Burroughs

B 6700 / B 7700

EXECUTIVE SYSTEM PROGRAMMING LANGUAGE (ESPOL)

INFORMATION MANUAL

THIS MANUAL REPLACES FORM NO. 5000094 DATED NOVEMBER 1970

Printed in U.S. America

27 June 1972

5000094

LIST OF EFFECTIVE PAGES

NOTE: Insert latest changed page; dispose of superseded pages.

TOTAL NUMBER OF PAGES IN THIS MANUAL IS 248 CONSISTING OF THE FOLLOWING:

Page No.	Issue
Title	Original
A	Original
Table of Contents	Original
Preface	Original
1-1 and 1-2	Original
2-1	Original
2-2 Blank	Original
3-1 and 3-2	Original
4-1 thru 4-10	Original
5-1 thru 5-18	Original
6-1 thru 6-3	Original
6-4 Blank	Original
7-1 thru 7-43	Original
7-44 Blank	Original
8-1 thru 8-44	Original
9-1 thru 9-26	Original
A-1 thru A-13	Original
A-14 Blank	Original
B-1 thru B-9	Original
B-10 Blank	Original
C-1 thru C-37	Original
C-38 Blank	Original
D-1 thru D-27	Original
D-28 Blank	Original

AA146565, AA235163

B6700/B7700 ESPOL

TABLE OF CONTENTS

PREFACE

THIS DOCUMENT, WHICH DESCRIBES THE FORMAL STRUCTURE OF THE B6700/B7700 EXECUTIVE SYSTEM PROGRAMMING LANGUAGE (ESPOL), IS INTENDED FOR THOSE ALREADY EXPERIENCED WITH ALGOL AND THE SPECIFICATIONS OF THE B6700/B7700 INFORMATION PROCESSING SYSTEM.

THIS DOCUMENT FIRST PRESENTS A REVIEW OF THE "BACKUS-NAUR FORM." NEXT, THE CHARACTER SET, BASIC SYMBOLS, AND THE BASIC AND GENERAL COMPONENTS OF THE LANGUAGE ARE DISCUSSED. BEGINNING WITH SECTION 6, THE SYNTAX OF ESPOL IS EXPLAINED AS FOLLOWS: FIRST, AN ESPOL PROGRAM IS DEFINED, FOLLOWED BY DECLARATIONS, STATEMENTS, AND FINALLY EXPRESSIONS. IN THIS WAY, THE READER MAY PROGRESS MORE EASILY IN HIS UNDERSTANDING OF ESPOL. FURTHERMORE, AS AN ADDITIONAL AID, THE MATERIAL IN MANY CASES, HAS BEEN ARRANGED IN ALPHABETICAL SEQUENCE.

FOUR APPENDICES ARE INCLUDED IN THIS DOCUMENT. THESE APPENDICES HAVE BEEN DESIGNED TO FURTHER FACILITATE THE READER'S UNDERSTANDING OF THE STRUCTURE AND USE OF THE ESPOL LANGUAGE.

APPENDIX A IS AN ORDERED LIST OF COMPILER ERROR MESSAGES AND ASSOCIATED ESPOL PROCEDURES.

APPENDIX B CONTAINS EXAMPLES OF MACHINE-LANGUAGE CODE GENERATED FOR VARIOUS COMMON CONSTRUCTS.

APPENDIX C IS A LOGICAL-ORDER LISTING OF THE COMPLETE ESPOL LANGUAGE SYNTAX TREE.

APPENDIX D IS AN ALPHABETICAL INDEX OF BACKUS-NAUR CONSTRUCTS AND RESERVED WORDS. THE BACKUS-NAUR CONSTRUCTS ARE PRECEDED BY A SYNTAX TREE (APPENDIX C) REFERENCE NUMBER.

1. INTRODUCTION

1.1. FUNCTION OF ESPOL

THE B6700/B7700 EXECUTIVE SYSTEM PROGRAMMING ORIENTED LANGUAGE (ESPOL) IS USED PRIMARILY IN WRITING THE B6700/B7700 MASTER CONTROL PROGRAM (MCP).

1.2. ESPOL COMPARED WITH ALGOL

WHILE IN MANY WAYS ESPOL IS SIMILAR TO ALGOL, ESPOL IS DESIGNED TO FACILITATE SUCH FUNCTIONS AS INTERRUPT HANDLING, STORAGE ALLOCATION, AND OVERLAY FUNCTIONS PECULIAR TO THE MCP.

1.3. ESPOL AND THE B6700/B7700

BECAUSE B6700/B7700 ESPOL IS UNIQUELY SUITED TO THE B6700/B7700 INFORMATION PROCESSING SYSTEM, CERTAIN CONSTRUCTS ARE PROVIDED TO ALLOW MANIPULATION OF HARDWARE FUNCTIONS.

1.4. THE METALANGUAGE

THE SYNTAX OF B6700/B7700 ESPOL IS DESCRIBED BY THE USE OF THE BACKUS-NAUR FORM (BNF) METALANGUAGE, WHICH IS A SYSTEM OF METALINGUISTIC SYMBOLS AND METALINGUISTIC FORMULAS DISCUSSED BELOW.

1.4.1. METALINGUISTIC SYMBOLS

< > LEFT AND RIGHT BROKEN BRACKETS ARE USED TO CONTAIN ONE OR MORE CHARACTERS REPRESENTING A METALINGUISTIC VARIABLE WHOSE VALUE IS GIVEN BY A METALINGUISTIC FORMULA.

::= THE SYMBOL "::=" MEANS "IS DEFINED AS". IT SEPARATES THE

METALINGUISTIC VARIABLE ON THE LEFT OF A METALINGUISTIC FORMULA FROM THE DEFINITION OF THE METALINGUISTIC VARIABLE.

/ THE SYMBOL "/" MEANS "OR". IT SEPARATES MULTIPLE DEFINITIONS OF A METALINGUISTIC VARIABLE.

< > THE SYMBOLS "<" AND ">" ARE USED TO ENCLOSE METALINGUISTIC ENGLISH-LANGUAGE EXPRESSIONS. THESE EXPRESSIONS ARE USED ONLY WHEN IT IS IMPOSSIBLE OR IMPRACTICAL TO USE A METALINGUISTIC FORMULA.

1.4.2. METALINGUISTIC FORMULAS

A METALINGUISTIC FORMULA IS A RULE WHICH WILL PRODUCE A SYNTACTICALLY CORRECT SEQUENCE OF SYMBOLS. IN A METALINGUISTIC FORMULA, ANY MARK OR SYMBOL WHICH IS NOT ONE OF THE DEFINED METALINGUISTIC SYMBOLS DENOTES ITSELF. THE JUXTAPOSITION OF THE METALINGUISTIC SYMBOLS IN A METALINGUISTIC FORMULA DENOTES THE JUXTAPOSITION OF THOSE ELEMENTS IN THE CONSTRUCT SO DEFINED.

EXAMPLE:

THE METALINGUISTIC FORMULA:

```
<IDENTIFIER> ::= <LETTER> / <IDENTIFIER> <LETTER> /
    <IDENTIFIER> <DIGIT>
```

IS READ AS FOLLOWS: AN IDENTIFIER IS DEFINED AS A LETTER, OR AN IDENTIFIER FOLLOWED BY A LETTER, OR AN IDENTIFIER FOLLOWED BY A DIGIT.

2. CHARACTER SET

THE CHARACTER SET FOR WHICH ESPOL IS DEFINED IS A SUBSET OF THE BURROUGHS COMMON LANGUAGE (BCL) CHARACTER SET.

SYNTAX:

<CHARACTER> ::= <STRING CHARACTER> / <STRING BRACKET CHARACTER> /
    <INVALID CHARACTER>
<STRING CHARACTER> ::= <LETTER> / <DIGIT> / <SPECIAL CHARACTER> /
    <SINGLE SPACE>
<LETTER> ::= A / B / C / D / E / F / G / H / I / J / K /
    L / M / N / O / P / Q / R / S / T / U / V / W / X / Y / Z
<DIGIT> ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9
<SPECIAL CHARACTER> ::= . / , / [/] / (/) / + / - / <SLASH> /
    > / < / ∠ / ↘ / = / ≠ / ← / % / & / * / # / @ / : / ; / $
<SLASH> ::= /
<SINGLE SPACE> ::= ∠ONE HORIZONTAL BLANK POSITION↘
<STRING BRACKET CHARACTER> ::= <QUOTE>
<QUOTE> ::= "
<INVALID CHARACTER> ::= ?

SEMANTICS:

THE INVALID CHARACTER IS EXCLUDED EXCEPT AS A CHARACTER WHICH MAY BE PROGRAMMATICALLY PRODUCED.

3. BASIC SYMBOLS

IN GENERAL, SYMBOLS ARE MADE UP FROM A CONTIGUOUS SET OF CHARACTERS THE BASIC SYMBOLS OF ESPOL ARE THOSE SYMBOLS WHICH ARE WELL-DEFINED WITHIN THE CONTEXT OF THAT LANGUAGE.

SYNTAX:

```
<BASIC SYMBOL> ::= <EMPTY> / <CHARACTER> / <DELIMITER> /
      <LOGICAL VALUE>
<EMPTY> ::= ⌜THE NULL STRING OF CHARACTERS⌟
<DELIMITER> ::= <BRACKET> / <DECLARATOR> / <OPERATOR> /
      <SEPARATOR> / <SPECIFICATOR>
<BRACKET> ::= ( / ) / [ / ] / " / BEGIN / END / #
<DECLARATOR> ::= ARRAY / BOOLEAN / DEFINE / DOUBLE / EVENT /
      FIELD / FORWARD / INTEGER / LABEL / LAYOUT /
      MONITOR / ON / OWN / PICTURE / POINTER / PROCEDURE /
      QUEUE / REAL / REFERENCE / SAVE / USING / WORD
<OPERATOR> ::= <ARITHMETIC OPERATOR> / <CONCATENATE OPERATOR> /
      <LOGICAL OPERATOR> / <RELATIONAL OPERATOR> /
      <REPLACEMENT OPERATOR> / <SEQUENTIAL OPERATOR>
<ARITHMETIC OPERATOR> ::= + / - / * / <SLASH> / DIV / MOD / MUX
<CONCATENATE OPERATOR> ::= &
<LOGICAL OPERATOR> ::= AND / EQV / IMP / NOT / OR
<RELATIONAL OPERATOR> ::= > / ⌜ / = / ⌟ / < / ≠ / IS
<REPLACEMENT OPERATOR> ::= ← / := / ←* / :=*
<SEQUENTIAL OPERATOR> ::= CASE / DO / ELSE / FOR / GO / IF / THEN /
      TO
<SEPARATOR> ::= , / . / @ / : / ; / <SPACE> / BY / COMMENT / IN /
      NULL / OF / STEP / UNTIL / WHILE / WITH
<SPECIFICATOR> ::= VALUE
<LOGICAL VALUE> ::= TRUE / FALSE
```

SEMANTICS:

1. INDIVIDUAL LETTERS DO NOT HAVE INDIVIDUAL MEANINGS EXCEPT IN THE CONTEXT IN WHICH THEY ARE USED (AS DEFINED IN THE SYNTAX OF THE LANGUAGE).

2. A SPACE MUST SEPARATE ANY TWO OF THE FOLLOWING:

   MULTICHARACTER DELIMITER (EXCEPT ←*,:=* AND :=)
   LOGICAL VALUE
   IDENTIFIER
   UNSIGNED NUMBER

3. FOR CERTAIN CASES THE REPLACEMENT OPERATOR :=* MAY BE USED TO UPDATE VARIABLES; FOR EXAMPLE, THE STATEMENT "W:=W+1;" COULD ALSO BE WRITTEN "W:=*+1;" (OR OPTIONALLY, THE OPERATOR ←* COULD BE USED; E.G., "W←*+1;").

4. BASIC COMPONENTS

4.1. GENERAL

BASIC COMPONENTS ARE THE MOST PRIMITIVE CONSTRUCTS OF THE LANGUAGE.

SYNTAX:

<BASIC COMPONENT> ::= <BASIC SYMBOL> / <IDENTIFIER> /
<NUMBER> / <STRING>

4.2. IDENTIFIERS

SYNTAX:

<IDENTIFIER> ::= <LETTER> / <IDENTIFIER> <LETTER> /
<IDENTIFIER> <DIGIT>

SEMANTICS:

1. IDENTIFIERS HAVE NO ABSOLUTE MEANING. IDENTIFIERS ARE USED TO NAME LABELS, VARIABLES, ARRAYS, PROCEDURES, ETC.

2. A RESERVED WORD MAY NOT BE USED AS AN IDENTIFIER.

3. THE MAXIMUM PERMISSIBLE LENGTH OF AN IDENTIFIER IS 63 CHARACTERS, AND IT MUST CONTAIN NO SPECIAL CHARACTERS.

4. NO SPACE MAY APPEAR WITHIN AN IDENTIFIER.

EXAMPLES:

```
X
A5
G76D3
NOTHINGTODO
```

4.3. NUMBERS

SYNTAX:

```
<NUMBER> ::= <SIGN> <UNSIGNED NUMBER>
<SIGN> ::= <EMPTY> / + / -
<UNSIGNED NUMBER> ::= <DECIMAL NUMBER> <EXPONENT PART> /
      <DECIMAL NUMBER> / <OCTAL NUMBER>
<DECIMAL NUMBER> ::= <UNSIGNED INTEGER> <DECIMAL FRACTION> /
      <UNSIGNED INTEGER> / <DECIMAL FRACTION> / <UNSIGNED INTEGER>.
<UNSIGNED INTEGER> ::= <DIGIT> / <UNSIGNED INTEGER> <DIGIT>
<EXPONENT PART> ::= @ <INTEGER> / @@ <INTEGER>
<INTEGER> ::= <SIGN> <UNSIGNED INTEGER>
<DECIMAL FRACTION> ::= . <UNSIGNED INTEGER>
<OCTAL NUMBER> ::= @ <OCTAL CONSTANT>
<OCTAL CONSTANT> ::= <OCTAL DIGIT> / <OCTAL CONSTANT>
      <OCTAL DIGIT>
<OCTAL DIGIT> ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7
```

SEMANTICS:

1. NUMBERS MAY BE INTEGER OR REAL: INTEGERS ARE OF TYPE "INTEGER" ALL OTHER NUMBERS ARE OF TYPE "REAL" (EITHER EXPLICITLY OR IMPLICITLY -- BY DEFAULT).

