4- 41, 4- 42
KEYSTART	4- 41, 4- 42
KEYTO option	8- 28, A1- 32
KEYTYPE	4- 41
KEYWORDS	3- 4
<label>	6- 3, 6- 4, 6- 7, 7- 22, 7- 36, 7- 37, 11- 4, 11- 6, 11- 7, 11- 8, 11- 9, 11- 10, 11- 11
<label-attribute>	4- 23
LABEL attribute	4- 24, 9- 7
Label Data	3- 6
Label Prefixes	4- 1, 6- 2
<labeled-statement>	6- 2
<labell>	11- 9
LBOUND Array Manipulation Function	A1- 29
<length>	7- 25, A1- 4, A1- 5, A1- 8, A1- 10
<length-attribute>	4- 10, 4- 11, 4- 12, 4- 35
length of data fields	8- 18
LENGTH String Built-In Function	A1- 9
<letter>	7- 27, 7- 28
<level>	7- 25, 7- 26
Libraries	9- 11
Library Attribute	4- 36, 7- 54, 7- 55, 9- 11
<like-attribute>	4- 44, 4- 45, 4- 46
LIMIT	7- 15, A7- 3, A7- 16
LINECNT	A7- 3, A7- 17
LINENO Built-In Function	A1- 36
<list-directed-data-specification>	8- 10
List-Directed Input Format	8- 10, 8- 15
List-Directed Output Format	8- 11, 8- 18
List-Directed Transmission	8- 6, 8- 14
LISTP	A7- 3, A7- 18

LIST1	A7- 3, A7- 17, A7- 18, A7- 24, A7- 26, A7- 33
LIST2	A7- 3, A7- 17, A7- 18, A7- 19, A7- 25, A7- 26, A7- 29, A7- 33
<locate-statement>	7- 39
LOCATE statement	4- 27, 4- 40, 4- 47, 4- 48, 7- 2, 7- 39
<locator-attribute>	4- 23, 4- 26
Locator Data	3- 6
<locator-qualifier>	4- 27, 4- 47, 7- 51
LOG Mathematical Built-In Function	A1- 24
LOGIC option	A7- 30
LOG10 Mathematical Built-In Function	A1- 24
LOG2 Mathematical Built-In Function	A1- 24
LOW String Built-In Function	A1- 10
<mantissa>	8- 22
MATHEMATICAL BUILT-IN FUNCTIONS	A1- 21
MAX Arithmetic Built-In Function	A1- 17
<member-attribute>	4- 44, 4- 45
<memory option>	7- 18
<memory-option>	7- 18, 7- 19
MERGE	A7- 3, A7- 19, A7- 35
Merging of Attributes	8- 2
MIN Arithmetic Built-In Function	A1- 18
MISCELLANEOUS BUILT-IN FUNCTIONS	A1- 34
MOD Arithmetic Built-In Function	A1- 18
MODE	3- 6, 4- 42, 5- 3, A1- 14, A1- 15, A1- 17, A1- 19, A1- 20, A1- 29, A1- 30
<mode-attribute>	4- 7, 4- 8
MODELII	A7- 3, A7- 19
MULTIPLE	A7- 3, A7- 20
MULTIPLY Arithmetic Built-In Function	A1- 19
NAME(<filename>)	A2- 4
NEW	A7- 3, A7- 16, A7- 20
NEW1	A7- 3, A7- 20, A7- 24, A7- 35
NEW1SEQERR	A7- 3, A7- 20, A7- 21
NEW2	A7- 3, A7- 20, A7- 21, A7- 25
NEW2SEQERR	A7- 3, A7- 22
NOBINDINFO	A7- 3, A7- 22
NODUPLICATE	A2- 9
NONVARYING	4- 12, 4- 61, A3- 1
notation constant	2- 1
notation variable	2- 1, 2- 2
NULL Based Storage Built-In Function	A1- 33
<null-statement>	7- 55
NULL statement	11- 11
NULLO Based Storage Built-In Function	A1- 33
<number-of-digits>	4- 9
Numeric Binary Data	8- 14
Numeric Decimal Data	8- 14
<numeric-picture-element>	4- 14
<numeric-picture-format-item>	8- 23
<numeric-picture-item>	4- 14
<numeric-picture-specification>	4- 12, 4- 14, 8- 23, 8- 24
OFFER	7- 40, 7- 42
OFFSET	3- 6, 4- 2, 4- 26, 4- 27, 4- 29, 4- 32, 4- 60, 7- 6, 7- 22, 7- 24, 7- 25, 7- 30, 7- 52, 9- 9, A1- 2, A1- 33, A1- 34
<offset-attribute>	4- 26
OFFSET Based Storage Built-In Function	A1- 33
<offset-expression>	4- 27
<ON compound statement>	6- 1
<on-statement>	7- 30
ON statement	4-36- B, 7- 1, 7- 3, 7- 30, 7- 31, 7- 33, 10- 4, 10- 8, 10- 9, 10- 10, 10- 12, 10- 13, 10- 14





