UNISYS

OS/3

1974 American Standard COBOL

Programming Quick-Reference Guide

Relative to Release Level 8.2

Priced Item

August 1987

Printed in U S America
UP-8612 Rev. 3

UNISYS

OS/3 1974 American Standard COBOL

Programming Quick-Reference Guide

Previous Title: OS/3 1974 American Standard COBOL Summary

Relative to Release Level 8.2

August 1987

Printed in U S America

Priced Item

UP-8612 Rev. 3

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

PAGE STATUS SUMMARY

ISSUE: Update B – UP-8612 Rev. 3
RELEASE LEVEL: 8.2 Forward

Part/Section	Page Number	Update Level
Cover		B
Title Page/Disclaimer		B*
PSS	1	B
Contents	1	Orig.
	1 thru 10	Orig.
	11, 12	A
	13 thru 30	Orig.
	31 thru 34	A
	35, 36	Orig.
	37	A
	38 thru 41	Orig.
	42	A
User Comment Form		

*New page

PAGE STATUS SUMMARY

CONTENTS

The SPERRY UNIVAC Operating System/3 (OS/3) American National Standard COBOL language is fully described in the *OS/3 1974 American National Standard COBOL programmer reference, UP-8613* (current version).

COBOL CHARACTER SET

COBOL Character Set in Collating Sequence					
Name	Symbol	Hex. Codes		80-Col. Card Code	Source Language Usage
		EBCDIC	ASCII		
Space	blank or 8	40	20	blank	Punctuation, editing
Period/ decimal point	.	4B	2E	12—8—3	Punctuation, editing
Less than	<	4C	3C	12—8—4	Relation
Left paren.	(	4D	28	12—8—5	Punctuation, grouping
Plus	+	4E	2B	12—8—6	Editing, arithmetic
Currency sign	$	5B	24	11—8—3	Editing
Asterisk, multiplication sign, comment line	*	5C	2A	11—8—4	Editing, arithmetic
Right paren.	)	5D	29	11—8—5	Punctuation, grouping
Semicolon	;	5E	3B	11—8—6	Punctuation
Minus, hyphen, continuation line	-	60	2D	11	Words, editing, arithmetic
Slash, division sign	/	61	2F	0—1	Arithmetic, editing
Comma	,	6B	2C	0—8—3	Punctuation, editing
Greater than	>	6E	3E	0—8—6	Relation
Apostrophe (1)	'	7D	27	8—5	Punctuation
Equal sign	=	7E	3D	8—6	Relation, punctuation
Quotation	"	7F	22	8—7	Punctuation

(1) *Single quote used as quotation mark for SPERRY UNIVAC extension to American National Standard COBOL.*

COBOL CHARACTER SET (cont)

COBOL Character Set in Collating Sequence					
Name	Symbol	Hex. Codes		80-Col. Card Code	Source Language Usage
		EBCDIC	ASCII		
Letters	A thru I	C1 thru C9	41 thru 49	12—1 thru 12—9	Words. (DB for editing)
Letters	J thru R	D1 thru D9	4A thru 52	11—1 thru 11—9	Words. (CR for editing)
Letters	S thru Z	E2 thru E9	53 thru 5A	0—2 thru 0—9	Words. (Z for editing)
Numbers	0 thru 9	F0 thru F9	30 thru 39	0 thru 9	Words. editing, arithmetic

NOTE:

Any character can be used as data, but the 1974 American National Standard COBOL source language uses only those shown.

SUMMARY NOTATION

- Key words (that is, words that result in action by the compiler) are capitalized and underscored.
- Optional words (that is, words included for readability only) are capitalized, but not underscored.
- Brackets [ ] enclose words, phrases, or clauses that may be omitted if their functions are not required.
- Braces { } indicate a mandatory choice of variant forms or functions.
- When braces or brackets enclose a portion of a format showing only one possibility, the function of the braces or brackets is to delimit that portion of a format to which a following ellipsis applies.
- Ellipses . . . indicate repetition of elements enclosed in the preceding pair of brackets or braces.
- The punctuation characters comma and semicolon are shown in some formats. They are optional and may be included or omitted by the user. In the source program, these two punctuation characters are interchangeable and either one may be used anywhere one of them is shown in the formats. Neither one may appear immediately preceding the first clause of an entry or paragraph.

SUMMARY NOTATION (cont)

If desired, a semicolon or comma may be used between statements in the procedure division.

- Lower case represents generic terms which must be supplied by the user.
- Periods must be used where shown and must also appear at the end of each paragraph. Statements which do not contain periods on the reference card must be followed by a period when used at the end of a paragraph.
- Level 2 module specifications for 1974 American National Standard COBOL are enclosed in boxes.
- Sperry Univac extensions to American National Standard COBOL are enclosed in dashed-line boxes.
- Default values are shown by shading.
- SAM and ISAM files apply only to 90/25, 90/30, 90/30B, and 90/40 systems.

RULES AND SUGGESTIONS FOR EFFICIENCY

1. Use legal abbreviations for reserved words to reduce compilation time, that is, PIC instead of PICTURE.
2. Use relational operators instead of relational clauses.
3. Avoid needless qualification and/or subscripting.
4. With ADD, SUBTRACT, IF, and MOVE:
   - use same size sending and receiving fields;
   - align decimal positions of sending and receiving fields.
5. Use indexing instead of subscripting whenever possible.

FIGURATIVE CONSTANTS

ZERO ZEROS ZEROES	0 or 0's; DISPLAY mode = code F0 (EBCDIC) or 30 (ASCII) COMPUTATIONAL mode = binary 0
QUOTE QUOTES	code 7F (EBCDIC) or 22 (ASCII); apostrophe is the generated character
HIGH-VALUE HIGH-VALUES	code FF (EBCDIC) or 7F (ASCII)
LOW-VALUE LOW-VALUES	code 00 (lowest value in collating sequence)
ALL literal	a sequence of any nonnumeric literal or figurative constant
SPACE SPACES	blank character(s); code (EBCDIC) or 20 (ASCII)

PICTURE SYMBOLS

Picture Symbol	Description	Special Picture Position
9	A numeric character. Used in combination with: P S V	None
S	An operational sign is associated with the data item. Used in combination with: P V 9 H	Can be preceded only by H. Only one S is permitted.
V	Assumed decimal point in data item. Used in combination with: Any symbol except A and X is redundant with P	Only one is permitted. Can precede leading P or follow trailing P.

PICTURE SYMBOLS (cont)

Picture Symbol	Description	Special Picture Position
P	Assumed decimal point outside of data item. Each P represents one character position. Used in combination with: Any symbol except A and X	Must be first or last symbol or symbols of PICTURE except for X, CR, D3, V, or single +, -, or $ but cannot be both first and last.
E	Signifies floating point. Used in combination with: V + - 9	Left of E is mantissa. Right of E is exponent.
A	An alphabetic character or space. Used in combination with: X 9 B 0	None
X	An alphanumeric character. Used in combination with: A 9 B 0	None
Z	Suppression of leading 0's (replaced by blanks or spaces). Used in combination with: Any symbol except * A X S H or more than one $ + or -	Can be preceded only by: V . , $ + - P B 0 (zero)
*	Check protection, replaces leading 0's with asterisks. Used in combination with: Any symbol except Z A X S H or more than one S - or +	Can be preceded only by: - + . , V S P B 0 (zero)

PICTURE SYMBOLS (cont)

Picture Symbol	Description	Special Picture Position
, (comma)	Insert comma in character position unless the preceding position has been blanked. Used in combination with: Any symbol except A X S H	None
. (period)	Actual decimal point to be inserted in character position unless following positions have been blanked. Used in combination with: Any symbol except A X P V S H	May not be last character
B	Insert a blank or space in character position. Used in combination with: Any symbol except S and H	None None
CR	Insert the two characters CR if data item is of negative value; insert two blanks or spaces if value is positive. Used in combiantion with: Any symbol except A X + - S D B H	Must be last symbol except for P or V.
DB	Insert the two characters DB if data item is of negative value; insert two blanks if value is positive. Used in combination with: Any symbol except A X + - S CR H	Must be last symbol except for P or V.