2. AN INTEGER IS A SINGLE-PRECISION OPERAND WITH AN EXPONENT OF ZERO.

3. NUMBERS MAY BE NEGATIVE (-) OR POSITIVE (+).

4. THE SMALLEST AND LARGEST NUMBERS REPRESENTABLE IN INTEGER, REAL, AND DOUBLE PRECISION ARE:

   A. SMALLEST SINGLE PRECISION INTEGER:
      = 1.

= 3"0000000000000001"

B. LARGEST SINGLE PRECISION INTEGER:

= 549755813887.

= 3"0007777777777777"

= 8*13 - 1

C. SMALLEST SINGLE PRECISION REAL:

= 1.27447352891@-57

= 3"1770000000000001"

D. LARGEST SINGLE PRECISION REAL:

= 4.31359146674@68

= 3"0777777777777777"

= (8*13 - 1) x 8*63

E. SMALLEST DOUBLE PRECISION INTEGER:

= 1.

= DOUBLE(3"0150000000000000", 3"0000000000000001")

F. LARGEST DOUBLE PRECISION INTEGER:

= 3022314549036572936765 43.

= DOUBLE(3"0157777777777777", 3"0007777777777777")

= 8*26 - 1

G. SMALLEST DOUBLE PRECISION REAL:

= 1.0@@-29603

= DOUBLE(3"1770000000000000", 3"7770000000000001")

= 1.0 x 8*-32767

H. LARGEST DOUBLE PRECISION REAL:

= 1.9488283820502807912446 9@@29603

= DOUBLE(3"0777777777777777", 3"7777777777777777")

= (8*26 - 1) x 8*32767

5. THE EXPONENT PART IS A SCALE FACTOR EXPRESSED AS IN INTEGRAL

POWER OF 10.

6. NO SPACE MAY APPEAR WITHIN AN UNSIGNED NUMBER.

7. AN EXPONENT PART WITH A DOUBLE "AT" SIGN "@@" SIGNIFIES AN EXTENDED PRECISION VALUE.

8. AN OCTAL NUMBER MAY HAVE NO MORE THAN 16 OCTAL DIGITS.

EXAMPLES:

UNSIGNED NUMBERS

```
1354.543
@67
1354.54@68
27
1@-43
```

DECIMAL NUMBERS

```
 3.14
37
  .57
```

INTEGERS

```
+10
+546
-62256
23
```

UNSIGNED INTEGERS

```
5
69
8
```

EXPONENT PARTS

```
@68
@-46
@+54
```

DECIMAL FRACTIONS

```
.5
.69
```

4.4. STRINGS

SYNTAX:

```
<STRING> ::= <SIMPLE STRING> / <STRING> <SIMPLE STRING>
<SIMPLE STRING> ::= <NUMERIC STRING> / <ALPHA STRING>
<NUMERIC STRING> ::=
      <BINARY CODE> <QUOTE> <BINARY STRING> <QUOTE> /
      <QUATERNARY CODE> <QUOTE> <QUATERNARY STRING> <QUOTE> /
      <OCTAL CODE> <QUOTE> <OCTAL STRING> <QUOTE> /
      <HEXADECIMAL CODE> <QUOTE> <HEXADECIMAL STRING> <QUOTE>
<BINARY CODE> ::= 1 / 10 / 12 / 13 / 14 / 16 / 17 / 18 /
      120 / 130 / 140 / 160 / 170 / 180
<BINARY STRING> ::= <BINARY CHARACTER> / <BINARY STRING>
      <BINARY CHARACTER>
<BINARY CHARACTER> ::= 0 / 1
<QUATERNARY CODE> ::= 2 / 20 / 24 / 26 / 28 / 240 / 260 / 280
<QUATERNARY STRING> ::= <QUATERNARY CHARACTER> /
      <QUATERNARY STRING> <QUATERNARY CHARACTER>
<QUATERNARY CHARACTER> ::= 0 / 1 / 2 / 3
<OCTAL CODE> ::= 3 / 30 / 36 / 360
<OCTAL STRING> ::= <OCTAL CHARACTER> /
      <OCTAL STRING> <OCTAL CHARACTER>
<OCTAL CHARACTER> ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7
<HEXADECIMAL CODE> ::= 4 / 40
<HEXADECIMAL STRING> ::= <HEXADECIMAL CHARACTER> /
      <HEXADECIMAL STRING> <HEXADECIMAL CHARACTER>
<HEXADECIMAL CHARACTER> ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9 /
      A / B / C / D / E / F
<ALPHA STRING> ::= <BCL CODE> <QUOTE> <BCL STRING> <QUOTE> /
      <ASCII CODE> <QUOTE> <ASCII STRING> <QUOTE> /
      <EBCDIC CODE> <QUOTE> <EBCDIC STRING> <QUOTE>
<BCL CODE> ::= <EMPTY> / 6 / 60
<BCL STRING> ::= <QUOTE> / <BCL CHARACTER> /
      <BCL STRING> <BCL CHARACTER>
```

<BCL CHARACTER> ::= <STRING CHARACTER>
<ASCII CODE> ::= 7 / 70
<ASCII STRING> ::= <BCL STRING>
<EBCDIC CODE> ::= 8 / 80
<EBCDIC STRING> ::= <BCL STRING>

SEMANTICS:

1. STRINGS MAY CONSIST OF THE FOLLOWING:

ONE-BIT CHARACTERS	(BINARY)
TWO-BIT CHARACTERS	(QUATERNARY)
THREE-BIT CHARACTERS	(OCTAL)
FOUR-BIT CHARACTERS	(HEXADECIMAL)
SIX-BIT CHARACTERS	(BCL)
SEVEN-BIT CHARACTERS	(ASCII, IN EIGHT-BIT FORMAT)
EIGHT-BIT CHARACTERS	(EBCDIC)

2. THE STRING CODE DETERMINES THE INTERPRETATION OF THE CHARACTERS BETWEEN THE QUOTES. IT SPECIFIES THE CHARACTER SET AND, IF THE STRING HAS FEWER THAN 48 BITS, THE JUSTIFICATION OF THE STRING WITHIN THE WORD WHICH WILL CONTAIN IT. THE FIRST DIGIT SPECIFIES THE CHARACTER SET IN WHICH THE SOURCE STRING IS WRITTEN. THE NEXT NONZERO DIGIT SPECIFIES THE CHARACTER SIZE OF THE INTERNAL STRING CREATED BY THE COMPILER. IF NO SECOND SIZE IS SPECIFIED, IT WILL BE THE SAME AS THE INITIAL SIZE. A TRAILING ZERO INDICATES THAT THE STRING IS TO BE LEFT-JUSTIFIED WITHIN A WORD (IF THE STRING CONTAINS FEWER THAN 48 BITS); IF NO ZERO IS PRESENT, THE STRING WILL BE RIGHT JUSTIFIED.

3. AN EMPTY STRING CODE IS TREATED AS A CODE OF 6 (RIGHT-JUSTIFIED BCL STRING).

4. AN ASCII CODE OR EBCDIC CODE MAY BE USED ONLY WITH CHARACTERS FROM THE BCL CHARACTER SET. THE COMPILER PRODUCES INTERNALLY

A STRING OF EIGHT-BIT CHARACTERS WHICH REPRESENT THE SAME GRAPHICS AS THE GIVEN BCL CHARACTERS. FOR CHARACTERS WHICH ARE NOT IN THE BCL CHARACTER SET, THE STRING MUST BE WRITTEN AS A HEXADECIMAL STRING, WHERE EACH PAIR OF HEXADECIMAL CHARACTERS REPRESENTS THE INTERNAL CODE OF ONE ASCII OR EBCDIC CHARACTER.

5. THE QUOTE CHARACTER MAY APPEAR ONLY AT THE BEGINNING OF A SIMPLE STRING. STRINGS WITH INTERNAL QUOTES MUST BE BROKEN INTO SEPARATE SIMPLE STRINGS BY THE USE OF THREE QUOTES IN SUCCESSION. THE LAST TWO QUOTES MUST BE CONTIGUOUS.

6. THE MAXIMUM PERMISSIBLE LENGTH OF A STRING DEPENDS UPON THE CONTEXT IN WHICH THE STRING IS USED. POINTER OPERATIONS (REPLACE AND STRING COMPARE) AND LIST ELEMENTS WHICH CONSIST OF ONLY A STRING MAY BE REPRESENTED BY STRINGS UP TO 256 48-BIT WORDS IN LENGTH. STRINGS USED AS OPERANDS IN EXPRESSIONS ARE LIMITED TO A LENGTH OF 48 BITS.

7. WHEN A STRING IS FORMED FROM SIMPLE STRINGS OF DIFFERENT CHARACTER SIZES, THE FOLLOWING APPLIES:

   A. THE JUSTIFICATION SPECIFIED BY EACH STRING CODE AFTER THE FIRST IS IGNORED.

   B. EVERY CHARACTER IN A STRING IS ALIGNED AT A CHARACTER BOUNDARY APPROPRIATE FOR THE SIZE OF THAT CHARACTER. THIS MAY RESULT IN ZERO BITS BEING INSERTED BETWEEN SIMPLE STRINGS. FOR EXAMPLE, 6"8" 4"8" PRODUCES 00100000010000. NOTE THAT TWO BITS ARE INSERTED BETWEEN THE SIMPLE STRINGS SO THAT THE 4"8" FALLS ON A FOUR-BIT CHARACTER BOUNDARY. HOWEVER, 8"8" 4"8" REQUIRES NO SUCH ALIGNMENT.

   C. WHEN IT IS NECESSARY FOR THE COMPILER TO KNOW THE "LENGTH" IN CHARACTERS OF A STRING AND THE CHARACTER

SIZE (E.G., REPLACE AND STRING COMPARE), THE CHARACTER SIZE IS THE MAXIMUM CHARACTER SIZE OF ALL THE SIMPLE STRINGS; THE LENGTH IS THE SMALLEST NUMBER OF CHARACTERS (OF THE MAXIMUM CHARACTER SIZE) REQUIRED TO CONTAIN ALL THE BITS OF THE STRING.

5. GENERAL COMPONENTS

5.1. GENERAL

GENERAL COMPONENTS DIFFER FROM BASIC COMPONENTS IN THAT GENERAL COMPONENTS ARE USUALLY SUBSCRIPTED, OR HAVE ASSOCIATED PARAMETERS.

SYNTAX:

<GENERAL COMPONENT> ::= <BASIC COMPONENT> / <VARIABLE> /
<ITEM DESIGNATOR> / <VALUE DESIGNATOR> /
<EVENT DESIGNATOR> / <INTRINSIC>

SEMANTICS:

THE RELATIONSHIP BETWEEN GENERAL COMPONENTS AND EXPRESSIONS IS RECURSIVE. EXPRESSIONS ARE COMPOSED OF GENERAL COMPONENTS AND CONVERSELY GENERAL COMPONENTS MAY BE COMPOSED IN PART OF EXPRESSIONS. IN THE SIMPLEST CASE A <GENERAL COMPONENT> IS COMPOSED OF A <BASIC COMPONENT>; E.G., A <SIMPLE VARIABLE> IS DEFINED TO BE AN <IDENTIFIER>.

5.2. VARIABLES

SYNTAX:

<VARIABLE> ::= <SIMPLE VARIABLE> / <SUBSCRIPTED VARIABLE>
<SIMPLE VARIABLE> ::= <IDENTIFIER>
<SUBSCRIPTED VARIABLE> ::= <ARRAY IDENTIFIER> [<SUBSCRIPT LIST>]
<ARRAY IDENTIFIER> ::= <IDENTIFIER> / <ARRAY INTRINSIC>
<SUBSCRIPT LIST> ::= <SUBSCRIPT> / <SUBSCRIPT LIST> , <SUBSCRIPT>
<SUBSCRIPT> ::= <ARITHMETIC EXPRESSION>

SEMANTICS:

1. A SUBSCRIPTED VARIABLE IS AN ARRAY IDENTIFIER WITH A SUBSCRIPT LIST. THE ARRAY IDENTIFIER REFERS TO A SET OF VALUES. AN ARRAY IDENTIFIER WITH A SUBSCRIPT LIST REFERS TO A SINGLE VALUE OR A SUBSET OF VALUES.

2. THE TOTAL NUMBER OF SUBSCRIPTS IN A SUBSCRIPT LIST MUST EQUAL THE NUMBER OF DIMENSIONS GIVEN IN THE ARRAY DECLARATION.

3. EACH SUBSCRIPT EXPRESSION IN A SUBSCRIPT LIST IS EVALUATED FROM LEFT TO RIGHT.

4. IF, UPON EVALUATION, A SUBSCRIPT EXPRESSION YIELDS A VALUE OF TYPE REAL, IT WILL BE ROUNDED AS FOLLOWS:

   INTEGRAL SUBSCRIPT VALUE = ENTIER (VALUE OF SUBSCRIPT EXPRESSION + 0.5)

5. IF THE VALUE OF A SUBSCRIPT FALLS OUTSIDE THE BOUNDS DECLARED FOR THAT DIMENSION, THERE WILL BE AN "INVALID INDEX" ERROR AND TERMINATION OF THE PROCESS.

EXAMPLES:

SIMPLE VARIABLES:

```
A
ABSOLUTE
ABS2
```

SUBSCRIPTED VARIABLES:

```
A[5]
QTY[Q+7,VXN,Z]
Q[7,2]
A[ITH]
KRONECKER[ITH+2,JTH-ITH]
MAXQ [ IF BETA = 30 THEN -2 ELSE K+ 2 ]
```

5.3. ITEM DESIGNATORS

SYNTAX:

<ITEM DESIGNATOR> ::= <ARITHMETIC ITEM> / <WORD ITEM>

<ARITHMETIC ITEM> ::= <ITEM>

<WORD ITEM> ::= <ITEM>

<ITEM> ::= <ITEM IDENTIFIER> @ <REFERENCE EXPRESSION>

<ITEM IDENTIFIER> ::= <IDENTIFIER> / <ITEM>

SEMANTICS:

ITEMS ARE USED WHEN IT IS NECESSARY TO HAVE MORE THAN ONE TYPE OF VARIABLE IN A QUEUE ELEMENT, IN PARTICULAR, ITEMS MAY BE TREATED AS CALL-BY-NAME VARIABLES. IN GENERAL, <ITEM IDENTIFIER>S ARE PROVIDED TO REFERENCE PARAMETERS IN A QUEUE STRUCTURE. THE <ITEM IDENTIFIER> MAY ALSO BE USED AS AN INDEX TO AUTOMATICALLY PROVIDE THE ORDINAL NUMBER OF A QUEUE ELEMENT CORRESPONDING TO THE INDICATED ITEM.