<pointer-expression>	4- 27
<position-attribute>	4- 54, 4- 55, 4- 58
<positioning-format-item>	8- 26
positioning format items	8- 25
<precision>	4- 32
Precision	3- 8, 4- 9
PRECISION	3- 6, 4- 9, 4- 32, 4- 60, 4- 61, 5- 3, 7- 25, 7- 28, 7- 30, 11- 3, 11- 7, 11- 8, A1- 1, A1- 14, A1- 15, A1- 17, A1- 19, A1- 20, A1- 29, A1- 30, A1- 31, A3- 1
<precision-attribute>	4- 7, 4- 9, 4- 13
<print-attribute>	4-36- A, 4- 38
PRINT attribute	4- 37, 4- 38, 4- 39, 8- 5, 8- 6, A1- 36
<printing-format-item>	8- 26
Problem Data	3- 5
<procedure-block>	6- 4
<procedure-reference>	9- 2
<procedure-statement>	6- 4, 7- 23, 7- 24
PROCEDURE statement	4- 1, 4- 32, 6- 4, 7- 3, 7- 22, 7- 23, 7- 24, 9- 1, 9- 3, 10- 9, 11- 1, 11- 2, 11- 4, 11- 5
PROCTIME Built-In Function	A1- 36
PROD Array Manipulation Function	A1- 29
PROGRAM CHECKOUT CONDITIONS	A2- 5
<program-control>	4- 7, 4- 23
PROGRAMMER NAMED CONDITIONS	A2- 7
PROLOGUES	10- 1
<pseudo-variable>	7- 4, 7- 10, 7- 53, 7- 54, 7- 55
<put-statement>	7- 43
PUT statement	4- 15, 4- 38, 7- 37, 7- 41, 7- 43, 7- 44, 8- 5, 8- 9, 8- 26, 8- 27, 11- 11, A2- 3, A2- 8
<qualified name>	3- 19
RANDOM Mathematical Built-In Function	A1- 24
<range-specification>	7- 27, 7- 28
READ INTO option	8- 28
READ operation	8- 28
<read-statement>	7- 44
READ statement	4- 2, 4- 4, 4- 27, 7- 44, 7- 45, 7- 46, 8- 3, 8- 28, 10- 4, 10- 8
REAL	4- 8, 4- 9, 4- 32, 4- 35, 4- 60, 4- 61, A1- 2, A1- 14
<real-arithmetic-constant>	3- 7
RECORD attribute	4- 38, 4- 39, 4- 40
RECORD data transmission	8- 1
RECORD(<filename>)	A2- 5
record operation statements	8- 27
<record-type>	4- 60
<refer-option>	4- 11, 4- 29, 4- 43, 4- 44, 4- 47, 4- 48, 4- 49
<remote-format-item>	8- 27
REPEAT String Built-In Function	A1- 10
<repetition-factor>	4- 13, 4- 14, 4- 15
<repetitive-specification>	8- 8
RESCAN option	11- 7
<return-statement>	7- 17
RETURN statement	7- 3, 7- 17, 7- 22, 7- 24, 7- 30, 10- 2, 10- 7, 10- 12, 11- 4, 11- 5
<returns-attribute>	4- 30, 4- 35
RETURNS attribute	4- 32, 4- 35, 9- 9, 10- 1
REVERSE String Built-In Function	A1- 10
<revert-statement>	7- 31
REVERT statement	7- 31, 7- 32, 10- 13
<rewrite-statement>	7-46- A
REWRITE statement	7- 2, 7-46- A, 8- 28
ROUND Arithmetic Built-In Function	A1- 19