PICTURE SYMBOLS (cont)

Picture Symbol	Description	Special Picture Position
$ (currency sign)	Insert $ sign in character position. If more than one, indicates floating $ sign. Used in combination with: Any symbol except that one $ cannot be used with A X S H; more than one $ cannot be used with S H A X * Z or more than one + -	Must be first symbols when more than one except for single + or - P B 0 (zero). If only one used, it can only be preceded by + - or P or V.
0 (zero)	Insert 0 in character position. Used in combination with: Any symbol except S and H	None
+	Insert + in character position if data item value positive; - if value negative. If more than one +, indicates floating sign. Used in combination with: Any symbol except one + cannot be used with A X - S CR DB H; more than one consecutive + cannot be used with A X - S CR DB Z H * or more than one $ sign.	If only one + is used, if must be either first or last except for P or V. If more than one + is used, it must be first symbol except for the $ sign.

PICTURE SYMBOLS (cont)

Picture Symbol	Description	Special Picture Position
- (minus)	Insert - in character position if data item value negative, blank if positive. If more than one -, indicates floating sign. Used in combination with: Any symbol except one - cannot be used with A X + S CR DB * Z H or more than one $ sign.	If only one - is used, it must be either first or last except for P or V. If more than one - is used, it must be the first symbol except for the $ sign.

ARITHMETIC EXPRESSIONS - SEQUENCE OF SYMBOLS

First Symbol	Second Symbol				
	Variable (identifier or literal)	* / ** + -	unary + unary -	(	)
Variable (identifier or literal)		P			P
* / ** + -	P		P	P	
Unary + or unary -	P			P	
(	P		P	P	
)		P			P

P Permitted combination
blank Not permitted

ARITHMETIC OPERATORS

Operators		Meaning
Binary	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	**	Exponentiation
Unary	+	The effect of multiplication by numeric literal +1
	-	The effect of multiplication by numeric literal -1

ORDER OF EVALUATION FOR ARITHMETIC EXPRESSIONS

Complex Expressions:

Innermost to outermost nested parentheses

Simple Expressions:

1st	Unary plus and minus
2nd	Exponentiation
3rd	Multiplication and division
4th	Addition and subtraction

Evaluate from left to right for operators at the same level.

PERMITTED DATA CATEGORIES IN MOVE STATEMENTS

Sending Data Item	Receiving Data Item		
	Alphabetic	Alphanumeric	Numeric
Alphabetic	P	P	
Alphanumeric	P	P	P
Alphanumeric edited	P	P	
Numeric integer		P	P
Numeric noninteger			P
Numeric edited		P	

P Permitted
blank Not permitted

SPECIAL LEVEL NUMBERS

66	Entries using RENAMES clause to regroup data items
77	Entries specifying noncontiguous working storage and linkage data items
88	Entries that specify condition-names to be associated with specific values of a conditional variable

DATA-ITEM RELATIONSHIPS

Level of Item	Class	Category
Elementary	Alphabetic	Alphabetic
	Numeric	Numeric
	Alphanumeric	Numeric edited Alphanumeric edited Alphanumeric
Group	Alphanumeric	Alphabetic Numeric Numeric edited Alphanumeric edited Alphanumeric

PICTURE CHARACTERS

Data Character Symbols

A	Alphabetic
X	Alphanumeric
9	Numeric

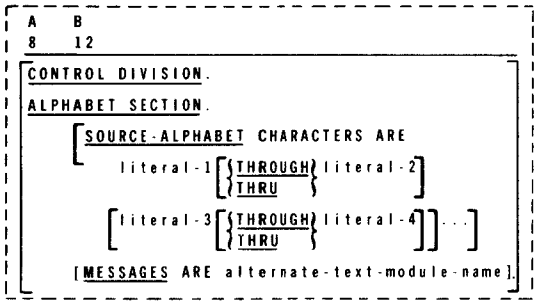
Operational Symbols

S	Operational sign
V	Assumed decimal point
P	Scale factor
E	Exponent

Editing Symbols

B	Space	*	Numeric char. or *
Z	Numeric char. or space	+	Plus or minus sign
0	Zero	-	Minus sign or space
/	Stroke char.		
,	Comma	CR	Credit symbol
.	Decimal point	DB	Debit symbol
		cs	Currency symbol

CONTROL DIVISION - BASIC FORMAT①



IDENTIFICATION DIVISION - BASIC FORMAT②

```
A    B
8    12
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.
[AUTHOR. [comment-entry] . . . ]
[INSTALLATION. [comment-entry] . . . ]
[DATE-WRITTEN. [comment-entry] . . . ]
[DATE-COMPILED. [comment-entry] . . . ]
[SECURITY. [comment-entry] . . . ]
```

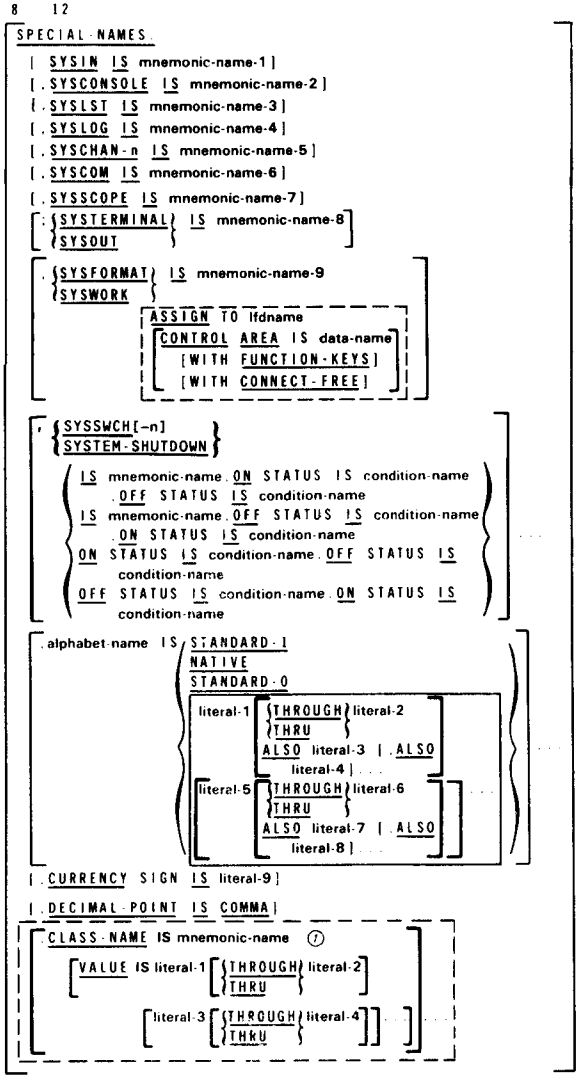
ENVIRONMENT DIVISION - BASIC FORMATS

```
A    B
8    12
ENVIRONMENT DIVISION.
     CONFIGURATION SECTION.
     SOURCE-COMPUTER. { SPERRY-OS3 }
                      { UNIVAC-OS3 }
         [WITH DEBUGGING MODE].
     OBJECT-COMPUTER. { SPERRY-OS3 }
                      { UNIVAC-OS3 }
         [,MEMORY SIZE integer { CHARACTERS }]
                               { MODULES    }
                               { WORDS      }
         [,PROGRAM COLLATING SEQUENCE IS
             alphabet-name]
         [,SEGMENT-LIMIT IS segment-number] .
```

① *The control division is required only when the non-English language feature is used.*

② *Comments may be included in this division by entering an asterisk (*) in column 7 of each line of comment coding.*