EXAMPLES:

```
FIRSTGO @ WAITCHANELQUE[CHANELNUMBER]
FAOWST@(REFERENCE(A[2,1]))
```

5.4. VALUE DESIGNATORS

SYNTAX:

<VALUE DESIGNATOR> ::= <VALUE ARRAY IDENTIFIER> [<SUBSCRIPT LIST>]

<VALUE ARRAY IDENTIFIER> ::= <IDENTIFIER>

SEMANTICS:

A VALUE DESIGNATOR REFERENCES A PARTICULAR ELEMENT OF A VALUE ARRAY

5.5. EVENT DESIGNATORS

SYNTAX:

<EVENT DESIGNATOR> ::= <EVENT IDENTIFIER> / <EVENT ITEM> /
<EVENT ARRAY IDENTIFIER> [<SUBSCRIPT LIST>] /
<EVENT ARRAY ITEM> [<SUBSCRIPT LIST>]
<EVENT ITEM> ::= <ITEM>
<EVENT ARRAY ITEM> ::= <ITEM>

SEMANTICS:

1. AN EVENT DESIGNATOR DESIGNATES AN EVENT QUANTITY.

2. THE ITEM IDENTIFIER OF AN ITEM USED AS AN EVENT ITEM MUST BE DECLARED AS AN EVENT.

3. THE ITEM IDENTIFIER OF AN ITEM USED AS AN EVENT ARRAY ITEM MUST APPEAR IN A DECLARATION STATEMENT DECLARING IT AS AN EVENT ARRAY.

EXAMPLES:

```
E1
ZEUS [2,3]
```

5.6. INTRINSICS

SYNTAX:

```
<INTRINSIC> ::= <ARRAY INTRINSIC> / <FUNCTION INTRINSIC> /
      <PROCEDURE INTRINSIC> / <WORD INTRINSIC>
<ARRAY INTRINSIC> ::= M / MEMORY / REGISTERS / STACK /
      STACKVECTOR / WORDSTACK
<FUNCTION INTRINSIC> ::= <ARITHMETIC INTRINSIC> /
      <BOOLEAN INTRINSIC>
<ARITHMETIC INTRINSIC> ::= ABS / BOOLEAN / DABS / DECIMAL /
      DINTEGER / DMAX / DMIN / DNABS / DOUBLE / DSCALELEFT /
      DUPLICATE / ENTIER / EVAL / EXCHANGE / FIRSTONE /
      FIRSTWORD / INTEGER / LISTLOOKUP / MASKSEARCH / MAX / MIN /
      MYSELF / NABS / NAME / NORMALIZE / OCRX / ONES / REAL /
      SCALELEFT / SCALERIGHT / SCALERIGHTF / SCALERIGHTT /
      SCANIN / SECONDWORD / SIZE / STUFF / XSIGN
<BOOLEAN INTRINSIC> ::= AVAILABLE / BUSY / HAPPENED /
      OVERFLOW / TOGGLE / UNLOCK
<PROCEDURE INTRINSIC> ::= ALLOW / BUZZ / DISALLOW /
      EXIT / HEYOU / IIO /
      MOVESTACK / PAUSE / RETURN / SCANOUT /
      STOP / STOREITEM / TIMER / TOUCH / ZAP
<WORD INTRINSIC> ::= TOPOFSTACK
```

SEMANTICS:

AN "INTRINSIC" IS DEFINED TO BE SOMETHING WHICH THE COMPILER ALREADY UNDERSTANDS; I.E., IT NEED NOT BE "DECLARED". THE PURPOSE OF AN INTRINSIC IS TO PROVIDE THE PROGRAMMER WITH FACILITIES SUCH THAT HE DOES NOT HAVE TO DO THE DETAIL CODING.

IN MANY CASES, THE FOLLOWING INTRINSICS NOT ONLY PERFORM A CERTAIN ACTION, BUT THEY RETURN SOMETHING TO THE TOP OF THE STACK. THE ESPOL COMPILER GENERATES A "DELETE" OPERATION IF IT SEES THE

INTRINSIC USED AS A PROCEDURE RATHER THAN AS A FUNCTION.

ARRAY INTRINSICS

INTRINSIC	RESULT
M[N] MEMORY [N]	THESE ARE EQUIVALENT. THEY ARE ONE-DIMENSIONAL WORD ARRAYS REFERENC MEMORY.
REGISTERS[N]	THIS IS A ONE DIMENSIONAL ARRAY WHICH REFERENCES THE VARIOUS MACHINE REGISTERS.
STACK[X,Y]	THIS IS TWO-DIMENSIONAL REAL ARRAY, WHERE X GIVES THE STACK NUMER AND Y THE WORD NUMBER.
STACKVECTOR[N]	THIS IS A ONE-DIMENSIONAL WORD ARRAY REFERRING TO A PARTICULAR ROW OF A STACK.
WORDSTACK[X,Y]	THIS IS EQUIVALENT TO STACK EXCEPT THAT IT IS A WORD ARRAY.

ARITHMETIC INTRINSICS

INTRINSIC	OPERATOR	RESULT
ABS(AE)	BRST	RETURNS ABSOLUTE VALUE OF AE.
BOOLEAN(AE)		RETURNS THE VALUE OF AE AS A BOOLEAN. IF AE IS DOUBLE THEN AE WILL BE TRUNCATED FIRST.

DABS(DAE)	BRST	RETURNS THE ABSOLUTE VALUE OF THE DOUBLE DAE.
DECIMAL(N)	SCRF	"OUTPUT CONVERT" THE BINARY REPRESENTATION OF N TO DECIMAL REPRESENTATION. IF N IS A POINTER THE CHARACTER SIZE IS DETERMINED BY THE COMPILER. IF N IS AN OPERAND IT IS TREATED AS AN EIGHT BIT/CHARACTER (EBCDIC) STRING.
DINTEGER(AE)	NTGD	CONVERTS AE INTO A DOUBLE-PRECISION INTEGER.
DMAX(A1,...,AN)		RETURNS THE MAXIMUM OF THE VALUES A1,...,AN.
DMIN(A1,...AN)		RETURNS THE MINIMUM OF THE VALUES A1,...,AN.
DNABS(DAE)	BSET	RETURN THE NEGATIVE ABSOLUTE VALUE OF THE DOUBLE DAE.
DOUBLE(PE,AE)	ICVD XTND	SAME AS INTEGER(PE,AE) EXCEPT THAT THE MAXIMUM VALUE OF AE IS 24 AND THE CHARACTERS IN THE STRING REFERENCED BY THE POINTER MUST BE LE THAN 8*26-1 = 302231454903657293676 5
DOUBLE(R1,R2)	JOIN	RETURNS THE DOUBLE VALUE WITH THE FIRST PART EQUAL TO R1 AND THE SECOND PART EQUAL TO R2.
DOUBLE(RE)	XTND	RETURNS THE DOUBLE VALUE EQUAL TO THE SINGLE RE.

DSCALELEFT(V,F)	SCLF (DSLF)	RETURNS THE DOUBLE INTEGER VALUE OF DOUBLE ARITHMETIC-VARIABLE V MULTIPL BY 10 RAISED TO THE F POWER. THE OPERATOR EMPLOYED IS SCLF IF F IS AN INTEGER CONSTANT; DSLF IS USED IF F IS AN INTEGER VARIABLE.
DUPLICATE	DUPL	ANYWHERE A <PRIMARY> IS ALLOWED, THIS INTRINSIC MAY BE USED. IT SIMPLY DUPLICATES THE TOP WORD OF THE STACK, THEREBY ADDING ANOTHER WORD TO THE TOP OF THE STACK.
ENTIER(N)		ROUNDS THE ABSOLUTE VALUE OF N DOWN TO AN INTEGER. RETURN THE RESULT WITH THE PROPER SIGN.
EVAL(X)	EVAL	LEAVES IN THE TOP OF THE STACK AN INDEXED DATA DESCRIPTOR OR STUFFED INDIRECT REFERENCE WORD WHICH POINTS TO THE LOCATION IN MEMORY OF THE ARITHMETIC TYPE VARIABLE X.
EXCHANGE(V)	EXCH	FIRST THE VARIABLE V IS BROUGHT TO THE TOP OF THE STACK. THEN THE TWO TOP WORDS OF THE STACK ARE EXCHANGED.
FIRSTONE(N)	LOG2	RETURNS BIT NUMBER (PLUS 1) OF FIRST NONZERO BIT IN N. FIRSTONE RETURNS ZERO IF THERE ARE NO NONZERO BITS.
FIRSTWORD(EVNT)		RETURNS THE HIGH-ORDER WORD OF EVNT WITH A SINGLE-PRECISION TAG. EVNT MAY BE AN EVENT,

		EVENT ARRAY OR DOUBLE-PRECISION OPERAND.
FIRSTWORD(D,V)		THE SINGLE-PRECISION VALUE RETURNED WILL BE THE FIRST-WORD OF THE DOUBLE D, AND THE SECOND-WORD OF D WILL BE STORED IN VARIABLE V.
INTEGER(AE)	NTGR	ROUNDS AE. (I.E., ENTIER(AE + 0.5))
INTEGER (PE,AE)	UABD ICVD	"INPUT CONVERTS" AND RETURNS, AS AN INTEGER, TE DECIMAL VALUE OF AE CHARACTERS, STARTING WITH THE CHARACTER POINTED AT BY POINTER EXPRESSION PE.
LISTLOOKUP (AE,AR,AE)	LLLU	RETURNS AN INTEGER VALUE. IT IS USED AS FOLLOWS: I := LISTLOOKUP (X,A,N) WHERE X IS THE ARGUMENT, A IS A ONE-DIMENSIONAL ARRAY IDENTIFIER OR ARRA ROW, AND N IS AN INDEX INTO THAT ARR
MASKSEARCH (AE,AE,AR)	SRCH	RETURNS AN INTEGER VALUE. ITS USED AS FOLLOWS: I := MASKSEARCH (X,M,A) WHERE X IS THE WORD TO LOOK FOR, M IS THE MASK VALUE, AND A IS A ONE-DIMENSIONAL ARRAY IDENTIFIER, AN ARRAY ROW, OR A FULLY SUBSCRIPTED VARIABLE.

		THE OPERATOR BEGINS AT THE LAST WORD OF THE ARRAY IF A IS NOT INDEXED (ARRAY ID OR ARRAY ROW CASE) OR AT THE SPECIFIED WORD. IT PERFORMS A LOGICAL "AND" OPERATION WITH THE MASK AND THEN COMPARES THE RESULT WITH THE VALUE 0 IF EQUAL, THE INDEX IS THE RESULT. IF NOT, THE OPERATOR REPEATS THE SAME FUNCTION ON EACH WORD WORKING TOWARD THE FIRST WORD OF THE ARRAY. IF NO MASKED VALUE MATCHES THE VALUE OF X, THE RESULT IS -1. NOTE: THE OPERATOR IS DESIGNED FOR ARRAYS WHOSE LOWER BOUNDS ARE 0. THE INDEX RETURNED IS ALWAYS IN THE RANGE 0 THROUGH N-1, WHERE THE ROW CONTAINS N ELEMENTS.
MAX(A1,...AN)		RETURNS THE MAXIMUM OF THE VALUES A1,...AN.
MIN(A1,...,AN)		RETURNS THE MINIMUM OF THE VALUES A1,...,AN.
MYSELF	WHOI	THE NUMBER OF THE PROCESSOR WHICH IS RUNNING.
NABS(AE)	BSET	RETURN NEGATIVE ABSOLUTE VALUE OF AE
NAME(VAR)	NAME	RETURNS THE ADDRESS-COUPLE OF THE VARIABLE VAR.
NORMALIZE(AE)	SNGL	AE IS CHANGED TO A ROUNDED NORMALIZED, SINGLE-PRECISION OPERAND. NOTE THAT THIS TAKES PLACE

		THE TOP OF THE STACK AND MEMORY IS NOT ALTERED.
OCRX(INDEX,OCW)	OCRX	THE OCRX OPERATOR LEAVES THE VALUE OF (OCW OFFSET+ (OCW LENGTH x (INDEX -1))) IN THE TOP OF THE STACK.
ONES(N)	CBON	RETURNS NUMBER OF NONZERO BITS IN N.
REAL(BE)		RETURNS THE REAL VALUE REPRESENTED B THE BOOLEAN EXPRESSION BE. ALL BITS OF BE ARE USED.
REAL(DBL)	SNGL	RETURNS THE DOUBLE EXPRESSION DBL NORMALIZED AND ROUNDED TO A SINGLE-PRECISION ITEM
REAL(PE,AE)	SISO	RETURNS, AS A REAL VALUE, A BIT IMAG OF THE STRING OF AE CHARACTERS START WITH THE CHARACTER INDICATED BY THE POINTER EXPRESSION PE. ALL BITS OF CHARACTER ARE USED.
SCALELEFT(V,F)	SCLF (DSLF)	RETURNS THE INTEGER VALUE OF THE ARITHMETIC TYPE VARIABLE MULTIPLIED BY 10 RAISED TO THE F POWER. THE OPERATOR EMPLOYED IS SCLF IF F IS AN INTEGER CONSTANT; DSLF IS USED IF F IS AN INTEGER VARIABLE.
SCALERIGHT(V,F)	SCRR (DSRR)	RETURNS THE ROUNDED INTEGER VALUE OF THE ARITHMETIC TYPE VARIABLE DIVIDED BY 10 RAISED TO THE F POWER. THE OPERATOR EMPLOYED IS SCRR IF F

		IS AN INTEGER CONSTANT; DSRR IS USED IF F IS AN INTEGER VARIABLE.
SCALERIGHTF(V,F)	SCRD (DSRF)	RETURNS THE REAL REMAINDER OF THE ARITHMETIC-TYPE VARIABLE V DIVIDED BY 10 RAISED TO THE F POWER. THE NUMBER OF SIGNIFICANT DIGITS RETURNED IS F. THE OPERATOR EMPLOYED IS SCRF IF F IS AN INTEGER CONSTANT; DSRF IS USED IF F IS AN INTEGER VARIABLE.
SCALERIGHTT(V,F)	SCRF (DSRF)	RETURNS THE TRUNCATED INTEGER VALUE ARITHMETIC-TYPE VARIABLE V DIVIDED B RAISED TO THE F POWER. THE OPERATOR EMPLOYED IS SCRT IF F IS AN INTEGER CONSTANT; DSRT IS USED IF F IS AN INTEGER VARIABLE.
SCANIN(N)	SCNI	GETS THE INFORMATION INDICATED BY N AND LEAVES IT IN THE TOP OF STACK.
SECONDWORD (EVNT)		RETURNS THE SECOND WORD WITH A SINGLE-PRECISION TAG. EVNT MAY BE AN EVENT, OR A DOUBLE-PRECISION OPERAND.
SIZE(Q)		RETURN NUMBER OF ENTRIES IN THE QUEUE Q.
STUFF(X)	NAMC STFF	LEAVES IN THE TOP OF THE STACK A STUFFED INDIRECT REFERENCE WORD WHICH POINTS TO THE LOCATION IN MEMORY OF X (WHICH CAN BE A PROCEDUR IDENTIFIER, ARRAY IDENTIFIER, EVENT IDENTIFIER, INTERRUPT IDENTIFIER,

		OR QUEUE). IF X IS AN INTERRUPT IDENTIFIER, THE RESULTING SIRW POINTS TO THE SECOND WORD OF THE INTERRUPT.
XSIGN	SXSN	SETS EXTERNAL-SIGN FLIP-FLOP EQUAL TO SIGN OF TOP OF STACK AND RETAIN THE VALUE OF THE TOP OF THE STACK.