<S-D digits> . . . . . 8- 23

SAVEDELETIONS . . . . . 4- 41, 4- 42

<scalar-area-variable> . . . . . 7- 47, 7- 51

<scalar-assignment> . . . . . 7- 4, 7- 5

<scalar-character-string-variable> . . . . . 7- 37, 7- 43, 7- 45, 8- 5

<scalar-character-variable> . . . . . 7- 36

<scalar-exp> . . . . . 4- 42

<scalar expression> . . . . . 8- 5

<scalar-expression> . . . . . 3- 17, 4- 52, 7- 4, 7- 5, 7- 10, 7- 16, 7- 17, 7- 34,  
7- 35, 7- 36, 7- 37, 7- 39, 7- 40, 7- 42, 7- 43, 7- 44, 7- 45, 7- 46,  
7-46- A, 7-46- B, 7-46- C, 7- 53, 7- 54, 8- 5, 8- 20, 8- 28

<scalar-expression list> . . . . . 7- 40

<scalar-expression-list> . . . . . 7- 42

Scalar Items . . . . . 3- 13

<scalar-locator-expression> . . . . . 4- 27, 4- 46, 7- 25

<scalar-locator-variable> . . . . . 7- 47

<scalar-name> . . . . . 4- 11, 4- 48, 4- 49

<scalar-pointer-variable> . . . . . 7- 39, 7- 40, 7- 44

<scalar-pseudo-variable> . . . . . 8- 8

<scalar-ptr-var> . . . . . 4- 42

<scalar-ptr-variable> . . . . . 4- 42

<scalar-ref> . . . . . 4- 42

<scalar-task-name> . . . . . 7- 9

<scalar-variable> . . . . . 4- 52, 7- 4, 7- 21, 7- 53, 7- 54, 7- 55, 8- 8, 8- 15

Scalar Variables . . . . . 3- 14

<scale-attribute> . . . . . 4- 7, 4- 8, 4- 13

<scale-factor> . . . . . 4- 9

<scope> . . . . . 4- 7

<scope-attribute> . . . . . 2- 3, 4- 35, 4-36- B, 4- 49, 4- 50

SCOPE attribute . . . . . 10- 7

SEG . . . . . A7- 3, A7- 23

SEGS . . . . . A7- 3, A7- 23

<selector-function> . . . . . 7- 27, 7- 28, 7- 29

SEPARATORS AND OTHER DELIMITERS . . . . . 3- 3

SEQ . . . . . 7- 41, A7- 3, A7- 4, A7- 7, A7- 9, A7- 14, A7- 24, A7- 31

SEQUENCE . . . . . 7- 3, A7- 24

SEQUENTIAL file . . . . . 8- 28

SEQ1 . . . . . A7- 3, A7- 24

SEQ2 . . . . . A7- 3, A7- 24, A7- 25

<set-option> . . . . . 4- 27

SET option . . . . . 7- 39, 7- 45, 7- 48, 7- 49, 8- 27

SIGN Arithmetic Built-In Function . . . . . A1- 20

<sign-character> . . . . . 4- 19

<sign-currency-character> . . . . . 4- 14

<signal-statement> . . . . . 7- 33

SIGNAL statement . . . . . 7- 3, 7- 31, 7- 33, 10- 15

<simple-character-string-constant> . . . . . 8- 10

Simple Names . . . . . 3- 17

<simple-statement> . . . . . 6- 1

SIN Mathematical Built-In Function . . . . . A1- 25

SIND Mathematical Built-In Function . . . . . A1- 25

SINGLE . . . . . A7- 3, A7- 26

<single-statement> . . . . . 6- 3

SINGLE1 . . . . . A7- 3, A7- 26

SINGLE2 . . . . . A7- 3, A7- 26, A7- 27

SINH Mathematical Built-In Function . . . . . A1- 25