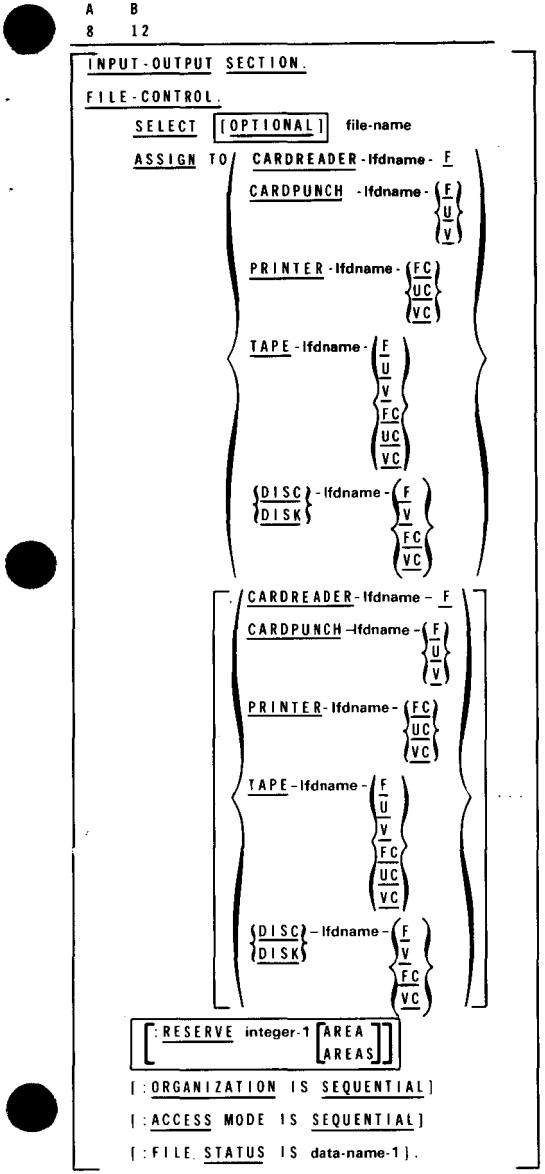
ENVIRONMENT DIVISION - Basic Formats (cont)



(1) *This clause is used when non-English language feature is invoked.*

ENVIRONMENT DIVISION -
Basic Formats (cont)

FORMAT 1 (Sequential Files):



ENVIRONMENT DIVISION -
Basic Formats (cont)

FORMAT 2 (Relative Files):

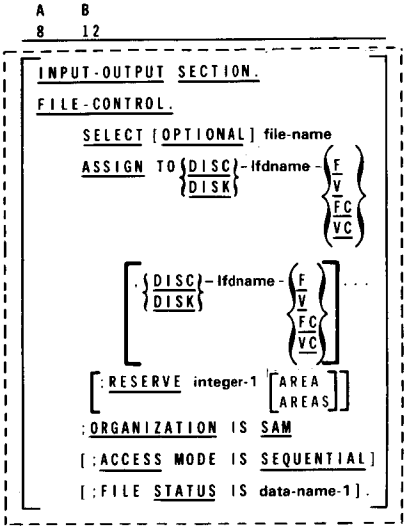
```
A    B
8    12
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT file-name
     ASSIGN TO {DISC} - lfdname - {F}
               {DISK}           {V}
        [, {DISC} - lfdname - {F} ]...
           {DISK}           {V}
     [:RESERVE integer-1 [AREA ]]
                         [AREAS]
     :ORGANIZATION IS RELATIVE
     [:ACCESS MODE IS {SEQUENTIAL [, RELATIVE KEY IS data-name-1 ]}
                      {RANDOM     }
                      {DYNAMIC    }  , RELATIVE KEY IS data-name-1 ]
     [:FILE STATUS IS data-name-2].
```

FORMAT 3 (Indexed Files):

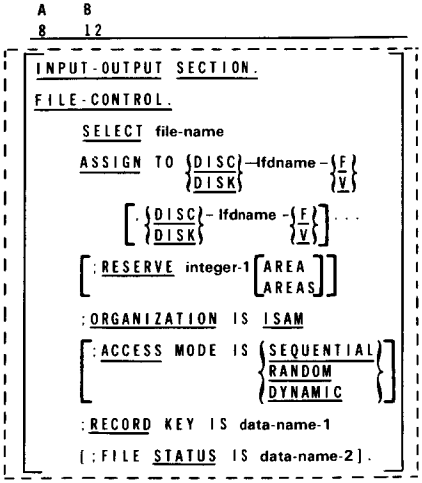
```
A    B
8    12
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT file-name
     ASSIGN TO {DISC} - lfdname - {F}
               {DISK}           {V}
        [, {DISC} - lfdname - {F} ]...
           {DISK}           {V}
     [:RESERVE integer-1 [AREA ]]
                         [AREAS]
     :ORGANIZATION IS INDEXED
     [:ACCESS MODE IS {SEQUENTIAL}]
                      {RANDOM    }
                      {DYNAMIC   }
     :RECORD KEY IS data-name-1
     [:ALTERNATE RECORD KEY IS data-name-2
        [WITH DUPLICATES]]...
     [:FILE STATUS IS data-name-3].
```

ENVIRONMENT DIVISION - Basic Formats (cont)

FORMAT 4 (SAM* Files):



FORMAT 5 (ISAM* Files):



**Applies only to 90/25, 90/30, 90/30B, and 90/40 systems.*

ENVIRONMENT DIVISION - Basic Formats (cont)

FORMAT 6 (Sort or Merge Files):

```
A   B
8   12
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT file-name
    ASSIGN TO {DISC} -lfdname- {F}
              {DISK}           {V}
              {TAPE}
        [{DISC} -lfdname- {F}]...
         {DISK}           {V}
         {TAPE}
[I-O-CONTROL.
    [;RERUN ON {{DISC} -lfdname- {1}}
               {{DISK}           {2}}
               {{TAPE}           }
               {[lfdname]        }
        EVERY integer-1 RECORDS OF file-name-1]...
    [;SAME [RECORD]      ] AREA FOR file-name-2 ...
           [SORT]
           [SORT-MERGE]
        {,file-name-3}...
    [;MULTIPLE FILE TAPE CONTAINS file-name-4
        [POSITION integer-2]
     [,file-name-5 [POSITION integer-3] ] ... ] ...
    [;APPLY BLOCK-COUNT ON                    ]...
        {file-name-6 [file-name-7]...}
        {TAPES                    }
    [;APPLY CYLINDER-INDEX AREA OF integer-4
        INDICES ON file-name-8 [,file-name-9] ...]...
    [;APPLY CYLINDER-OVERFLOW AREA OF integer-5
        PERCENT ON file-name-10 [,file-name-11] ...]...
    [;APPLY VERIFY ON file-name-12
        [,file-name-13]...]...
    [;APPLY INDEX-AREA OF integer-6
    CHARACTERS ON file-name-14
    [,file-name-15] ...] ...]   .]
```

DATA DIVISION - BASIC FORMATS

```
A   B
8   12
DATA DIVISION.
[FILE SECTION.]
FD file-name
    [;BLOCK CONTAINS [integer-1 TO] integer-2
        {CHARACTERS}
        {RECORDS   }]
    [;RECORD CONTAINS [integer-3 TO] integer-4
        CHARACTERS]
    ;LABEL {RECORD IS  } {STANDARD   }
           {RECORDS ARE} {OMITTED    }
                         {[data-name-0]}
```

```
A    B
8    12
```

```
[ [ ,VALUE OF
    ( FILE-ID IS { data-name-1 }
                 { literal-1   }
      [ [ ,PASSWORD IS { data-name-2 } ] ]
                       { literal-2   }
    | PASSWORD IS { data-name-2 }
                  { literal-2   }
      [ ,FILE-ID IS { data-name-1 } ]
                    { literal-1   }
    ) ] ]
[ [ ,DATA { RECORD IS  } data-name-3 [ ,data-name-4 ] ... ] ]
          { RECORDS ARE }
[ [ ,LINAGE IS { data-name-5 } LINES                    (1)
               { integer-5   }
    [ ,WITH FOOTING AT { data-name-6 } ]
                       { integer-6   }
    [ ,LINES AT TOP { data-name-7 } ]
                    { integer-7   }
    [ ,LINES AT BOTTOM { data-name-8 } ] ] ]
               { integer-8   }
[ LINAGE IS SYSTEM LINES ]
[ ,CODE-SET IS alphabet-name ]  (1)
```

```
SD file-name
   [ ,RECORD CONTAINS [ integer-1 TO ]
       integer-2 CHARACTERS ]
   [ ,DATA { RECORD IS  } data-name-1 [ ,data-name-2 ] ... ]
           { RECORDS ARE }
```