BOOLEAN INTRINSICS

INTRINSIC	OPERATOR	RESULT
AVAILABLE(E)		RETURNS TRUE IF THE EVENT E IS AVAILABLE (BIT2=0).
BUSY(X)		TRUE IF X IS LOCKED (BIT0=1).
HAPPENED(E)		RETURNS TRUE IF E HAS "HAPPENED" (BIT0=1).
OVERFLOW	ROFF	RETURNS VALUE OF OVERFLOW FLIP-FLOP.
TOGGLE	RTFF	RETURNS VALUE OF TRUE-FALSE FLIP-FLOP.
UNLOCK(X)		TRUE IF X WAS PREVIOUSLY LOCKED. UNLOCK LEAVES X UNLOCKED.

PROCEDURE INTRINSICS

INTRINSIC	OPERATOR	RESULT

ALLOW	EEXI	ENABLES EXTERNAL INTERRUPTS.
BUZZ(X)	DEXI RDLK	A ONE(1) IS CONTINUALLY INTERCHANGED WITH THE CURRENT CONTENTS OF X UNTIL BIT0 OF X IS OFF, AT WHICH POINT THAT VALUE IS RETURNED. (NOTE THE DEXI OPERATOR WHICH IS FIRST AUTOMAGICALLY EXECUTED.)
DISALLOW	DEXI	DISABLES EXTERNAL INTERRUPTS.
EXIT	EXIT	GENERATES EXIT OPERATOR.
HEYOU	HEYU	INTERRUPTS OTHER PROCESSORS.
IIO(A,U)	SCNO	INITIATES I/O, USING UNIT SPECIFIED BY U AND ARRAY ROW "A" IS PASSED TO THE MULTIPLEXOR.
MOVESTACK(N)	MVST	TRANSFERS CONTROL TO STACK N.
PAUSE	IDLE	IDLES UNTIL AN EXTERNAL INTERRUPT OCCURS.
RETURN(N)	RETN	RETURNS THE VALUE N AS A RESULT. NOTE THAT "BLOCKEXIT CODE" IS AVOIDED VIA THIS INTRINSIC.
SCANOUT(V,T)	SCNO	T AND V CAN BE ARITHMETIC TYPE OR WORD TYPE ESPRESSIONS. V IS THE VALUE WHICH IS SCANNED OUT AND T INDICATES WHAT TYPE OF SCANOUT WILL TAKE PLACE.

STOP	HALT	MAY HAVE ZERO, OR ONE, OR TWO ARGUME STOP LOADS THE ARGUMENTS INTO A AND B REGISTERS AND HALTS THE PROCESSOR IF THE CONDITIONAL HALT SWITCH IS ON. DEPENDING ON THE NUMBER OF ARGUMENTS THE COMPILER GENERATES THE APPROPRIA NUMBER OF "DELETES" TO EMPTY THE STACK AFTER THE HALT OPERATOR.
STOREITEM(FITM,AITM)		STORES THE NAME OF AITM IN THE LOCATION SPECIFIED BY FITM.
TIMER(N)	SINT	SETS INTERVAL TIMER TO VALUE N, WHERE N IS IN 512 MICROSECOND INCREMENTS.
TOUCH(EXP)		THE EXPRESSION EXP IS EVALUATED AND THEN DELETED. IT IS PRIMARILY INTEN TO ALLOW SIDE EFFECTS SUCH AS PRESENCEBIT ACTION.
ZAP	ZAP	ALSO KNOWN AS "SUPER HEYOU" BECAUSE IT UNCONDITIONALLY INTERRUPTS ALL OTHER PROCESSORS WITH AN ALARM INTERRUPT.

WORD INTRINSIC

INTRINSIC	RESULT
TOPOFSTACK	THIS INTRINSIC REFERENCES THE TOP OF THE STACK. IT MAY APPEAR ANYWHERE A <WORD VARIABLE> COULD APPEAR. FOR EXAMPLE: TOPOFSTACK := A := I+J;

```
OR RETURN (TOPOFSTACK).
```

6. PROGRAMS, BLOCKS, AND COMPOUND STATEMENTS

SYNTAX:

<PROGRAM> ::= <BLOCK> . <SPACE> / <COMPOUND STATEMENT> . <SPACE>
<BLOCK> ::= <BLOCK HEAD> ; <COMPOUND TAIL>
<BLOCK HEAD> ::= BEGIN <DECLARATION> / <BLOCK HEAD> ; <DECLARATION>
<COMPOUND TAIL> ::= <STATEMENT> END / <STATEMENT> ; <COMPOUND TAIL>
<COMPOUND STATEMENT> ::= BEGIN <COMPOUND TAIL>
<SPACE> ::= <SINGLE SPACE> / <SPACE> <SINGLE SPACE>

SEMANTICS:

1. LOCAL: EVERY BLOCK AUTOMATICALLY INTRODUCES A NEW LEVEL OF NOMENCLATURE FOR IDENTIFIERS DECLARED IN THAT BLOCK. THAT IS, ANY IDENTIFIER OCCURRING WITHIN THE BLOCK MAY, THROUGH AN APPROPRIATE DECLARATION, BE DECLARED "LOCAL" TO THE BLOCK. THE MEANING OF LOCAL IS AS FOLLOWS:

    A. THE VALUE(S) REPRESENTED BY THE IDENTIFIER INSIDE THE BLOCK WILL NOT BE RECOGNIZED BY THAT IDENTIFIER OUTSIDE THE BLOCK.

    B. CORRESPONDINGLY, ANY VALUE(S) REPRESENTED BY THE IDENTIFIER OUTSIDE THE BLOCK WILL NOT BE RECOGNIZED BY THAT IDENTIFIER INSIDE THE BLOCK.

2. GLOBAL: AN IDENTIFIER OCCURRING WITHIN A BLOCK AND NOT DECLARED WITHIN THAT BLOCK IS "GLOBAL" TO THE BLOCK. THE MEANING OF GLOBAL IS AS FOLLOWS: THE IDENTIFIER REPRESENTS THE SAME VALUE(S) INSIDE THE BLOCK AS IN THE LEVEL(S) OUTSIDE IT, UP TO AND INCLUDING THE LEVEL AT WHICH IT IS DECLARED.

EXAMPLES:

PROGRAM:

```
BEGIN REAL V; V←V+V+I END.
```

BLOCKS:

```
BEGIN REAL Q; Q←Q+I END
BEGIN
INTEGER I,K; REAL W;
FOR I←I STEP I UNTIL M DO
FOR K←I+I STEP I UNTIL M DO
BEGIN
W←A[I,K];
A[I,K]←A[K,I];
A[K,I]←W
END
END
```

COMPOUND STATEMENTS:

```
BEGIN X←0;FOR Y←I STEP I UNTIL N DO X←X+A[Y];
IF X>Q THEN GO TO STOP ELSE IFX>W←2 THEN GO TO W;
AW←ST←X+BOB
END
BEGIN V←V+I END
BEGIN Q←Q+I;V←V+I END
BEGIN END
```

BLOCK HEADS:

```
BEGIN REAL V
BEGIN REAL V; BOOLEAN Q
```

COMPOUND TAILS:

```
V←V+I END
```

```
Q←Q+I;V←Q+V END
```

7. DECLARATIONS

7.1. GENERAL

DECLARATIONS DEFINE FOR THE COMPILER CERTAIN PROPERTIES OF IDENTIFIERS IN THE BLOCK OF A PROGRAM. DECLARATIONS ARE USED ONLY DURING COMPILATION AND ARE, THEREFORE, NOT EXECUTED.

SYNTAX:

<DECLARATION> ::= <ARRAY DECLARATION> /
<DEFINE DECLARATION> / <DEFINE INVOCATION> /
<DIRECT ARRAY DECLARATION> / <EVENT DECLARATION> /
<EVENT ARRAY DECLARATION> / <FIELD DECLARATION> /
<FILE DECLARATION> / <INTERRUPT DECLARATION> /
<LABEL DECLARATION> / <LAYOUT DECLARATION> /
<MONITOR DECLARATION> / <PICTURE DECLARATION> /
<PROCEDURE DECLARATION> / <QUEUE ARRAY DECLARATION> /
<QUEUE DECLARATION> / <TYPE DECLARATION> /
<VALUE ARRAY DECLARATION>

SEMANTICS:

1. SCOPE: THE SCOPE OF THE IDENTIFIER IS THE BLOCK IN WHICH IT IS DECLARED EXCEPT FOR THE FOLLOWING:

   A. FORMAL SYMBOLS IN A DEFINE DECLARATION; THE SCOPE OF THESE FORMAL SYMBOLS CONSISTS OF THE CORRESPONDING DEFINES.

   B. FORMAL PARAMETERS IN A PROCEDURE DECLARATION; THE SCOPE OF THESE FORMAL PARAMETERS IS THE PROCEDURE DECLARATION.

C. INVISIBLE FORMAL ITEMS IN A QUEUE DECLARATION; THE SCOPE OF THESE FORMAL ITEMS IS THE QUEUE DECLARATION

2. LOCAL: AN IDENTIFIER IS SAID TO BE "LOCAL" TO THE BLOCK IN WHICH IT IS DECLARED. THAT IS, A VALUE REPRESENTED BY THE IDENTIFIER INSIDE THE BLOCK IS ISOLATED FROM THAT IDENTIFIER OUTSIDE THE BLOCK.

3. AT THE TIME OF EXIT FROM THE BLOCK, ALL IDENTIFIERS WHICH ARE DECLARED FOR THE BLOCK LOSE THEIR SIGNIFICANCE.

4. GLOBAL: AN IDENTIFIER IS SAID TO BE "GLOBAL" TO A BLOCK IF IT MEETS THE FOLLOWING CONDITIONS:

   A. IT IS NOT DECLARED IN THE BLOCK.

   B. IT IS DECLARED IN AN EXTERIOR BLOCK.

5. WHEN THE BLOCK IS ENTERED, ALL IDENTIFIERS DECLARED FOR THE BLOCK ASSUME THE SIGNIFICANCE IMPLIED BY THE NATURE AND ORDER OF THE DECLARATIONS IN THE BLOCK HEAD.

6. SOME IDENTIFIERS MAY BE DECLARED WITH THE DECLARATOR "OWN". THIS DECLARATOR CAUSES THE IDENTIFIED QUANTITY TO RETAIN ITS VALUE(S) FROM ONE EXIT OF A BLOCK TO THE ENTRY INTO THAT BLOCK

7. AN IDENTIFIER MAY NOT BE DECLARED TO REPRESENT MORE THAN ONE ENTITY IN A SINGLE BLOCK HEAD, FORMAL SYMBOL LIST, FORMAL PARAMETER LIST, OR FORMAL ITEM LIST.

7.2. ARRAY DECLARATIONS

SYNTAX:

```
<ARRAY DECLARATION> ::= <ARRAY KIND> ARRAY <ARRAY LIST>
<ARRAY KIND> ::= <EMPTY> / <LOCAL OR OWN TYPE> /
      SAVE <LOCAL OR OWN TYPE>
<LOCAL OR OWN TYPE> ::= <TYPE> / OWN <TYPE>
<ARRAY LIST> ::= <ARRAY SEGMENT> / <ARRAY LIST> , <ARRAY SEGMENT>/
      <INITIALIZED ARRAY> / <ARRAY LIST> , <INITIALIZED ARRAY>
<ARRAY SEGMENT> ::= <ARRAY IDENTIFIER> <ADDRESS PART>
      [<BOUND LIST>] / <ARRAY IDENTIFIER> <ADDRESS PART> ,
      <ARRAY SEGMENT>
<BOUND LIST> ::= <BOUND> / <BOUND LIST> , <BOUND>
<BOUND> ::= <WORD COUNT> / *
<WORD COUNT> ::= <ARITHMETIC EXPRESSION>
<INITIALIZED ARRAY> ::=
      <ARRAY IDENTIFIER> <REPLACEMENT OPERATOR> (<CONSTANT LIST>)
```

SEMANTICS:

1. AN ARRAY DECLARATION DEFINES ONE OR SEVERAL IDENTIFIERS TO REPRESENT ARRAYS OF SUBSCRIPTED VARIABLES; IT ALSO PROVIDES THE FOLLOWING:

    A. THE DIMENSIONS OF THE ARRAY.

    B. THE BOUNDS OF THE SUBSCRIPTS.

    C. THE TYPES OF THE VARIABLES.

2. THE VALUE OF AN ARRAY IDENTIFIER IS A DATA DESCRIPTOR REPRESENTING THE ORDERED SET OF VALUES OF THE CORRESPONDING ARRAY OF SUBSCRIPTED VARIABLES.

3. AN EMPTY ARRAY KIND MEANS A DEFAULT DECLARATION OF REAL.

4. THE LOCATION SPECIFIED BY THE ADDRESS PART MUST CONTAIN A DATA DESCRIPTOR OR AN INDIRECT REFERENCE WORD POINTING TO A DATA DESCRIPTOR.

5. IF "SAVE" IS SPECIFIED, THE ARRAY WILL BE MADE NON-OVERLAYABLE WHEN IT IS ALLOCATED. NOTE THAT THE COMBINATION OF "SAVE" AND A <BOUND> OF "*" ARE NOT ALLOWED.

6. THE BOUND LIST GIVES THE NUMBER OF VALUES FOR EACH SUBSCRIPT TAKEN IN ORDER FROM LEFT TO RIGHT.

7. A BOUND OF <WORD COUNT> INDICATES THAT <WORD COUNT> IS THE TOTAL NUMBER OF WORDS IN THE <ARRAY ROW>.

8. A BOUND OF "*" MEANS THAT THE PROGRAMMER IS RESPONSIBLE FOR THE ALLOCATION OF THE SPACE FOR THE ARRAY, BUT THE COMPILER PRODUCES A ZERO WORD WITH A TAG OF 5.