SIXTY . . . . . A7- 3, A7- 27

<size-attribute> . . . . . 4- 16, 7-21- A, 8- 22, 8- 23, 10- 12, 10- 13, A2- 2

SIZE condition . . . . . 8- 26

SKIP format item . . . . . 7- 38, 7- 41, 7- 43, 7-46- B, A2- 3

SKIP option . . . . . 7- 18

<sort identifier> . . . . . 7- 18

<sort-identifier> . . . . . 7- 18

<sort option> . . . . . 7- 18

<sort-option> . . . . . 7- 18

<sort-statement>	7- 18
SORT statement	7- 18
<spacing-format-item>	8- 25
special characters	3- 1
<specification>	8- 8
<specification-string>	A1- 6, A1- 7
SQRT Mathematical Built-In Function	A1- 25
STACK	A7- 3, A7- 27
STATCONTROL	A7- 3, A7- 27, A7- 28, A7- 29
<statement>	6- 2, 6- 3, 6- 7
<statement-body>	6- 1
<statement-identifier>	6- 1
statement identifier	3- 3, 3- 4
<statement-label-constant>	4- 23, 4- 25
Statement Label Constants	3- 9
<statement-label-designator>	8- 27
<statement-label-name>	7- 14
Statement Label Variables	3- 11
STATIC attribute	4- 24, 4- 44, 4- 49, 4- 53
STATIC storage attribute	10- 6
STATISTICS	A7- 3, A7- 28, A7- 29, A7- 30
STMTNO	A7- 3, A7- 30
<stop-statement>	7- 21
STOP statement	7- 14, 7- 20, 10- 2
<storage>	4- 7, 4- 60
STORAGE	4- 46, 10- 5, A1- 32, A6- 1
<storage-attribute>	4- 44
<storage-class>	4- 46, 4- 49, 4- 60
Storage Classes	10- 5
<stored-data>	4- 60
STREAM attribute	4- 38, 4- 39, 4- 40, A1- 35
STREAM data transmission	4- 30, 8- 1
<stream-input-statement>	8- 5
<stream-output-statement>	8- 5
STREAM transmission	8- 6
<stream-type>	4- 60
<string>	4- 7, 4- 10, A1- 6, A1- 9, A1- 10, A1- 11, A1- 12, A1- 37
String attributes	4- 10
String Data	3- 6, 8- 14
STRING HANDLING BUILT-IN FUNCTIONS	A1- 3
<string-one>	A1- 13
STRING option	4- 15, 7- 38
<string-two>	A1- 13
<string-type>	4- 60
<string-type-attribute>	4- 10
STRINGRANGE	6- 7, 10- 9, 10- 10, A1- 11, A2- 1, A2- 6, A3- 1
<structure>	4- 7, 4- 44
Structure Arrays	3- 14, 3- 15
<structure-assignment>	7- 4, 7- 5, 7- 7
<structure-attribute>	4- 44, 4- 45
<structure-expression>	7- 4
<structure-variable>	4- 45, 4- 46, 7- 4
Structures	3- 14, 7- 26, A6- 2, A6- 3
SUBFILE	7- 34, 7- 40, 7- 41, 7- 44, 7- 46, 7-46- A, 7-46- B, 7-46- C
<subroutine-reference>	9- 2
<subscript>	3- 17, 3- 21
<subscripted-name>	3- 17
<subscripted-qualified-name>	3- 21
SUBSCRIPTRANGE	4- 55, 6- 7, 10- 9, 10- 11, A2- 5, A3- 1
SUBSTR Pseudo-Variable	A1- 37
SUBSTR String Built-In Function	A1- 11
SUM Array Manipulation Function	A1- 30
syntax rule	2- 2
SYSTEM ACTION CONDITIONS	A2- 7