Data Description Clauses

FORMAT 1:

```
A    B
8    12
```

```
level-number { data-name-1 } (2)
             { FILLER      }
   [ ,REDEFINES data-name-2 ]
   [ [ { PICTURE } IS character-string ] ]
       { PIC     }
   [ [ ,[ USAGE IS ] ( COMPUTATIONAL
                     | COMP
                     | COMPUTATIONAL-1
                     | COMP-1
                     | COMPUTATIONAL-2
                     | COMP-2
                     | COMPUTATIONAL-3
                     | COMP-3
                     | COMPUTATIONAL-4
                     | COMP-4
                     | DISPLAY
                     | INDEX ) ] ]
```

(1) *Used only with sequential files*

(2) *Level-number 01 is the only level-number that must be started in margin A. Other level-numbers and these data description clauses may begin in either margin A or B.*

DATA DIVISION - Basic Formats (cont)

```
A    B
8    12
```

```
[ : [ SIGN IS ] { LEADING  } [ SEPARATE CHARACTER ] ]
                { TRAILING }

[ : OCCURS
      { integer-1 TO integer-2 TIMES                  }
      {     DEPENDING ON data-name-3                  }
      { integer-2 TIMES                               }
    [ { ASCENDING  } KEY IS data-name-4
      { DESCENDING }         [ , data-name-5 ] . . . ] . . .
    [ INDEXED BY index-name-1 [ , index-name-2 ] . . . ] ]

[ { SYNCHRONIZED } [ { LEFT  } ] ]
  { SYNC         }   { RIGHT }

[ : { JUSTIFIED } RIGHT ]
  { JUST      }

[ : BLANK WHEN ZERO ]

[ : VALUE IS literal ] .
```

FORMAT 2:

```
66 data-name-1 ; RENAMES data-name-2
     [ { THROUGH } data-name-3 ]
     { THRU    }
```

FORMAT 3:

```
88 condition-name ; { VALUE IS  } literal-1
                    { VALUES ARE }

     [ { THROUGH } literal-2 ]
     { THRU    }

     [ , literal-3 [ { THROUGH } literal-4 ] ] . . . .
                   { THRU    }
```

```
[ WORKING-STORAGE SECTION
[ 77 data-name ;
     ( data description clauses ) . ]
[ 01 record-name ;
     ( subordinate data items and clauses ) . ] ]

[ LINKAGE SECTION .
[ 77 data-name ;
     ( data description clauses ) . ]
[ 01 record-name ;
     ( subordinate data items and clauses . ] ]
```

```
A    B
8    12
[COMMUNICATION SECTION]
[CD cd-name; FOR [INITIAL] INPUT
    [[;SYMBOLIC QUEUE IS data-name-1]
        [;SYMBOLIC SUB-QUEUE-1 IS data-name-2]
        [;SYMBOLIC SUB-QUEUE-2 IS data-name-3]
        [;SYMBOLIC SUB-QUEUE-3 IS data-name-4]
        [;MESSAGE DATE IS data-name-5]
        [;MESSAGE TIME IS data-name-6]
        [;SYMBOLIC SOURCE IS data-name-7]
        [;TEXT LENGTH IS data-name-8]
        [;END KEY IS data-name-9]
        [;STATUS KEY IS data-name-10]
        [;MESSAGE COUNT IS data-name-11]]
    [data-name-1, data-name-2, . . . , data-name-11]
[CD cd-name; FOR OUTPUT
    [;DESTINATION COUNT IS data-name-1]
    [;TEXT LENGTH IS data-name-2]
    [;STATUS KEY IS data-name-3]
    [;DESTINATION TABLE OCCURS integer-2 TIMES
        [;INDEXED BY index-name-1 [,index-name-2]...]]
    [;ERROR KEY IS data-name-4]
    [;SYMBOLIC DESTINATION IS data-name-5]
```

PROCEDURE DIVISION - BASIC FORMATS

FORMAT 1:

```
PROCEDURE DIVISION [USING data-name-1
  [,data-name-2]...].
[DECLARATIVES.
{section-name SECTION [segment-number].
  declarative-sentence
[paragraph-name. [sentence]...]...}...
END DECLARATIVES.]
{section-name SECTION [segment-number].
[paragraph-name. [sentence]...]...}...
```

PROCEDURE DIVISION - Basic Formats (cont)

FORMAT 2:

```
PROCEDURE DIVISION [USING data-name-1
  [ ,data-name-2] . . . ].

[paragraph-name.  [sentence] . . . ] . . .
```

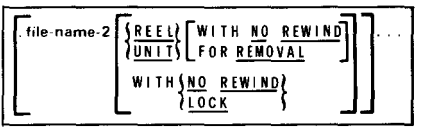
COBOL Verbs

```
ACCEPT identifier [FROM mnemonic-name]

ACCEPT identifier FROM { DATE }
                       { DAY  }
                       { TIME }

ACCEPT cd-name MESSAGE COUNT

ACCEPT identifier-1 [ ,identifier-2] . . .
       FROM [SPECIFIC] mnemonic-name
       [USING { identifier-3 }]
              { literal      }
       [ON EXCEPTION imperative-statement]

ACCEPT identifier-1 FROM mnemonic-name
       [ON EXCEPTION imperative-statement]

ADD { identifier-1 } [ ,identifier-2 ] . . .
    { literal-1    } [ ,literal-2    ]
  TO identifier-m [ROUNDED]
  [ ,identifier-n [ROUNDED]] . . .
  [;ON SIZE ERROR imperative-statement]

ADD { identifier-1 } , { identifier-2 } [ ,identifier-3 ] . . .
    { literal-1    }   { literal-2    } [ ,literal-3    ]
  GIVING identifier-m [ROUNDED]
  [ ,identifier-n [ROUNDED]] . . .
  [;ON SIZE ERROR imperative-statement]

ADD { CORRESPONDING } identifier-1 TO identifier-2
    { CORR          }
  [ROUNDED]
[;ON SIZE ERROR imperative-statement]

ALTER procedure-name-1 TO [PROCEED TO]
  procedure-name-2
  [ ,procedure-name-3 TO [PROCEED TO]
    procedure-name-4] . . .

CALL { identifier-1 } [USING { data-name-1  } [ { data-name-2  } ]]
     { literal-1    }        { cd-name-1    }   { cd-name-2    }
                             { identifier-2 }   { identifier-3 }
                             { file-name-1  }   { file-name-2  }
  [;ON OVERFLOW imperative-statement]

CANCEL { identifier-1 } [ ,identifier-2 ] . . .
       { literal-1    } [ ,literal-2    ]

(1) CLOSE file-name-1 [ { REEL } ]
                       { UNIT }
                      [WITH NO REWIND]
                      [FOR REMOVAL   ]
                      WITH { NO REWIND }
                           { LOCK      }
```

(1) This format is for sequential and SAM files.*

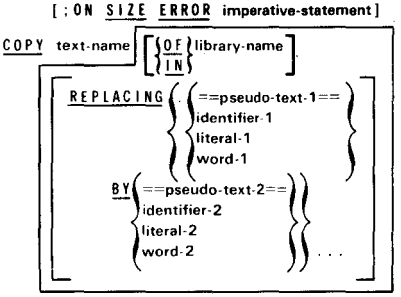
**Applies only to 90/25, 90/30, 90/30B, and 90/40 systems.*

PROCEDURE DIVISION - Basic Formats (cont)



① CLOSE file-name-1 [WITH LOCK] [,file-name-2 [WITH LOCK]]...