9. EXPRESSIONS USED AS BOUNDS ARE EVALUATED ONCE, FROM LEFT TO RIGHT, UPON ENTRANCE INTO THE BLOCK. THESE EXPRESSIONS CAN DEPEND ONLY ON VARIABLES AND PROCEDURES WHICH ARE NONLOCAL TO THE BLOCK FOR WHICH THE ARRAY DECLARATION IS VALID OR WHICH ARE LOCAL AND HAVE INITIAL VALUE PARTS. ARRAYS DECLARED IN THE OUTERMOST BLOCK MUST USE CONSTANT OR "*" BOUNDS.

10. A BOUND OF "*" MAY NOT APPEAR TO THE LEFT OF AN EXPRESSION USED AS A BOUND IN THE SAME BOUND LIST.

11. WHEN AN ARRAY SEGMENT INCLUDES NON-EMPTY ADDRESS PARTS, A BOUND OF "*" MUST BE USED.

12. DYNAMIC OWN ARRAYS ARE NOT PERMITTED.

EXAMPLES:

ARRAY DECLARATIONS:

```
OWN REAL ARRAY AZ, BZ = (3,4), CZ [27]
ARRAY BZ[10]
```

ARRAY LISTS:

```
AZ,BZ=(3,4),CZ[27]
PQ,RQ[31]
PC[15],RC[31]
```

ARRAY SEGMENTS:

```
C7C=(-2,5),C8C=C7C,C9C=(4)[*]
C27[1022]
```

BOUND LISTS:

```
1,56
7
1,*
```

BOUNDS:

```
27
*
IF A THEN 1 ELSE Q-7
```

7.3. DEFINE DECLARATIONS AND DEFINE INVOCATIONS

SYNTAX:

```
<DEFINE DECLARATION> ::= DEFINE <DEFINITION LIST>
<DEFINITION LIST> ::= <DEFINITION> / <DEFINITION LIST> ,
      <DEFINITION>
<DEFINITION> ::= <DEFINED IDENTIFIER>
      <FORMAL SYMBOL PART> = <TEXT> #
<DEFINED IDENTIFIER> ::= <IDENTIFIER>
<FORMAL SYMBOL PART> ::= <EMPTY> / (<FORMAL SYMBOL LIST>)
<FORMAL SYMBOL LIST> ::= <FORMAL SYMBOL> / <FORMAL SYMBOL LIST>,
      <FORMAL SYMBOL>
<FORMAL SYMBOL> ::= <IDENTIFIER>
<TEXT> ::= ∠ ANY SEQUENCE OF VALID <CHARACTER>S NOT INCLUDING
      A FREE # ↘
<DEFINE INVOCATION> ::= <DEFINED IDENTIFIER> <ACTUAL TEXT PART>
<ACTUAL TEXT PART> ::= <EMPTY> / (<CLOSED TEXT LIST>) /
      [ <CLOSED TEXT LIST> ]
<CLOSED TEXT LIST> ::= <CLOSED TEXT> / <CLOSED TEXT LIST>,
      <CLOSED TEXT>
<CLOSED TEXT> ::= ∠ AN ACTUAL TEXT NOT CONTAINING
      UNMATCHED BRACKETING SYMBOLS OR UNBRACKETED COMMAS↘
```

SEMANTICS:

1. THE <DEFINE DECLARATION> ASSOCIATES ESPOL SOURCE LANGUAGE TEXTS WITH DEFINE IDENTIFIERS.

2. A <DEFINE INVOCATION> CAUSES THE REPLACEMENT OF THE <DEFINED IDENTIFIER> BEING INVOKED BY THE <TEXT> WHICH IS ASSOCIATED WITH THE IDENTIFIER. NOTE THAT A <DEFINE INVOCATION> CAUSES THE GENERATION OF IN-LINE CODE AT COMPILE TIME.

3. IF THE DEFINITION OF A <DEFINED IDENTIFIER> INCLUDES ANY

FORMAL SYMBOLS, ANY APPEARANCE OF THESE SYMBOLS IN THE <TEXT> OF THE DEFINITION(BUT NOT IN A STRING OR COMMENT) WILL BE REPLACED BY THE CORRESPONDING <ACTUAL TEXT PART>.

4. ALTHOUGH THE INVOCATION LOOKS IDENTICAL TO A <PROCEDURE STATEMENT> OR <SUBSCRIPTED VARIABLE>, BEWARE OF "PASSING" AN <ARITHMETIC EXPRESSION> INTO A <DEFINE INVOCATION>. IF THERE IS ANY DOUBT, SET THE $ CODE OPTION PRIOR TO THE INVOCATION AND VERIFY THE GENERATED CODE. IN MOST INSTANCES, EXTRA PARENTHESES INSIDE THE INVOCATION WILL ELIMINATE ANY PROBLEMS.

5. THE WORD "COMMENT" IS RECOGNIZED IN A <TEXT>, WITH THE RESULT THAT IT AND ALL CHARACTERS UP TO AND INCLUDING THE NEXT SEMICOLON ARE DELETED FROM THE <TEXT> , NO <TEXT> MAY INCLUDE AN INCOMPLETE COMMENT.

6. IN A <CLOSED TEXT LIST>, THE CLOSED TEXTS ARE SEPARATED BY COMMAS, AND THE <CLOSED TEXT LIST> IS TERMINATED BY A RIGHT PARENTHESES OR BRACKET.

7. IN A <CLOSED TEXT>, A COMMA MAY APPEAR ONLY BETWEEN MATCHING BRACKETING SYMBOLS. NO UNMATCHED OR UNPAIRED BRACKETS MAY APPEAR.

8. THE SCOPE OF A <FORMAL SYMBOL> IS THE <TEXT> OF THE DEFINITION IN WHICH THE <FORMAL SYMBOL> APPEARS.

9. BRACKETING SYMBOLS ARE [ ], ( ), AND THE GROUP CONSISTING OF THE FOLLOWING:

```
DEFINE = # ;
```

EXAMPLES:

DEFINE DECLARATION:

```
      DEFINE FORI= FOR I←1 STEP 1 UNTIL#,ADDUP = AxB+C/D#

DEFINITION LISTS:
---------- ------

      MOVER = ← #
      SPLIT = GO TO#,FOOL(GRANT,MAYBE) = IF GRANT THEN
      MAYBE#

DEFINITION:
-----------

      GRANTED (ARITHEXP) = ARITHEXP > NOTSO #

FORMAL SYMBOL LISTS:
------ ------ ------

      IDENTIFIERONE, TWO
      ONLY

TEXT:
-----

      (
      PROCEDURE
      ANYID
      IF A THEN GO TO SOUTH ELSE BEGIN X←ZxQ;GO TO NORTH END
      EG;

INVOCATION:
-----------

      GUARANTY ( X←Y+1 )

ACTUAL TEXT PART:
------ ---- -----

      (ERGO)
      [X←1;GO TO L;]
```

7.4. DIRECT ARRAY DECLARATIONS

SYNTAX:

<DIRECT ARRAY DECLARATION> ::= DIRECT <DIRECT ARRAY KIND>
ARRAY <ARRAY LIST>
<DIRECT ARRAY KIND> ::= <EMPTY> / <TYPE>

SEMANTICS:

AN ARRAY WHICH IS DECLARED "DIRECT" PROVIDES FOR FASTER INPUT/OUTPUT OPERATIONS. WHEN AN ARRAY IS DECLARED "DIRECT", THE ARRAY SPACE ITSELF CAN BE USED AS AN INPUT/OUTPUT BUFFER.

EXAMPLES:

```
DIRECT INTEGER ARRAY A[10], B[15,15]
DIRECT ARRAY A[4,10]
```

7.5. EVENT AND EVENT ARRAY DECLARATIONS

SYNTAX:

<EVENT DECLARATION> ::= <EVENT IDENTIFIER LIST>
<EVENT IDENTIFIER LIST> ::= <EVENT IDENTIFIER> <ADDRESS PART> /
    <EVENT LIST> , <EVENT IDENTIFIER> <ADDRESS PART>
<EVENT IDENTIFIER> ::= <IDENTIFIER>
<EVENT ARRAY DECLARATION> ::= EVENT ARRAY <EVENT SEGMENT LIST>
<EVENT SEGMENT LIST> ::= <EVENT SEGMENT> / <EVENT SEGMENT LIST> ,
    <EVENT SEGMENT>
<EVENT SEGMENT> ::= <EVENT ARRAY LIST> [<BOUND LIST>]
<EVENT ARRAY LIST> ::= <EVENT ARRAY IDENTIFIER> <ADDRESS PART> /
    <EVENT ARRAY LIST> , <EVENT ARRAY IDENTIFIER> <ADDRESS PART>
<EVENT ARRAY IDENTIFIER> ::= <IDENTIFIER>

SEMANTICS:

1. AN EVENT DECLARATION DEFINES THE IDENTIFIER OF A QUANTITY WHICH MAY BE USED TO RECORD AN OCCURRENCE.

2. AN EVENT ARRAY DECLARATION DEFINES THE IDENTIFIER OF AN ARRAY OF THESE QUANTITIES.

3. THE QUANTITIES ARE USED TO REPORT AN OCCURRENCE TO AN ASYNCHRONOUS PROCESS.

EXAMPLES:

EVENT DECLARATION:

```
EVENT E1,E2 = (3,4)
```

EVENT ARRAY DECLARATION:

```
EVENT ARRAY E3=(NONCE),E4[73],E5=(-2)[6]
```

7.6. FIELD DECLARATIONS

SYNTAX:

<FIELD DECLARATION> ::= FIELD <FIELD PART LIST>
<FIELD PART LIST> ::= <FIELD PART> / <FIELD PART LIST> ,
<FIELD PART>
<FIELD PART> ::= <FIELD IDENTIFIER> = <FIELD>
<FIELD IDENTIFIER> ::= <IDENTIFIER>
<FIELD> ::= <ARITHMETIC EXPRESSION> : <ARITHMETIC EXPRESSION>

SEMANTICS:

1. THE BASIC B6700/B7700 WORD CONSISTS OF 51 BITS:

   A. BITS 50, 49, AND 48 ARE REFERRED TO AS "TAG" BITS. THESE TAG BITS CANNOT BE ADDRESSED DIRECTLY; HOWEVER, THEY MAY BE ADDRESSED BY THE RESERVED FIELD IDENTIFIER "TAG".

   B. THE BALANCE OF THE B6700/B7700 WORD IS REFERRED TO AS THE "INFORMATION FIELD". THE INFORMATION FIELD CONSISTS OF BITS 47 THROUGH ZERO, WHERE 47 IS THE BIT ON THE FAR LEFT AND ZERO IS THE BIT ON THE FAR RIGHT.

2. "TAG" IS AN INTRINSIC FIELD; IT IS EQUIVALENT TO THE FIELD 50:3, EXCEPT THAT THE LATTER IS NOT EXPLICITLY PERMITTED.

3. A FIELD DECLARATION DEFINES EACH IDENTIFIER IN ITS FIELD PART LIST AS A FIELD IDENTIFIER AND SPECIFIES THE FIELD THAT IT IS IDENTIFYING.