<system-file-attribute> . . . . . 4- 40, 7- 40, 7- 53  
 <system-file-attribute-assignment-statement> . . . . . 7- 53  
 <system-file-attribute-reference-statement> . . . . . 7- 53  
 <system-file-attribute-specification> . . . . . 7- 40  
 <system-file-attribute-specification-list> . . . . . 7- 40  
 <system-task-attribute> . . . . . 7- 54  
 <system-task-attribute-assignment-statement> . . . . . 7- 54  
 <system-task-attribute-reference-statement> . . . . . 7- 54

TAN Mathematical Built-In Function . . . . . A1- 26  
 TAND Mathematical Built-In Function . . . . . A1- 26  
 TANH Mathematical Built-In Function . . . . . A1- 26  
 TAPECOL . . . . . A7- 3, A7- 31  
 Task Data . . . . . 3- 6  
 <task-designator> . . . . . 7- 54  
 TASK option . . . . . 4- 2, 7- 9  
 <task-variable> . . . . . 7- 54  
 The System Library Attribute Reference Statement . . . . . 7- 55, 9- 11  
 The System Task Attribute Reference Statement . . . . . 7- 54  
 TIME . . . . . 1- 1, 11- 1, 11- 3, 11- 4, 11- 6, A1- 2, A1- 34, A1- 35, A1- 36,  
 A7- 3, A7- 32, A7- 33  
 TIME Built-In Function . . . . . A1- 36  
 TITLE . . . . . 4- 36, 4-36- A, 4- 43, 7- 40, 7- 41, 7- 42, 7- 55, 8- 3, A7- 3,  
 A7- 12, A7- 33  
 TO-clause . . . . . 7- 11  
 TRACE . . . . . A7- 3, A7- 33, A7- 34  
 TRACEENTRYRS . . . . . A7- 3, A7- 34  
 TRACELABELS . . . . . A1- 12  
 TRANSLATE String Built-In Function . . . . . 4-36- A, 4- 37, 4- 38  
 <transmission-attribute> . . . . . 8- 9  
 Transmission of Data List Elements . . . . . A2- 4  
 TRANSMIT(<filename>) . . . . . A1- 20  
 TRUNC Arithmetic Built-In Function . . . . . A2- 4  
 A1- 20

UNDEFINEDFILE(<filename>) . . . . . A2- 4  
 UNDERFLOW . . . . . 6- 7, 10- 9, 10- 10, A2- 1, A2- 3, A3- 1  
 <unit-1> . . . . . 7- 16  
 <unit-2> . . . . . 7- 16  
 UNSPEC Pseudo-Variable . . . . . A1- 38  
 UNSPEC String Built-In Function . . . . . A1- 12  
 UPDATE attribute . . . . . 4- 37

<value> . . . . . 11- 11, A1- 4, A1- 5  
 <value-item> . . . . . 4- 52  
 <variable> . . . . . 4- 42, 7- 10, 7- 11, 7- 39, 7- 44, 7- 45, 7- 46, 7-46- A,  
 7-46- B  
 <variable-attribute> . . . . . 4- 50, 4- 54  
 VARIABLES . . . . . 1- 1, 3- 11, 11- 3, A1- 36  
 VARYING . . . . . 2- 2, 4- 12, 4-36- A, 4- 54, 4- 55, 4- 60, 5- 8, 7- 5, 11- 3,  
 11- 8, A3- 1  
 <varying-attribute> . . . . . 4- 10, 4- 11, 4- 12  
 VERIFY String Built-In Function . . . . . A1- 13  
 VOID . . . . . A7- 3, A7- 20, A7- 35  
 VOIDT . . . . . A7- 3, A7- 20, A7- 35  
 VOLUME . . . . . 7- 34, 7- 35

WAIT statement . . . . . 7- 21  
 WAITUPDATEIO . . . . . 4- 41  
 WARN . . . . . A7- 3, A7- 12, A7- 13, A7- 36  
 <while-expression> . . . . . 7- 10, 7- 11, 7- 13  
 <write-statement> . . . . . 7-46- A  
 WRITE statement . . . . . 7- 40, 7-46- A, 7-46- B, 7-46- C, 8- 28, 10- 8

XREF . . . . . A7- 3, A7- 36

<zero-suppression> . . . . . 4- 17  
<zero-suppression-character> . . . . . 4- 14, 4- 17