COMPUTE identifier-1 [ROUNDED]
[,identifier-2 [ROUNDED]]...
= arithmetic-expression
[;ON SIZE ERROR imperative-statement]



DELETE file-name RECORD
[;INVALID KEY imperative-statement]

DISABLE {INPUT} [TERMINAL] cd-name WITH KEY
{OUTPUT}
{identifier-1}
{literal-1}

DISPLAY {identifier-1} [,identifier-2]...
{literal-1} [,literal-2]
[UPON mnemonic-name]

DISPLAY {identifier-1} [,identifier-2]...
{literal-1} [,literal-2]
UPON mnemonic-name
[USING {identifier-3}]
{literal-3}
[ON EXCEPTION imperative-statement]

DISPLAY {identifier-1} [,identifier-2]...
{literal-1} [,literal-2]
UPON mnemonic-name
[ON EXCEPTION imperative-statement]

DIVIDE {identifier-1} INTO identifier-2 [ROUNDED]
{literal-1}
[,identifier-3 [ROUNDED]]...[;ON SIZE ERROR
imperative-statement]

① *This format is for relative, indexed, and ISAM* files.*

**Applies only to 90/25, 90/30, 90/30B, and 90/40 systems.*

PROCEDURE DIVISION - Basic Formats (cont)

```
DIVIDE {identifier-1} INTO {identifier-2} GIVING
       {literal-1   }      {literal-2   }
   identifier-3 [ROUNDED]

  [ [,identifier-4 [ROUNDED]]... ] [;ON SIZE ERROR
      imperative-statement]

DIVIDE {identifier-1} BY {identifier-2} GIVING identifier-3
       {literal-1   }    {literal-2   }
   [ROUNDED]

  [ [,identifier-4 [ROUNDED]]... ] [;ON SIZE ERROR
      imperative-statement]
```

```
DIVIDE {identifier-1} INTO {identifier-2} GIVING identifier-3
       {literal-1   }      {literal-2   }
   [ROUNDED]

   REMAINDER identifier-4 [;ON SIZE ERROR
      imperative-statement]

DIVIDE {identifier-1} BY {identifier-2} GIVING
       {literal-1   }    {literal-2   }
   identifier-3 [ROUNDED]

   REMAINDER identifier-4 [;ON SIZE ERROR
      imperative-statement]
```

```
ENABLE {INPUT  [TERMINAL]} cd-name WITH KEY
       {OUTPUT           }

   {identifier-1}
   {literal-1   }

EXHIBIT {NAMED        } {identifier          } ...
        {CHANGED NAMED} {nonnumeric-literal  }
        {CHANGED      }

EXIT [PROGRAM]

GO TO [ procedure-name-1 ]

GO TO procedure-name-1 [,procedure-name-2]...,
   procedure-name-n DEPENDING ON identifier

GO TO MORE-LABELS

IF condition [THEN] {statement-1  }
             [ ;  ] {NEXT SENTENCE}
   {;ELSE statement-2  }
   {;ELSE NEXT SENTENCE}

INSPECT identifier-1 TALLYING

  { ,identifier-2 FOR                                                      }
  {   { , { {ALL    } {identifier-3} }                               }     }
  {   {   { {LEADING} {literal-1   } }                               }     }
  {   {   { CHARACTERS               }                               }     }
  {   {   [ {BEFORE} INITIAL {identifier-4} ]                        } ... } ...
  {   {   [ {AFTER }         {literal-2   } ]                        }     }

INSPECT identifier-1 REPLACING

  { CHARACTERS BY                                                                      }
  {    {identifier-6} [ {BEFORE} INITIAL {identifier-7} ]                              }
  {    {literal-4   } [ {AFTER }         {literal-5   } ]                              }
  {                                                                                    }
  { { , {ALL    } { , {identifier-5} BY {identifier-6}                      }     }     }
  { {   {LEADING} {   {literal-3   }    {literal-4   }                      }     }     }
  { {   {FIRST  } {   [ {BEFORE} INITIAL {identifier-7} ]                   } ... } ... }
  { {           } {   [ {AFTER }         {literal-5   } ]                   }     }     }
```

PROCEDURE DIVISION -
Basic Formats (cont)

```
INSPECT identifier-1 TALLYING
    ( . identifier-2 FOR
        ( . { ALL       } { identifier-3 }
            { LEADING   } { literal-1   }
            { CHARACTERS }
          [ { BEFORE } INITIAL { identifier-4 } ] ) ...
          [ { AFTER  }         { literal-2   } ]
    ) ...

REPLACING
    { CHARACTERS BY { identifier-6 } [ { BEFORE } INITIAL { identifier-7 } ]
                    { literal-4    } [ { AFTER  }         { literal-5   } ]
    ( . { ALL       } ( . { identifier-5 } BY { identifier-6 }
        { LEADING   }     { literal-3   }     { literal-4   }
        { FIRST     }   [ { BEFORE } INITIAL { identifier-7 } ] ) ...
                        [ { AFTER  }         { literal-5   } ]
    ) ...
    }
```

```
MERGE file-name-1 ON { ASCENDING  } KEY data-name-1
                     { DESCENDING }
     [ , data-name-2 ] ...
     ON { ASCENDING  } KEY data-name-3
        { DESCENDING }
          [ , data-name-4 ] ...
     [ COLLATING SEQUENCE IS alphabet-name ]
     USING file-name-2, file-name-3 [ , file-name-4 ] ...
     { OUTPUT PROCEDURE IS section-name-1
                 [ { THROUGH } section-name-2 ]
                 [ { THRU    }                ]
     { GIVING file-name-5 }
```

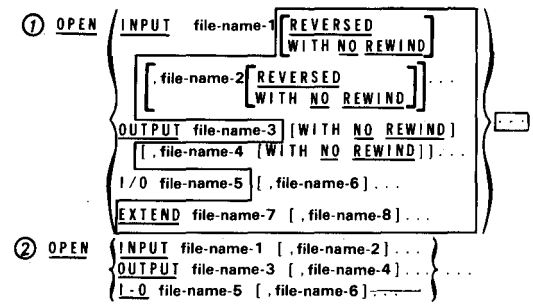
```
MOVE { identifier-1 } TO identifier-2 [ . identifier-3 ] ...
     { literal      }
```

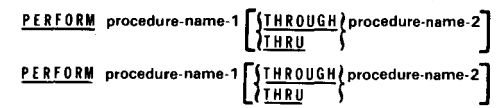
```
MOVE { CORRESPONDING } identifier-1 TO identifier-2
     { CORR          }
```

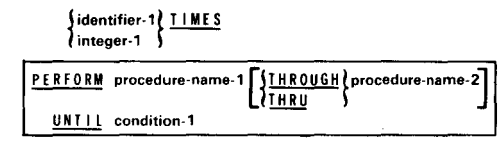
```
MULTIPLY { identifier-1 } BY identifier-2 [ ROUNDED ]
         { literal-1    }
    [ . identifier-3 [ ROUNDED ] ] ...
      [ ; ON SIZE ERROR imperative-statement ]

MULTIPLY { identifier-1 } BY { identifier-2 }
         { literal-1    }    { literal-2   }
    GIVING identifier-3 [ ROUNDED ]
    [ . identifier-4 [ ROUNDED ] ] ...
      [ ; ON SIZE ERROR imperative-statement ]
```

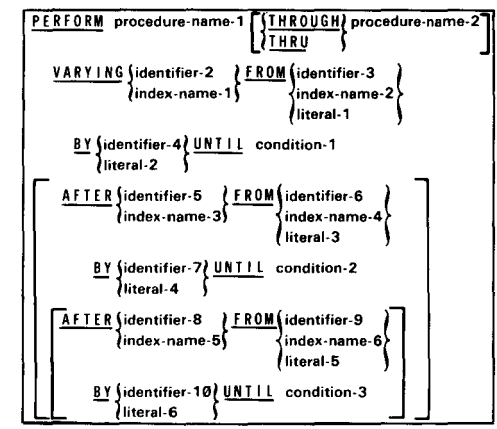
```
ON integer-1 [ AND EVERY integer-2 ] [ UNTIL integer-3 ]
   { statement-1 } [ ELSE { statement-2 } ]
   { NEXT SENTENCE }      { NEXT SENTENCE }
```

PROCEDURE DIVISION -
Basic Formats (cont)









① READ file-name RECORD [INTO identifier] [;AT END imperative-statement]

② READ file-name [NEXT] RECORD [INTO identifier]
[;AT END imperative-statement]

③ READ file-name RECORD [INTO identifier]
[;INVALID KEY imperative-statement]

④ READ file-name RECORD INTO identifier [;KEY IS data-name]
[;INVALID KEY imperative-statement]

---

① This format is for sequential and [SAM] files.*

② This format is for relative, indexed, and [ISAM] files.*

③ This format is for relative and [ISAM] files.*

④ This format is for indexed files only.