EXAMPLES:

```
FIELD A1 = 3:1, AZ = B:1
FIELD QUIZ = 20:21
```

7.7. FILE DECLARATIONS

SYNTAX:

```
<FILE DECLARATION> ::= <DIRECT SPECIFIER> FILE <FILE LIST>
<FILE LIST> ::= <FILE LIST PART> / <FILE LIST>, <FILE LIST PART>
<FILE LIST PART> ::= <FILE DESIGNATOR> /
      <FILE DESIGNATOR> ( <INITIAL ATTRIBUTE LIST> )
<FILE DESIGNATOR> ::= <FILE IDENTIFIER> /
      <DIRECT FILE IDENTIFIER>
<FILE IDENTIFIER> ::= <IDENTIFIER>
<DIRECT FILE IDENTIFIER> ::= <IDENTIFIER>
<INITIAL ATTRIBUTE LIST> ::= <INITIAL ATTRIBUTE> /
      <INITIAL ATTRIBUTE LIST>, <INITIAL ATTRIBUTE>
<INITIAL ATTRIBUTE> ::=
      <FILE ATTRIBUTE NAME> = <FILE ATTRIBUTE VALUE>
<FILE ATTRIBUTE NAME> ::=
      <ARITHMETIC-VALUED FILE ATTRIBUTE NAME> /
      <BOOLEAN-VALUED FILE ATTRIBUTE NAME> /
      <MNEMONIC-VALUED FILE ATTRIBUTE NAME> /
      <POINTER-VALUED FILE ATTRIBUTE NAME>
<ARITHMETIC-VALUED FILE ATTRIBUTE NAME> ::= AREACLASS / AREAS /
      AREASIZE / ASSIGNTIME / ATTVALUE / ATTYPE / BLOCK /
      BLOCKSIZE / BUFFERS / CENSUS / COPIES / CYCLE / DATE /
      DIRECTION / DISPOSITION / ENABLEINPUT / ERRORTYPE /
      FAMILYSIZE / FILETYPE / LASTRECORD /
      LASTSTATION / LINENUM / MAXGENNO / MAXRECSIZE /
      MINRECSIZE / NEXTRECORD / PAGE / PAGESIZE /
      POPULATION / RECEPTIONS / RECORD / RECORDSZ / REEL /
      ROWADDRESS / ROWINUSE / SAVEFACTOR /
      SECURITYTYPE / SECURITYUSE / SCREEN / SERIALNO /
      SIZEMODE / SIZEOFFSET / SIZE2 / SPEED / STATE /
      TAPEREELRECORD / TRANSMISSIONO / TRANSMISSIONS / UNITNO /
      UNITS / UNITSLEFT / USEDATE / VERSION / WIDTH
```

```
<BOOLEAN-VALUED FILE ATTRIBUTE NAME> ::= ATTERR / BEGINAT1 /
      BREAK / CODEFILE / COPIES / CYLINDERMODE / DUPLICATED /
      EOF / INTERCHANGE / NULINPUT / OPEN / OPTIONAL / PRESENT /
      READCHECK / RESIDENT / SINGLEPACK / UPDATE
<MNEMONIC-VALUED FILE ATTRIBUTE NAME> ::= DENSITY / EXTMODE /
      FILEKIND / INTMODE / KIND / LABELTYPE / MYUSE /
      OTHERUSE / PARITY / PROTECTION / SECURITYTYPE /
      SECURITYUSE / SIZEMODE / SPEED / UNITS
<POINTER-VALUED FILE ATTRIBUTE NAME> ::= FAMILY / FORMESSAGE /
      INTNAME / TITLE
<FILE ATTRIBUTE VALUE> ::= <CONSTANT> /
      <FILE ATTRIBUTE MNEMONIC VALUE>
<FILE ATTRIBUTE MNEMONIC VALUE> ::= <DENSITY MNEMONIC> /
      <ERRORTYPE MNEMONIC> / <EXTMODE MNEMONIC> /
      <FILEKIND MNEMONIC> / <INTMODE MNEMONIC> / <KIND MNEMONIC> /
      <LABELTYPE MNEMONIC> / <MYUSE MNEMONIC> /
      <OTHERUSE MNEMONIC> / <PARITY MNEMONIC> /
      <PROTECTION MNEMONIC> / <SECURITYTYPE MNEMONIC> /
      <SECURITYUSE MNEMONIC> / <SIZEMODE MNEMONIC> /
      <SPEED MNEMONIC> / <STATE MNEMONIC> /
      <UNITS MNEMONIC>
<DENSITY MNEMONIC> ::= HIGH / MEDIUM / LOW /
      SUPER
<ERRORTYPE MNEMONIC> ::= NOERROR / SUNOTREADY /
      READPARITYERROR / READCHECKFAILURE /
<EXTMODE MNEMONIC> ::= SINGLE / BCL / EBCDIC
<FILEKIND MNEMONIC> ::= SYSTEMDIRECTORY / VERSIONDIRECTORY /
      DIRECTORY / CONTROLDECK / BACKUPDISK  /
      RECONSTRUCTIONFILE / SYSTEMDIRFILE /
      COMPILERCODE / LIBRARYCODE / INTRINSICFILE /
      MCPCODEFILE / ALGOLCODE / COBOLCODE / FORTRANCODE /
      XALGOLCODE / PLICODE / JOVIALCODE / ESPOLCODE /
      DCALGOLCODE / BASICCODE / XFORTRANCODE / BOUNDCODE /
      CODEFILE / ALGOLSYMBOL / COBOLSYMBOL / FORTRANSYMBOL /
      XALGOLSYMBOL / PLISYMBOL / JOVIALSYMBOL / ESPOLSYMBOL /
      DCALGOLSYMBOL / BASICSYMBOL / XFORTRANSYMBOL / DATA
```

```
<INTMODE MNEMONIC> ::= SINGLE / BCL / EBCDIC / ASCII
<KIND MNEMONIC> ::= DISK / DISKPACK / DISPLAY / PAPER /
      PAPERPUNCH / PAPERREADER / PETAPE / PRINTER / PUNCH /
      READER / REMOTE / TAPE / TAPE7 / TAPE9
<LABELTYPE MNEMONIC> ::= STANDARD / OMITTED / OMITTEDEOF
<MYUSE MNEMONIC> ::= CLOSED / IN / OUT / IO
<OTHERUSE MNEMONIC> ::= SECURED / IN / OUT / IO
<PARITY MNEMONIC> ::= STANDARD / NONSTANDARD
<PROTECTION MNEMONIC> ::= TEMPORARY / SAVE / PROTECTED
<SECURITYTYPE MNEMONIC> ::= PRIVATE / CLASSA / CLASSB /
      CLASSC
<SECURITYUSE MNEMONIC> ::= SECURED / IN / OUT / IO
<SIZEMODE MNEMONIC> ::= SINGLE / HEX / BCL / EBCDIC
<SPEED MNEMONIC> ::= FAST / MEDIUMFAST / MEDIUMSLOW /
      SLOW
<STATE MNEMONIC> ::= NORMAL / ATEND / PARITYERROR /
      DATA ERROR / LOCKEDOUT / NOINPUT / BREAKHERE /
      TIMEOUT / NEWUSER
<UNITS MNEMONIC> ::= WORDS / CHARACTERS
```

SEMANTICS:

<FILE DECLARATION>S MAY NOT BE DECLARED IN THE OUTER BLOCK, NOR MAY THEY BE USED IN ESPOL INTRINSICS.

A <FILE DECLARATION> ASSOCIATES AN <IDENTIFIER> WITH A FILE. THAT FILE"S ATTRIBUTES NEED NOT ALL BE SPECIFIED IN THE <FILE DECLARATION>. THOSE ATTRIBUTES NOT SPECIFIED IN THE <FILE DECLARATION> MUST BE SET BY CALLING THE MCP PROCEDURE "ATTRIBUTEHANDLER" USING THE APPROPRIATE PARAMETER(S).

A <BOOLEAN-VALUED FILE ATTRIBUTE NAME> APPEARING WITHOUT THE "= <LOGICAL VALUE>" PART IMPLIES "=TRUE".

REFER TO THE B6700 I/O SUBSYSTEM INFORMATION MANUAL (5000185) FOR DESCRIPTIONS AND EXPLANATIONS OF THE FILE ATTRIBUTES.

EXAMPLES:

```
FILE LINE (KIND=PRINTER, BUFFERS=1, MAXRECSIZE=17);
FILE FID (KIND=TAPE9, TITLE="CLUES", DENSITY=HIGH),
     FYLE (KIND=DISK);
```

7.8. INTERRUPT DECLARATIONS

SYNTAX:

<INTERRUPT DECLARATION> ::= INTERRUPT <INTERRUPT SEGMENT>
<INTERRUPT SEGMENT> ::= <INTERRUPT IDENTIFIER>; <STATEMENT>
<INTERRUPT IDENTIFIER> ::= <IDENTIFIER>

SEMANTICS:

1. INTERRUPT DECLARATIONS PROVIDE A MEANS OF FORCING A PROCESS TO DEPART FROM ITS CURRENT POINT OF CONTROL AND EXECUTE THE STATEMENT ASSOCIATED WITH THE INTERRUPT DECLARATION.

2. IF THE PROCESS IS INACTIVE WHEN THE EVENT IS CAUSED, MORE THAN ONE INTERRUPT STATEMENT MAY BE PENDING WHEN THE PROCESS IS REACTIVATED. IN THIS CASE, THE INTERRUPT STATEMENTS ARE PROCESSED IN THE BLOCK ORDER, WITH THE OUTERMOST BLOCKS FIRST, BEFORE RETURN IS MADE TO THE REACTIVATION POINT.

3. AN INTERRUPT MUST BE ENABLED AND ATTACHED TO AN EVENT BEFORE IT CAN HAVE ANY EFFECT.

   AN INTERRUPT IS ATTACHED TO AN EVENT BY USING AN <ATTACH STATEMENT>. THIS STATEMENT IMPLICITLY CAUSES THE INTERRUPT (BEING ATTACHED) TO BE ENABLED. THE <DETACH STATEMENT> SEVERS THE ASSOCIATION BETWEEN THE INTERRUPT AND THE EVENT. IT ALSO IMPLICITLY CAUSES THE INTERRUPT TO BE DISABLED.

   IF AN INTERRUPT IS ALREADY ATTACHED TO AN EVENT, THE INTERRUPT MAY BE ENABLED OR DISABLED BY AN <ENABLE STATEMENT> OR A <DISABLE STATEMENT>. USING ONE OF THESE TWO STATEMENTS DOES NOT AFFECT THE ASSOCIATION BETWEEN AN INTERRUPT AND THE EVENT ATTACHED TO IT.

4. WITHIN ANY BLOCK, TWO INTERRUPT DECLARATIONS MAY REFERENCE THE SAME EVENT.

EXAMPLES:

```
INTERRUPT I1;  A := A+B;
INTERRUPT I2;  GO TO LBL;
```

7.9. LABEL DECLARATIONS

SYNTAX:

```
<LABEL DECLARATION> ::= LABEL <LABEL LIST>
<LABEL LIST> ::= <LABEL IDENTIFIER> / <LABEL LIST> ,
      <LABEL IDENTIFIER>
<LABEL IDENTIFIER> ::= <IDENTIFIER>
```

SEMANTICS:

1. A DECLARED LABEL DECLARATION DEFINES EACH IDENTIFIER IN ITS LABEL LIST AS A LABEL IDENTIFIER.

2. A LABEL IDENTIFIER MUST APPEAR IN A LABEL DECLARATION IN THE HEAD OF THE BLOCK IN WHICH IT IS USED TO LABEL A STATEMENT.

3. A LABEL IDENTIFIER NEED NOT BE DECLARED IF ITS FIRST APPEARANCE IN THE BLOCK IS IN A GO TO STATEMENT AND NO NONLOCAL LABEL WITH THE SAME IDENTIFIER HAS PREVIOUSLY APPEARED.

EXAMPLES:

LABEL DECLARATIONS:

```
LABEL FOG;
LABEL L7,L8;
```

LABEL LISTS:

```
START,EXIT,LOOP
NEXT
```

7.10. LAYOUT DECLARATIONS

SYNTAX:

```
<LAYOUT DECLARATION> ::= LAYOUT <LAYOUT PART LIST>
<LAYOUT PART LIST> ::= <LAYOUT PART> / <LAYOUT PART LIST> ,
      <LAYOUT PART>
<LAYOUT PART> ::= <LAYOUT IDENTIFIER> (<LAYOUT ITEM LIST>)
<LAYOUT IDENTIFIER> ::= <IDENTIFIER>
<LAYOUT ITEM LIST> ::= <LAYOUT ITEM> / <LAYOUT ITEM LIST> ,
      <LAYOUT ITEM>
<LAYOUT ITEM> ::= <LAYOUT FIELD> <FIELD VALUE PART>
<LAYOUT FIELD> ::= <FIELD PART> / <FIELD> / <FIELD IDENTIFIER>
<FIELD VALUE PART> ::= <EMPTY> /
      <REPLACEMENT OPERATOR> <UNSIGNED INTEGER>
```

SEMANTICS:

1. A LAYOUT DECLARATION IDENTIFIES EACH IDENTIFIER IN ITS LAYOUT PART LIST AS A LAYOUT IDENTIFIER AND SPECIFIES A LAYOUT ITEM LIST.

2. A LAYOUT ITEM LIST IS COMPOSED OF ONE OR MORE FIELDS, REFERRED TO AS LAYOUT ITEMS.

3. A LAYOUT ITEM MAY BE ANY OF THE FOLLOWING:

   A. A PREVIOUSLY DECLARED FIELD IDENTIFIER.

   B. AN UNIDENTIFIED FIELD.

   C. A FIELD PART.

4. A LAYOUT ITEM MAY SPECIFY A DEFAULT VALUE FOR THE FIELD. IF NO DEFAULT VALUE IS GIVEN AND NO VALUE IS ASSIGNED, THE FIELD

IS IGNORED.

EXAMPLES:

```
LAYOUT C2S (7:6 ← 5, QA = B:6 ← 92, QUIZ), C3S (4:2 ← 3)
LAYOUT LOOK (42:6 ← 9, QA = BxQ-R:2)
B1 = B:6
```

7.11. MONITOR DECLARATIONS

SYNTAX:

```
<MONITOR DECLARATION> ::= MONITOR <PROCEDURE IDENTIFIER>
      ( <MONITOR LIST> )
<MONITOR LIST> ::= <MONITORED ITEM> /
      <MONITOR LIST> , <MONITORED ITEM>
<MONITORED ITEM> ::= <SIMPLE VARIABLE> / <ARRAY IDENTIFIER>
```

SEMANTICS:

WITHIN THE SCOPE OF A MONITOR DECLARATION, THE PROCEDURE IDENTIFIED IN THE MONITOR DECLARATION WILL BE EXECUTED WHENEVER A VALUE IS TO BE ASSIGNED --BY MEANS OF THE ASSIGNMENT STATEMENT-- TO A MONITORED ITEM. THE PROCEDURE MUST HAVE THE SAME TYPE AS THE ITEM AND MUST RETURN THE VALUE TO BE ASSIGNED TO THE ITEM. THE PROCEDURE MAY ONLY HAVE TWO PARAMETERS. THE PARAMETERS MUST BE SPECIFIED AS VALUE PARAMETERS. THE FIRST PARAMETER CORRESPONDS TO THE FIRST EIGHT CHARACTERS OF THE IDENTIFIER OF THE MONITORED ITEM. THE SECOND PARAMETER CORRESPONDS TO THE VALUE OF THE EXPRESSION.

7.12. PICTURE DECLARATIONS

SYNTAX:

```
<PICTURE DECLARATION> ::= <SAVE OR OWN> PICTURE <PICTURE PART LIST>
<SAVE OR OWN> ::= SAVE / OWN
<PICTURE PART LIST> ::= <PICTURE PART> / <PICTURE PART LIST> ,
      <PICTURE PART>
<PICTURE PART> ::= <PICTURE IDENTIFIER> ( <PICTURE> )
<PICTURE IDENTIFIER> ::= <IDENTIFIER>
<PICTURE> ::= <PICTURE SYMBOL> / <PICTURE> <PICTURE SYMBOL>
<PICTURE SYMBOL> ::= "<HEXADECIMAL STRING>" / "<BCL STRING>" /
      "<ASCII STRING>" / "<EBCDIC STRING>" /
      <PICTURE CHARACTER> <REPEAT PART> /
      <CONTROL CHARACTER> / <INTRODUCTION> /
      <SKIP CHARACTER> <REPEAT PART> /
      <SINGLE PICTURE CHARACTER>
<REPEAT PART> ::= <EMPTY> / ( <UNSIGNED INTEGER> ) / (*)
<CONTROL CHARACTER> ::= 4 / 6 / 7 / 8 / :
<SKIP CHARACTER> ::= > / <
<INTRODUCTION> ::= <INTRODUCTION CODE> <NEW CHARACTER>
<INTRODUCTION CODE> ::= B / C / M / N / P / U
<NEW CHARACTER> ::= <STRING CHARACTER> / "
<SINGLE PICTURE CHARACTER> ::= J / S
<PICTURE CHARACTER> ::= A / D / E / F / I / Q / R / X / Z / 9
```

SEMANTICS:

1. THE PICTURE DECLARATION PROVIDES A CONSTRUCT FOR GENERALIZED CHARACTER EDITING. THE FOLLOWING EDITING OPERATIONS MAY BE PERFORMED:

   A. UNCONDITIONAL CHARACTER MOVES.

   B. MOVE CHARACTERS WITH LEADING ZERO EDITING.

C. MOVE CHARACTERS WITH LEADING ZERO EDITING AND FLOATING CHARACTER INSERTION.

D. MOVE CHARACTERS WITH CONDITIONAL CHARACTER INSERTION.

E. MOVE NUMBERIC PART OF CHARACTERS ONLY.

F. MOVE CHARACTERS WITH UNCONDITIONAL CHARACTER INSERTION.

G. SKIP SOURCE CHARACTERS (FORWARD AND REVERSE).

H. INSERT OVERPUNCH SIGN ON THE PREVIOUS CHARACTER.

2. A PICTURE CONSISTS OF A NAMED STRING OF EDITING SYMBOLS WHICH ARE ENCLOSED IN PARENTHESES. THE PICTURE EDITING SYMBOLS LISTED BELOW MAY BE COMBINED IN ANY ORDER TO PERFORM A WIDE RANGE OF EDITING FUNCTIONS. "OWN" PICTURES PRODUCE IN-LINE CODE.