**Applies only to 90/25, 90/30, 90/30B, and 90/40 systems.*

PROCEDURE DIVISION -
Basic Formats (cont)

```
RECEIVE cd-name { MESSAGE } INTO identifier-1
                { SEGMENT }
  [ : NO DATA imperative-statement ]
RELEASE record-name [ FROM identifier ]
RETURN file-name RECORD [ INTO identifier ]
  : AT END imperative-statement
```

(1) `REWRITE record-name [ FROM identifier ]`

(2)
```
REWRITE record-name [ FROM identifier ]
  [ : INVALID KEY imperative-statement ]
```

```
SEARCH identifier-1 [ VARYING { identifier-2 } ]
                              { index-name-1 }
  [ : AT END imperative-statement-1 ]
  : WHEN condition-1 { imperative-statement-2 }
                     { NEXT SENTENCE          }
  [ : WHEN condition-2 { imperative-statement-3 } ... ]
                       { NEXT SENTENCE          }
SEARCH ALL identifier-1 [ : AT END
      imperative-statement-1 ]
  : WHEN { data-name-1 { IS EQUAL TO } { identifier-3      } }
         {             { IS =       } { literal-1         } }
         {                            { arithmetic-expression-1 } ...
         { condition-name-1                                      }
    [ AND { data-name-2 { IS EQUAL TO } { identifier-4      } } ]
          {             { IS =       } { literal-2         } }
          {                            { arithmetic-expression-2 } ...
          { condition-name-2                                      }
    { imperative-statement-2 }
    { NEXT SENTENCE          }
```

```
SEND cd-name FROM identifier-1
SEND cd-name [ FROM identifier-1 ] { WITH identifier-2 }
                                   { WITH EST          }
                                   { WITH EMI          }
                                   { WITH EGI          }
  [ { BEFORE } ADVANCING { { identifier-3 } [ LINE  ] } ]
    { AFTER  }           { { integer      } [ LINES ] }
                         { PAGE                       }
SET { identifier-1 [ . identifier-2 ] ... } TO { identifier-3 }
    { index-name-1 [ . index-name-2 ] ... }    { index-name-3 }
                                               { integer-1    }
SET index-name-4 [ . index-name-5 ] ... { UP BY   }
                                        { DOWN BY }
    { identifier-4 }
    { integer-2    }
```

---

(1) *This format is for sequential and SAM* files.*

(2) *This format is for relative, indexed, and ISAM* files.*

**Applies only to 90/25, 90/30, 90/30B, and 90/40 systems.*

PROCEDURE DIVISION -
Basic Formats (cont)

```
SORT file-name-1 ON {ASCENDING } KEY data-name-1
                    {DESCENDING}
        [ data-name-2 ] . . .
  [ ON {ASCENDING } KEY data-name-3
       {DESCENDING}
        [ data-name-4 ] . . . ] . . .
  [ COLLATING SEQUENCE IS alphabet-name ]
  { INPUT PROCEDURE IS section-name-1 [ {THROUGH} section-name-2 ] }
  {                                    {THRU    }                 }
  { USING file-name-2 [ filename-3 ] . . .                        }
  { OUTPUT PROCEDURE IS section-name-3 [ {THROUGH} section-name-4 ] }
  {                                    {THRU    }                 }
  { GIVING file-name-4                                            }
```

(1)

```
START file-name [ KEY { IS EQUAL TO          } data-name ]
                      { IS =                 }
                      { IS GREATER THAN      }
                      { IS >                 }
                      { IS NOT LESS THAN     }
                      { IS NOT <             }
  [ : INVALID KEY imperative-statement ]
```

(2)

```
START file-name [ KEY { IS EQUAL TO          } data-name ]
                      { IS =                 }
                      { IS NOT LESS THAN     }
                      { IS NOT <             }
  [ : INVALID KEY imperative-statement ]
```

```
STOP {RUN    }
     {literal}
```

```
STRING {identifier-1} [ identifier-2 ] . . . DELIMITED BY
       {literal-1   } [ literal-2    ]
  {identifier-3}
  {literal-3   }
  {SIZE        }
  [ {identifier-4} [ identifier-5 ]
  { {literal-4   } [ literal-5    ] . . . DELIMITED BY
  [ {identifier-6}                                    ] ]
  { {literal-6   }                                    } . . .
  { {SIZE        }                                    }
  INTO identifier-7 [ WITH POINTER identifier-8 ]
  [ : ON OVERFLOW imperative-statement ]
```

```
SUBTRACT {identifier-1} [ identifier-2 ] . . . FROM
         {literal-1   } [ literal-2    ]
  identifier-m [ ROUNDED ]
  [ identifier-n [ ROUNDED ] ] . . .
  [ : ON SIZE ERROR imperative-statement ]
```

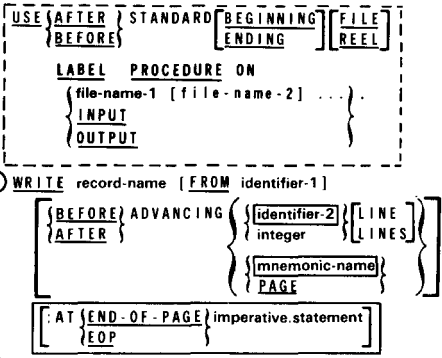
(1) This format is for relative and indexed files.

(2) This format is for relative, indexed, and ISAM files and applies only to 90/25, 90/30, 90/30B, and 90/40 systems.

PROCEDURE DIVISION - Basic Formats (cont)

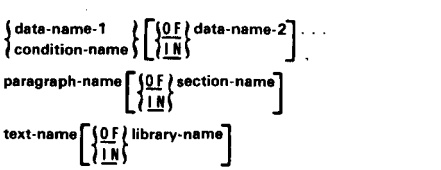
```
SUBTRACT {identifier-1} [,identifier-2]... FROM
         {literal-1   } [,literal-2   ]

  {identifier-m}
  {literal-m   }

  GIVING identifier-n [ROUNDED]
  [,identifier-o [ROUNDED]]...

  [;ON SIZE ERROR imperative-statement]

SUBTRACT {CORRESPONDING} identifier-1 FROM
         {CORR         }
  identifier-2 [ROUNDED]

  [;ON SIZE ERROR imperative-statement]

{READY} TRACE
{RESET}

TRANSFORM identifier-1 [,identifier-2]...CHARACTERS
  FROM {identifier-3         } TO {identifier-4         }
       {nonnumeric-literal-1 }    {nonnumeric-literal-2 }
       {figurative-constant-1}    {figurative-constant-2}
TRANSFORM identifier-1 [,identifier-2]...CHARACTERS
  {ON}identifier-5
  {BY}

UNSTRING identifier-1
  [DELIMITED BY [ALL] {identifier-2} ]
                      {literal-1   }
  [[,OR [ALL] {identifier-3} ]...]
              {literal-2   }
  INTO identifier-4 [,DELIMITER IN identifier-5]
  [,COUNT IN identifier-6]
  [,identifier-7 [,DELIMITER IN identifier-8]
    [,COUNT IN identifier-9]]...
  [WITH POINTER identifier-10]
  [TALLYING IN identifier-11]
  [;ON OVERFLOW imperative-statement]

USE AFTER STANDARD {EXCEPTION}
                   {ERROR    }

PROCEDURE ON { file-name-1 [,file-name-2]... }
             { INPUT                         }
             { OUTPUT                        }
             { I-O                           }
             { EXTEND                        }

USE FOR DEBUGGING ON
  { cd-name-1                                  }
  { [ALL REFERENCES OF] identifier-1           }
  { file-name-1                                }
  { procedure-name-1                           }
  { ALL PROCEDURES                             }

  [ cd-name-2                                  ]
  [ [ALL REFERENCES OF] identifier-2           ] ...
  [ file-name-2                                ]
  [ procedure-name-2                           ]
  [ ALL PROCEDURES                             ]
```

PROCEDURE DIVISION -
Basic Formats (cont)



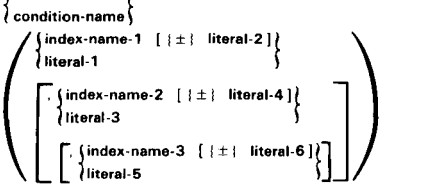
MISCELLANEOUS FORMATS

QUALIFICATION:



SUBSCRIPTING:

INDEXING



① This format is for sequential and SAM files.
② This format is for relative, indexed, and ISAM files.

MISCELLANEOUS FORMATS (cont)

IDENTIFIER: FORMAT 1

```
data-name-1 [ { OF } data-name-2 ] ...
            [ { IN }             ]

    [ ( subscript-1 [ , subscript-2 [ , subscript-3 ] ] ) ]
```

IDENTIFIER: FORMAT 2

```
data-name-1 [ { OF } data-name-2 ] ...
            [ { IN }             ]

  [ ( { index-name-1 [ { + } literal-2 ] }
      { literal-1    [ { - }           ] }

    [ , { index-name-2 [ { + } literal-4 ] }
        { literal-3    [ { - }           ] }

      [ , { index-name-3 [ { + } literal-6 ] } ] ) ]
          { literal-5    [ { - }           ] }
```

RELATION CONDITION:

```
{ identifier-1 }   { IS [NOT] GREATER THAN }
{ literal-1    }   { IS [NOT] LESS THAN    }
{ arithmetic-expression-1 }   { IS [NOT] EQUAL TO }
                              { IS [NOT] >        }
                              { IS [NOT] <        }
                              { IS [NOT] =        }

{ identifier-2 }
{ literal-2    }
{ arithmetic-expression-2 }
```

CLASS CONDITION:

```
identifier IS [NOT] { ALPHABETIC }
                    { NUMERIC    }

(1) identifier IS [NOT] CLASS-NAME mnemonic-name
```

SIGN CONDITION:

```
arithmetic-expression is [NOT] { POSITIVE }
                               { NEGATIVE }
                               { ZERO     }
```

CONDITION-NAME CONDITION:

```
condition-name
```

SWITCH-STATUS CONDITION:

```
condition-name
```

(1) This clause used when the non-English language feature is invoked.

MISCELLANEOUS FORMATS (cont)

NEGATED SIMPLE CONDITION:

NOT simple-condition

COMBINED CONDITION:

condition {{AND / OR} condition} ...

ABBREVIATED COMBINED RELATION CONDITION:

relation-condition {{AND / OR} [NOT] [relational-operator] object} ...

CONDITIONAL EXPRESSIONS SEQUENCE OF ELEMENTS

Element	End Position		Intermediate Position (left-to-right)	
	First	Last	May Be Immediately Preceded by Only:	May Be Immediately Followed by Only:
C	Yes	Yes	OR, NOT, AND, (	OR, AND,)
OR or AND	No	No	C,)	C, NOT, (
NOT	Yes	No	OR, AND, (	C, (
(	Yes	No	OR, NOT, AND, (	C, NOT, (
)	No	Yes	C,)	OR, AND,)

C = Simple-condition

RESERVED WORDS

ACCEPT
ACCESS
ADD
ADVANCING
AFTER
ALL
ALPHABET
ALPHABETIC
ALSO
ALTER
ALTERNATE
AND
APPLY
ARE
AREA
AREAS
ASCENDING
ASSIGN
AT
AUTHOR
BEFORE
BEGINNING
BLANK
BLOCK
BLOCK-COUNT

RESERVED WORDS (cont)

BOTTOM
BY

CALL
CANCEL
CD
CF
CH
CHANGED
CHARACTER
CHARACTERS
CLASS-NAME
CLOCK-UNITS
CLOSE
COBOL
CODE
CODE-SET
COLLATING
COLUMN
COMMA
COMMUNICATION
COMP
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
COMPUTE
COMP-1
COMP-2
COMP-3
COMP-4
CONFIGURATION
CONNECT-FREE
CONTAINS
CONTROL
CONTROLS
COPY
CORR
CORRESPONDING
COUNT
CURRENCY
CYLINDER-INDEX
CYLINDER-OVERFLOW

DATA
DATE
DATE-COMPILED
DATE-WRITTEN
DAY
DE
DEBUG-CONTENTS
DEBUG-ITEM
DEBUG-LINE
DEBUG-NAME
DEBUG-SUB-1
DEBUG-SUB-2
DEBUG-SUB-3
DEBUGGING
DECIMAL-POINT
DECLARATIVES
DELETE
DELIMITED
DELIMITER
DEPENDING
DESCENDING
DESTINATION
DETAIL
DISABLE
DISPLAY
DIVIDE
DIVISION
DOWN
DUPLICATES
DYNAMIC

EGI
ELSE
EMI
ENABLE
END
ENDING
END-OF-PAGE
ENVIRONMENT
EOP
EQUAL
ERROR
ESI
EVERY
EXCEPTION
EXHIBIT
EXIT
EXTEND

FD
FILE
FILE-CONTROL
FILE-ID
FILLER
FINAL
FIRST
FOOTING
FOR
FROM
FUNCTION-KEYS

RESERVED WORDS (cont)

GENERATE
GIVING
GO
GREATER
GROUP

HEADING
HIGH-VALUE
HIGH-VALUES

I-O
I-O-CONTROL
IDENTIFICATION
IF
IN
INDEX
INDEX-AREA
INDEXED
INDICATE
INDICES
INITIAL
INITIATE
INPUT
INPUT-OUTPUT
INSPECT
INSTALLATION
INTO
INVALID
IS
ISAM

JUST
JUSTIFIED

KEY

LABEL
LAST
LEADING
LEFT
LENGTH
LESS
LIMIT
LIMITS
LINAGE
LINAGE-COUNTER
LINE
LINE-COUNTER
LINES
LINKAGE
LOCK
LOW-VALUE
LOW-VALUES

MEMORY
MERGE
MESSAGE
MESSAGES
MODE
MODULES
MORE-LABELS
MOVE
MULTIPLE
MULTIPLY

NAMED
NATIVE
NEGATIVE
NEXT
NO
NOT
NUMBER
NUMERIC

OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OR
ORGANIZATION
OUTPUT
OVERFLOW

PAGE
PAGE-COUNTER
PASSWORD
PERCENT
PERFORM
PF
PH
PIC
PICTURE
PLUS
POINTER
POSITION
POSITIVE
PRINTING
PROCEDURE
PROCEDURES
PROCEED
PROGRAM

RESERVED WORDS (cont)

PROGRAM-ID
PUBLIC

QUEUE
QUOTE
QUOTES

RANDOM
RD
READ
READY
RECEIVE
RECORD
RECORDS
REDEFINES
REEL
REFERENCES
RELATIVE
RELEASE
REMAINDER
REMOVAL
RENAMES
REPLACING
REPORT
REPORTING
REPORTS
RERUN
RESERVE
RESET
RETURN
REVERSED
REWIND
REWRITE
RF
RH
RIGHT
ROUNDED
RUN

SAM
SAME
SD
SEARCH
SECTION
SECURITY
SEGMENT
SEGMENT-LIMIT
SELECT
SEND
SENTENCE
SEPARATE
SEQUENCE
SEQUENTIAL
SET
SIGN
SIZE
SORT
SORT-FILE-SIZE
SORT-MERGE
SORT-MODE-SIZE
SOURCE
SOURCE-ALPHABET
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
SPECIFIC
SPERRY-OS3
SPERRY-VS9
SPERRY-VS9-MODEL 80
STANDARD
STANDARD-0
STANDARD-1
START
STATUS
STOP
STRING
SUB-QUEUE-1
SUB-QUEUE-2
SUB-QUEUE-3
SUBTRACT
SUM
SUPPRESS
SYMBOLIC
SYNC
SYNCHRONIZED
SYSCHAN-n
(n=1 thru 15)
SYSCOM
SYSCONSOLE
SYSFORMAT
SYSIN
SYSIPT
SYSLOG
SYSLST
SYSOPT
SYSOUT
SYSSCOPE
SYSSWCH
SYSWCH-n
(n=0 thru 31)
SYSTEM
SYSTEM-SHUTDOWN
SYSTERMINAL
SYSWORK