3. IF THE REPEAT PART IS EMPTY, IT IS ASSUMED TO BE EQUAL TO ONE. IF THE REPEAT PART IS OF THE FORM "(*)", AN UNSIGNED INTEGER VALUE IS EXPECTED FROM THE <REPEAT PARAMETERS> IN A STRING TRANSFER STATEMENT.

4. THE FOLLOWING OUTPUT CHARACTERS ARE ASSUMED FOR THE INTRODUCTION CODES. ANOTHER CHARACTER MAY BE SUBSTITUTED FOR THE ASSUMED CHARACTER BY THE USE OF THE INTRODUCTION PHRASE, AS DEFINED IN THE SYNTAX.

OUTPUT CHARACTER	INTRODUCTION CODE	NORMAL USE
SPACE(BLANK)	B	REPLACEMENT OF LEADING ZEROS.
,	C	CONDITIONAL INSERT CHARACTER.
-	M	CHARACTER INSERTION IF MINUS.

.	N	UNCONDITIONAL INSERT CHARACTER.
+	P	CHARACTER INSERTION IF PLUS.
$	U	FLOATING CHARACTER INSERTION.

5. THE CONTROL CHARACTERS SHOWN BELOW CAUSE THE FOLLOWING ACTION:

   4 SET THE DEFAULT CHARACTER SIZE OF INSERTED CHARACTERS AND STRINGS TO FOUR BITS.

   6 SET THE DEFAULT CHARACTER SIZE OF INSERTED CHARACTERS AND STRINGS TO SIX BITS.

   7 SET THE DEFAULT CHARACTER SIZE OF INSERTED CHARACTERS AND STRINGS TO SEVEN BITS.

   8 SET THE DEFAULT CHARACTER SIZE OF INSERTED CHARACTERS AND STRINGS TO EIGHT BITS.

   : REINITIATES LEADING ZERO REPLACEMENT.

6. QUOTED STRINGS ARE INSERTED UNCONDITIONALLY IN THE DESTINATION STRING. THE CONTROL CHARACTERS 4,6,7,AND 8 TELL THE COMPILER WHICH CHARACTER SET TO EXPECT. BCL IS THE DEFAULT CHARACTER SET.

7. THE SINGLE PICTURE CHARACTERS PERFORM THE FOLLOWING ACTION:

   J IF A MOVE WITH FLOAT (E OR F) HAS NOT INSERTED A FLOAT CHARACTER, TERMINATE THE FLOAT AND INSERT THE "U" CHARACTER. OTHERWISE, PERFORM NO OPERATION.

   S INSERT A SINGLE "P" CHARACTER IF THE SIGN IS PLUS. OTHERWISE, INSERT A SINGLE "M" CHARACTER.

8. THE PICTURE CHARACTERS LISTED BELOW PERFORM THE FOLLOWING ACTION:

A MOVE THE NUMBER OF CHARACTERS SPECIFIED BY THE REPEAT FIELD.

D IF AN E OR F FLOAT HAS NOT ENDED, INSERT THE "B" CHARACTER. OTHERWISE, INSERT THE "C" CHARACTER.

E MOVE THE NUMERIC PART ONLY FOR THE NUMBER OF CHARACTERS SPECIFIED BY THE REPEAT FIELD. SUPPRESS LEADING ZEROS BY SUBSTITUTING THE "B" CHARACTER. IF THE SIGN IS PLUS, INSERT A "P" CHARACTER IN FRONT OF THE FIRST NONZERO NUMBER. OTHERWISE, INSERT AN "M" CHARACTER. END THE FLOAT ACTION.

F PERFORM A MOVE NUMERIC WITH LEADING ZEROS REPLACED BY THE "B" CHARACTER. INSERT A "U" CHARACTER IN FRONT OF THE FIRST NONZERO NUMBER. END THE FLOAT ACTION.

I INSERT THE "N" CHARACTER UNCONDITIONALLY.

Q BACK UP THE NUMBER OF CHARACTERS INDICATED BY THE <REPEAT PART> AND INSERT A SIGN OVERPUNCH.

R IF AN E OR F FLOAT HAS NOT ENDED, INSERT THE "P" CHARACTER. OTHERWISE, INSERT THE "M" CHARACTER.

X SKIP THE DESTINATION POINTER FORWARD BY THE NUMBER OF CHARACTERS SPECIFIED IN THE REPEAT FIELD.

Z PERFORM A MOVE NUMERIC WITH LEADING ZEROS REPLACED BY BLANKS.

9 MOVE THE NUMERIC PART ONLY OF THE NUMBER OF CHARACTERS SPECIFIED BY THE REPEAT FIELD.

9. THE PICTURE SKIP CHARACTERS PERFORM THE FOLLOWING ACTION:

< SKIP THE SOURCE POINTER BACKWARDS BY THE NUMBER OF CHARACTERS SPECIFIED IN THE REPEAT FIELD.

> SKIP THE SOURCE POINTER FORWARD BY THE NUMBER OF CHARACTERS SPECIFIED IN THE REPEAT FIELD.

7.13. PROCEDURE DECLARATIONS

SYNTAX:

```
<PROCEDURE DECLARATION> ::= <SAVE PART> <PROCEDURE TYPE> PROCEDURE
      <PROCEDURE HEADING> <PROCEDURE BODY>
<SAVE PART> ::= <EMPTY> / SAVE / SAVE 1
<PROCEDURE TYPE> ::= <EMPTY> / <TYPE>
<PROCEDURE HEADING> ::= <PROCEDURE IDENTIFIER> <ADDRESS PART>
      <FORMAL PARAMETER PART>;
<PROCEDURE IDENTIFIER> ::= <IDENTIFIER>
<FORMAL PARAMETER PART> ::= <EMPTY> / (<FORMAL PARAMETER LIST>) ;
      <VALUE PART> <SPECIFICATION PART>
<FORMAL PARAMETER LIST> ::= <FORMAL PARAMETER>/
      <FORMAL PARAMETER LIST> <PARAMETER DELIMITER>
      <FORMAL PARAMETER>
<FORMAL PARAMETER> ::= <IDENTIFIER>
<VALUE PART> ::= <EMPTY> / VALUE <IDENTIFIER LIST> ;
<SPECIFICATION PART> ::= <SPECIFICATION> / <SPECIFICATION PART> ;
      <SPECIFICATION>
<SPECIFICATION> ::= <SPECIFIER> <IDENTIFIER LIST> /
      <ARRAY SPECIFICATION>
<SPECIFIER> ::= <TYPE> / <PROCEDURE TYPE> PROCEDURE /
      QUEUE / EVENT / PICTURE
<IDENTIFIER LIST> ::= <IDENTIFIER> / <IDENTIFIER LIST> ,
      <IDENTIFIER>
<ARRAY SPECIFICATION> ::= <DIRECT SPECIFIER> <ARRAY TYPE> ARRAY
      <ARRAY SPECIFIER LIST>
<DIRECT SPECIFIER> ::= <EMPTY> / DIRECT
<ARRAY TYPE> ::= <EMPTY> / <TYPE> / EVENT
<ARRAY SPECIFIER LIST> ::= <ARRAY SPECIFIER> /
      <ARRAY SPECIFIER LIST>, <ARRAY SPECIFIER>
<ARRAY SPECIFIER> ::= <ARRAY IDENTIFIER LIST> [ <BOUND SPECIFIER> ]
<ARRAY IDENTIFIER LIST> ::= <ARRAY IDENTIFIER> /
      <ARRAY IDENTIFIER LIST> , <ARRAY IDENTIFIER>
```

<BOUND SPECIFIER> ::= * / <BOUND SPECIFIER> , *

<PROCEDURE BODY> ::= <STATEMENT> / FORWARD / EXTERNAL / NULL

SEMANTICS:

1. A PROCFDURE DECLARATION DEFINES THE PROCEDURE IDENTIFIER AS THE NAME OF A PROCEDURE.

2. A <SAVE PART> OF "SAVE" INDICATES THAT THE CODE FOR THE PROCEDURE CURRENTLY BEING DECLARED IS TO BE IN THE SAME SEGMENT AS THE BLOCK IN WHICH IT IS DECLARED. THUS THE COMPILER WILL NOT CREATE A NEW SEGMENT FOR THE PROCEDURE. "SAVE" ALSO SPECIFIES CONTROL STATE.

3. A <SAVE PART> OF "SAVE 1" INDICATES THAT THE PROCEDURE BEING DECLARED IS AN "INITIALIZATION" PROCEDURE. THE CODE FOR THE PROCEDURE WILL BE AT THE END OF THE INITIAL BLOCK OF INFORMATION WHICH IS LOADED BY THE HARDWARE. THERE WILL BE THREE WORDS OF INFORMATION BETWEEN THE "SAVE 1 CODE" AND THE REST OF THE INITIAL BLOCK, IN ORDER TO FACILITATE RELOCATION OF THE AREA AFTER INITIALIZATION IS COMPLETE.

4. THE DIRECTIONS GIVEN FOR THE TYPE DECLARATION ADDRESS PART APPLY TO PROCEDURE DECLARATIONS. THE ADDRESS SPECIFIED MUST BE THAT OF A PROGRAM CONTROL WORD OR AN INDIRECT REFERENCE WORD POINTING TO A PROGRAM CONTROL WORD.

5. THE VALUE PART SPECIFIES WHICH FORMAL PARAMETERS ARE TO BE CALLED BY VALUE. WHEN A FORMAL PARAMETER IS CALLED BY VALUE, THE FORMAL PARAMETER IS SET TO THE VALUE OF THE CORRESPONDING ACTUAL PARAMETER. THEREAFTER, THE FORMAL PARAMETER IS HANDLED AS A VARIABLE THAT IS LOCAL TO THE PROCEDURE BODY. THAT IS, ANY CHANGE OF VALUE OF THE VARIABLE WILL NOT RAMIFY OUTSIDE THE PROCEDURE BODY.

6. ONLY ARITHMETIC, BOOLEAN, POINTER, AND REFERENCE EXPRESSIONS

MAY BE GIVEN AS ACTUAL PARAMETERS TO BE CALLED BY VALUE. THESE EXPRESSIONS WILL BE EVALUATED ONCE, BEFORE ENTRY INTO THE PROCEDURE BODY.

7. FORMAL PARAMETERS NOT IN THE VALUE PART ARE CALLED BY NAME (AN EXCEPTION IS THE EVENT FORMAL PARAMETER; SEE BELOW). THIS MEANS THAT WHEREVER A FORMAL PARAMETER -- CALLED BY NAME -- APPEARS IN THE PROCEDURE BODY, THE FORMAL PARAMETER IS REPLACED BY THE ACTUAL PARAMETER.

   EVENT FORMAL PARAMETERS ARE CALLED BY REFERENCE (NOT IN ANY WAY CONNECTED WITH THE ESPOL REFERENCE TYPE). CALL BY REFERENCE DIFFERS FROM CALL BY NAME IN THAT WHEN THE ACTUAL PARAMETER IS A SUBSCRIPTED VARIABLE, THE REFERENCED ARRAY ELEMENT IS DETERMINED AT THE TIME OF CALL. IT IS THIS ARRAY ELEMENT WHICH IS ACCESSED AT EACH APPEARANCE OF THE FORMAL PARAMETER WITHIN THE PROCEDURE BODY.

8. EVERY FORMAL PARAMETER MUST APPEAR IN THE SPECIFICATION PART.

9. AN "*" MUST APPEAR IN A BOUND SPECIFIER FOR EACH DIMENSION OF THE ARRAY.

10. ESPOL PROCEDURES ARE RECURSIVE.

11. IN CERTAIN SITUATIONS WHERE PROCEDURES ARE CALLED RECURSIVELY, IT IS NECESSARY TO CALL A PROCEDURE THAT HAS NOT BEEN DECLARED THE DECLARATOR "FORWARD" IS USED IN THIS CIRCUMSTANCE. THAT IS, THERE IS FIRST AN APPROPRIATE PROCEDURE DECLARATION, WITH THE PROCEDURE BODY REPLACED BY "FORWARD", THEN THE CALL ON THE PROCEDURE, AND LATER, THE COMPLETE PROCEDURE DECLARATION.

12. A <PROCEDURE BODY> OF "EXTERNAL" INDICATES THAT THE PROCEDURE WILL BE BOUND IN LATER AND NOT COMPILED IN NOW.

13. IF "NULL" IS SPECIFIED AS THE <PROCEDURE BODY>, THE COMPILER

GENERATES NOTHING FURTHER. THIS IS TYPICALLY USED WHEN THE PROGRAMMER KNOWS MORE THAN THE COMPILER.