RESERVED WORDS (cont)

TABLE
TALLYING
TAPE
TAPES
TERMINAL
TERMINATE
TEXT
THAN
THEN
THROUGH
THRU
TIME
TIMES
TO
TOP
TRACE
TRAILING
TRANSFORM
TYPE

UNIT
UNIVAC-0S3
UNIVAC-VS9
UNIVAC-VS9-MODEL-80
UNSTRING
UNTIL
UP
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING
VERIFY

WHEN
WHEN-COMPILED
WITH
WORDS
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

*DEBUG
+
-
*
/
**
>
<
=
==

PARAM STATEMENT OPTIONS

PARAM Card/Result
// PARAM COBL74,AXNON={YES / NO} Suppresses nonreferenced entries in alphabetically ordered cross-reference listing
// PARAM AXREF={YES / NO} Specifies an alphabetically ordered cross-reference listing

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result

```
// PARAM CALLST={YES}
                {NO }
```

Specifies static CALL of subprograms referenced by the literal option. YES indicates that subprograms named by the literal option of the CALL statements are to be linked with the main program. NO indicates that subprograms named either by the literal or identifier option of the CALL statements are to be dynamically loaded when called.

```
// PARAM CDMIO={YES}
               {NO }
               {MI }
```

YES specifies that consolidated data management will be used for all files except ISAM and SAM disk files. NO specifies that DTF interfaces will be used for all files except workstation files. MI specifies that CDM interfaces will be used for MIRAM and workstation files only; DTF interfaces will be used for all other files. This parameter is applicable only for a Series 90 system that supports both CDM and DTF interfaces.

```
// PARAM CMCS=name
```

Specifies a 1- to 8-character module name of the COBOL communication control system. If this parameter is not specified for a COBOL communication program, a default name, consisting of 6 characters of the PROGRAM-ID name (left-justified and zero-filled, if necessary) and a suffix of 2 characters (CM) is used.

```
// PARAM CMCSST={YES}
                {NO }
```

YES indicates that the CMCS module is bound with the COBOL object program. NO indicates that the CMCS module will be dynamically loaded at execution time.

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result
`// PARAM CPYTXT={YES / NO}` Includes COBOL library text in source listing
`// PARAM DIAG={YES / NO}` Specifies a diagnostic listing
`// PARAM DIAGWN={YES / NO}` Includes warning diagnostics in the diagnostic listing
`// PARAM ERRFIL=module-name/lfdname` Specifies generation of an error-file element of compile-time diagnostics. The module-name is the 1- to 8-character module name of the element. The lfdname is the 1- to 8-character name of the MIRAM library where the element will be generated. The ERRFIL parameter is ignored unless the IN parameter is also specified. The error-file element is used by the OS/3 editor error file processing facility (@EFP command).

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result
`// PARAM FIPS=(1 | 2 | 3 | 4 | 5)` Specifies a FIPS PUB 21-1 flagging option.
`// PARAM IMSCOD={YES | NO}` Specifies IMS compatible; i.e., COBOL programs are to be executed under control of IMS as action programs. When IMSCOD=YES is specified, the COBOL language elements restricted by IMS are flagged and deleted.
`// PARAM IN=m-n/f-n` The m-n is a 1- to 8-character source module name in the library; f-n is a 1- to 8-character LFD name identifying the file on which the source module resides. If f-n is omitted, the default name **SYSSRC** is used.
`// PARAM LIN=filename/filename/` Filename is a 1- to 8-character LFD name identifying the file or files where the COPY library resides. A maximum of 10 LFD names can be specified, allowing multiple COPY libraries to be searched. If the library-name is specified in the COPY statement, the library-name takes precedence. If the library-name is omitted in the COPY statement, the filename or filenames in the LIN parameter are used. Multiple file names are searched sequentially in the order specified on the LIN parameter. if the parameter is omitted, the name COPY$ is used as the default name of the LIN parameter.

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result
// PARAM LIST={YES / NO} Specifies a source program listing.
// PARAM LNKCON={YES / NO} Specifies generation or suppression of linker control statements in the object module
// PARAM LSTREF={YES / NO} Specifies a source listing with definition references
// PARAM LSTWTH=nnn Specifies the page width; nnn ranges from 120 through 160. Default value is 120 characters a line.
// PARAM MAP={YES / NO} Specifies an object program locator/map listing
// PARAM MXNON={YES / NO} Suppresses nonreferenced entries in the map listing with cross-references
// PARAM MXREF={YES / NO} Specifies a map listing with cross-references

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result
`// PARAM OBJ=filename` Filename is a 1- to 8-character LFD name of the file on which the generated object module is to be stored. If the parameter is not specified, the default name YRUN is used.

`// PARAM OBJLST={YES}`
`{NO }`
 	
Specifies an object program listing

`// PARAM OBJMOD={YES}`
`{NO }`
 	
Specifies object module production

`// PARAM PAGOVF={YES}`
`{NO }`
 	
YES provides automatic printer page eject feature in the object program. NO indicates omission of the eject feature in the object program. PAGOVF=YES should not be specified, if the LINAGE clause or the ADVANCING PAGE phrase is specified in the source program.

`// PARAM PROVER={YES}`
`{NO }`
 	
YES specifies the production of a listing of procedure-names and verbs with associated source line numbers and object program relative addresses. NO indicates suppression of the listing.

`// PARAM SPRLST={YES}`
`{NO }`
 	
Suppresses all listings unconditionally. This parameter overrides all other listing parameters.

PARAM STATEMENT OPTIONS (cont)

PARAM Card/Result
// PARAM SPROUT={1 / 2 / 3} Suppresses compiler output (except source listing, diagnostic listing, and related options) when severity code level 1, 2, or 3 errors are encountered.
// PARAM SYNCHK={YES / NO} Specifies syntax check only on normal compilation. When SYNCHK=YES is specified, only the FIPS, LSTDTH, and LSTWTH parameters may be specified. A source program listing and a diagnostic listing are produced automatically by the compiler.
// PARAM TRNADR={YES / NO} YES indicates generation of a transfer address in the object module. NO indicates suppression of a transfer address, in which case, the program cannot be executed unless it is called.
// PARAM TRUNC={YES / NO} YES indicates that data truncation and detection of SIZE ERROR on binary items are based on the decimal digits specified in the PICTURE character-string. NO indicates that data truncation and detection are based on the actual storage size allocated to the items.

PARAM STATEMENT CONSISTENCY CHECKS

User Specifications		Compiler Actions	
Parameter	Value	Parameter	Value
LIST	NO	LSTREF	NO
LIST	NO	CPYTXT	NO
MXNON	YES	MXREF	YES
MXREF	YES	MAP	YES
AXNON	YES	AXREF	YES
IMSCOD	YES	CALLST	YES
OBJMOD	NO	PROVER	NO
SYNCHK	YES	LIST	YES
SYNCHK	YES	DIAG	YES
SYNCHK	YES	OBJLST	NO
SYNCHK	YES	OBJMOD	NO
SYNCHK	YES	MAP	NO
SYNCHK	YES	MXREF	NO
SYNCHK	YES	AXREF	NO
SYNCHK	YES	PROVER	NO
SYNCHK	YES	LNKCOM	NO
SYNCHK	YES	PAGOVF	NO
SYNCHK	YES	TRNADR	NO
SPRLST	YES	LIST	NO
SPRLST	YES	LSTREF	NO
SPRLST	YES	CPYTXT	NO
SPRLST	YES	OBJLST	NO
SPRLST	YES	MAP	NO
SPRLST	YES	MXREF	NO
SPRLST	YES	AXREF	NO
SPRLST	YES	DIAG	NO

JPROC TO EXECUTE COBL74 LANGUAGE PROCESSOR