EXAMPLES:

PROCEDURE DECLARATIONS:

```
PROCEDURE FAKE;X←X+Z
PROCEDURE NEXT(A); VALUE A; REAL A; X←A+1
PROCEDURE NEXT(A); VALUE A; REAL A;  FORWARD
```

PROCEDURE HEADING:

```
COMMUNE = INTRIN (A); REAL A;
```

FORMAL PARAMETER PARTS:

```
(A,B,C); VALUE A; REAL A,B,C;
(ABSTRACT,DEGENERATE); VALUE ABSTRACT; REAL ABSTRACT;
PROCEDURE DEGENERATE;
```

FORMAL PARAMETER LISTS:

```
A,B,C
X) "VALUE OF EXPRESSION X PLUS 2 "(CALCRULE
```

VALUE PART:

```
VALUE X,Y,Z
```

SPECIFICATION PART:

```
INTEGER N; ARRAY A,B,C,X1,X2[*];ALPHA ARRAY X3[*];
```

SPECIFICATIONS:

```
INTEGER N,O,P,Q
PROCEDURE MIN, MAX, FIX
QUEUE X,Y,Z
EVENT A,B,C
QUEUE ARRAY FRED[*]
```

ARRAY SPECIFICATIONS:

```
REAL ARRAY GYM [*, *]
BOOLEAN ARRAY WADSUP [*,*,*], CUMON[*,*]
```

ARRAY SPECIFIER LISTS:

```
GIGO[*]
HERE[*],THERE[*,*,*,*]
```

ARRAY SPECIFIER:

```
X1, X2, X3 [*]
```

PROCEDURE BODIES:

```
BEGIN I←x+QxR;V←I+RAYG(Q);END SAMPLE
FORWARD
```

7.14. QUEUE AND QUEUE ARRAY DECLARATIONS

SYNTAX:

```
<QUEUE DECLARATION> ::= QUEUE <QUEUE HEAD> <QUEUE BODY>
<QUEUE ARRAY DECLARATION> ::= QUEUE ARRAY <QUEUE ARRAY HEAD>
      [<INDEX BOUND>] <QUEUE BODY>
<QUEUE HEAD> ::= <QUEUE IDENTIFIER> <INTERNAL NAME PART>
<QUEUE ARRAY HEAD> ::= <QUEUE ARRAY IDENTIFIER>
      <INTERNAL NAME PART>
<INDEX BOUND> ::= <ARITHMETIC EXPRESSION>
<INTERNAL NAME PART> ::= <EMPTY> / : <SECOND NAME> <ADDRESS PART>
<SECOND NAME> ::= <IDENTIFIER>
<QUEUE BODY> ::= <ENTRY DESCRIPTION> ; <ALGORITHM PART>
<ENTRY DESCRIPTION> ::= (<ENTRY ITEM LIST>) ; <VALUE PART> ;
      <SPECIFICATION PART>
<ENTRY ITEM LIST> ::= <ITEM LIST> <INVISIBLE ITEM LIST>
<ITEM LIST> ::= <ITEM IDENTIFIER> /
      <ITEM LIST> <PARAMETER DELIMITER> <ITEM IDENTIFIER>
<INVISIBLE ITEM LIST> ::= <ITEM LIST>
<ALGORITHM PART> ::= <EMPTY> / USING <ALGORITHM LIST>
<ALGORITHM LIST> ::= <ALGORITHM> / <ALGORITHM> : <ALGORITHM LIST>
<ALGORITHM> ::= <BOOLEAN ALGORITHM IDENTIFIER> IF
      <BOOLEAN EXPRESSION> / <REFERENCE ALGORITHM IDENTIFIER> IS
      <REFERENCE EXPRESSION> / TO <ALGORITHM IDENTIFIER> ,
      <STATEMENT> / <LOCK SPECIFICATION> /
      <INTEGER ALGORITHM IDENTIFIER>, <STATEMENT>
<LOCK SPECIFICATION> ::= LOCKED / LOCKED <QUEUE NAME>
<QUEUE IDENTIFIER> ::= <IDENTIFIER>
<QUEUE ARRAY IDENTIFIER> ::= <IDENTIFIER>
<REFERENCE ALGORITHM IDENTIFIER> ::= ALLOCATE / NEXT / LAST /
      FIRST / PRIOR
<ALGORITHM IDENTIFIER> ::= INSERT / REMOVE / DELINK / <IDENTIFIER>
<BOOLEAN ALGORITHM IDENTIFIER> ::= EMPTY / FULL
<INTEGER ALGORITHM IDENTIFIER> ::= POPULATION
```

SEMANTICS:

1. A QUEUE IS AN ORDERED LIST OF ENTRIES. EACH ENTRY HAS THE FORM OF AN <ARRAY ROW>, WITH ONE WORD FOR EACH <ITEM IDENTIFIER> IN THE <ENTRY ITEM LIST>.

2. A QUEUE ARRAY IS AN ARRAY OF QUEUES. THE <INDEX BOUND> DESIGNATES THE NUMBER OF QUEUES INVOLVED AND IS A STRICT UPPER BOUND FOR THE QUEUE ARRAY SUBSCRIPTS.

3. THE <ENTRY ITEM LIST> IS IN EFFECT A DECLARATION FOR EACH OF THE <ITEM IDENTIFIER>S APPEARING IN IT. THOSE <ITEM IDENTIFIER>S APPEARING IN THE <INVISIBLE ITEM LIST> ARE UNDERSTOOD TO BE LOCAL TO THE QUEUE DECLARATION AND MAY NOT BE REFERENCED EXCEPT IN THE <QUEUE BODY>. THE REMAINING <ITEM IDENTIFIER>S WILL BE LOCAL TO THE BLOCK IN WHICH THE QUEUE IS DECLARED.

4. WHEN AN <ENTRY EXPRESSION> IS CREATED, THE ELEMENTS IN THE <ACTUAL ITEM LIST> MUST BE IN A ONE-TO-ONE-CORRESPONDENCE WITH THE <ITEM IDENTIFIER>S IN THE <ITEM LIST> OF THE <ENTRY DESCRIPTION> FOR THE QUEUE IN QUESTION. THE INVISIBLE <ITEM IDENTIFIER>S WILL NOT HAVE CORRESPONDING <ACTUAL PARAMETER>S.

5. THE <SECOND NAME> OF THE QUEUE OR QUEUE ARRAY IS LOCAL TO THE <QUEUE BODY>. IT IS USED WITHIN THE <QUEUE BODY> WHENEVER IT IS NECESSARY TO REFER TO THE QUEUE ITSELF IN A SITUATION WHICH WOULD OTHERWISE LEAD TO A RECURSIVE CALL ON THE QUEUE ALGORITHMS.

6. THE RESERVED WORD "ENTRY" IS LOCAL TO THE <QUEUE BODY>. IT IS USED TO REFER TO THE ENTRY WHICH IS CURRENTLY BEING PLACED IN OR REMOVED FROM THE QUEUE. FOR <QUEUE ARRAY DECLARATION>S THERE IS ALSO THE RESERVED WORD "INDEX" WHICH IS LOCAL TO THE <QUEUE BODY>. "INDEX" IS THE CURRENT SUBSCRIPT TO THE QUEUE

ARRAY.

7. THE "ALLOCATE" ALGORITHM WILL BE INVOKED WHENEVER IT IS NECESSARY TO ALLOCATE SPACE FOR AN ENTRY OR AN <ENTRY EXPRESSION>. IF NO "ALLOCATE" ALGORITHM IS GIVEN, THEN AN <ENTRY EXPRESSION> USING THE <QUEUE NAME> CANNOT BE USED.

8. THE "INSERT" ALGORITHM IS USED TO INSERT NEW ENTRIES IN THE QUEUE. IT WILL BE CALLED EACH TIME A <QUEUE ASSIGNMENT> IS USED. THUS A <QUEUE ASSIGNMENT> MAY NOT BE USED FOR A QUEUE LACKING A "INSERT" ALGORITHM.

9. THE QUEUE ALGORITHMS USED BY A QUEUE ARE LOCAL TO THE BLOCK IN WHICH THE QUEUE IS DECLARED. IN AN EXPLICIT CALL ON A QUEUE ALGORITHM, THE <ACTUAL PARAMETER LIST> WILL HAVE THE FOLLOWING GENERAL FORMAT: <QUEUE NAME>, <REFERENCE EXPRESSION>, <ARITHMETIC EXPRESSION>. THE <QUEUE NAME> PARAMETER MUST ALWAYS BE PRESENT. THE <REFERENCE EXPRESSION> IS NOT REQUIRED FOR ALL ALGORITHMS; WHEN USED, IT IS PASSED TO "ENTRY" IN THE <QUEUE BODY>. THE <ARITHMETIC EXPRESSION> IS REQUIRED ONLY FOR QUEUE ARRAYS, AND EVEN THEN IT MAY NOT BE NECESSARY. THE <ARITHMETIC EXPRESSION> IS PASSED TO "INDEX" IN THE <QUEUE BODY> OF THE <QUEUE ARRAY DECLARATION>. THE FOLLOWING TABLE SHOULD BE HELPFUL.

ALGORITHM	REFERENCE	INTEGER
REMOVE	NO	YES
INSERT	YES	YES
DELINK	YES	YES
ALLOCATE	NO	NO
NEXT	NO	YES
LAST	NO	YES
FIRST	NO	YES
PRIOR	YES	YES
EMPTY	NO	YES
FULL	NO	YES

POPULATION	NO	YES
ALL OTHERS	YES	YES

EXAMPLES:

```
REFERENCE LASTREADY;
QUEUE READYLIST:FIRSTREADY(STKNR,PRIORITY:NEXTREADY, PREADY);
                        VALUE NEXTREADY,PREADY,STKNR,PRIORITY;
                        INTEGER STKNR,PRIORITY;
                        REFERENCE NEXTREADY,PREADY;
USING
TO INSERT, IF LASTREADY=NULL

     THEN NEXTREADY@(LASTREADY←FIRSTREADY←ENTRY)←PREADY←NULL
     ELSE IF PRIORITY@ENTRY = PRIORITY@(PREADY*(ENTRY)←LASTREADY)
     THEN
          NEXTREADY@(NEXTREADY@(LASTREADY)←LASTREADY←ENTRY)←NULL
     ELSE IF PRIOITY@ENTRY < PRIORITY THEN COMMENT GOES AT HEAD;
          PREADY@(PREADY@(NEXTREADY@(ENTRY)←FIRSTREADY)←FIRSTREADY
               ←ENTRY)←NULL
     ELSE
     BEGIN
          WHILE PRIORITY@ENTRY<PRIORITY@PREADY@ENTRY DO
               PREADY@(ENTRY)←PREADY@PREADY@ENTRY;
          NEXTREADY@(ENTRY)←NEXTREADY@PREADY@ENTRY;
          NEXTREADY@(PREADY@ENTRY)←PREADY@(NEXTREADY@ENTRY)←ENTRY
     END:
EMPTY IF FIRSTREADY=NULL:
TO REMOVE,
          IF ENTRY=FIRSTREADY THEN
               IF FIRSTREADY=LASTREADY THEN
               FIRSTREADY←LASTREADY←NULL
               ELSE PREADY@(FIRSTREADY←NEXTREADY)←NULL
          ELSE IF ENTRY=LASTREADY THEN
               NEXTREADY@(LASTREADY←PREADY@ENTRY)←NULL
          ELSE PREADY@(NEXTREADY@(PREADY@ENTRY)←NEXTREADY@ENTRY)
```

```
←PREADY@ENTRY;
```

7.15. TYPE DECLARATIONS

SYNTAX:

```
<TYPE DECLARATION> ::= <TYPE> <TYPE LIST> /
      OWN <TYPE> <TYPE LIST>
<TYPE> ::= BOOLEAN / DOUBLE / EVENT / INTEGER / NAME / POINTER /
      REAL / REFERENCE / WORD
<TYPE LIST> ::= <TYPE PART> / <TYPE LIST> , <TYPE PART>
<TYPE PART> ::= <TYPE IDENTIFIER> <ADDRESS PART> /
      <TYPE IDENTIFIER> <REPLACEMENT OPERATOR> <INITIAL VALUE>
<ADDRESS PART> ::= <EMPTY> / = <ADDRESS>
<ADDRESS> ::= <IDENTIFIER> / <ADDRESS COUPLE> /
      <IDENTIFIER> <ADDING OPERATOR> <UNSIGNED INTEGER>
<ADDRESS COUPLE> ::= (<LEVEL> <DISPLACEMENT>)
<LEVEL> ::= <UNSIGNED INTEGER> / - <UNSIGNED INTEGER>
<DISPLACEMENT> ::= <EMPTY> / , <UNSIGNED INTEGER>
<INITIAL VALUE> ::= <EXPRESSION>
<TYPE IDENTIFIER> ::= <IDENTIFIER>
```

SEMANTICS:

1. A <TYPE DECLARATION> DEFINES THE TYPE OF VALUE OF EACH <TYPE IDENTIFIER>.

2. A <TYPE DECLARATION> MAY ALSO ASSIGN THE ADDRESS OF THE IDENTIFIER, OR ASSIGN THE INITIAL VALUE OF THE IDENTIFIER, OR SPECIFY THAT THE IDENTIFIER HAS THE PROPERTY SPECIFIED BY THE DECLARATOR. THE OWN PROPERTY ACTION IS ESSENTIALLY THE SAME AS THAT WHICH WOULD OCCUR IF THE PROGRAMMER WERE TO SPECIFY AN ADDRESS COUPLE REFERENCING THE ZERO LEVEL WITH AN EMPTY DISPLACEMENT.

3. AN <ADDRESS> WHICH IS AN <IDENTIFIER> ASSIGNS THE <TYPE IDENTIFIER> TO THE SAME LOCATION AS THE <IDENTIFIER>. THE

<IDENTIFIER> MUST HAVE BEEN DECLARED PREVIOUSLY AND MUST IDENTIFY A QUANTITY HAVING A STACK ADDRESS.

4. AN EMPTY ADDRESS PART RESULTS IN THE ASSIGNMENT OF AN ADDRESS COUPLE BY THE COMPILER.

5. AN ADDRESS COUPLE IS AN ADDRESSING LEVEL AND A DISPLACEMENT FROM THE BASE OF THAT LEVEL.

6. THE ADDRESSING LEVEL MAY RANGE FROM ZERO TO 31.

7. A NEGATIVE INTEGER USED AS A LEVEL DIRECTS THE COMPILER TO DETERMINE THE LEVEL REFERENCED BY SUBTRACTING THE INTEGER FROM THE LEVEL OF THE BLOCK BEING COMPILED.

8. AN EMPTY DISPLACEMENT DIRECTS THE COMPILER TO USE THE NEXT AVAILABLE DISPLACEMENT FOR THE LEVEL INDICATED.

9. THE RANGE OF DISPLACEMENT DEPENDS UPON THE VALUE OF LEVEL:

LEVEL	DISPLACEMENT RANGE
0,1	0 - 8191
2,3	0 - 4095
4 - 7	0 - 2047
8 - 15	0 - 1023
16 - 31	0 - 511

10. THE SPECIFIED LEVEL MUST BE LESS THAN OR EQUAL TO THE CURRENT LEVEL.

11. IT IS POSSIBLE FOR TWO OR MORE IDENTIFIERS TO HAVE THE SAME ADDRESS COUPLE (DIRECTED BY THE PROGRAMMER). IT IS THE RESPONSIBILITY OF THE PROGRAMMER TO SEE THAT THE ANTECEDENTS OF AN ADDRESS COUPLE USED AS AN ADDRESS PART ARE CORRECT.

12. AN INITIAL VALUE EXPRESSION ASSIGNS THE VALUE THAT THE

IDENTIFIER HAS UPON ENTERING THE BLOCK IN WHICH THE TYPE DECLARATION APPEARS. THE EXPRESSION IS EVALUATED UPON EACH ENTRY TO THE BLOCK. ALL INITIAL VALUE EXPRESSIONS ARE EVALUATED IN THE ORDER IN WHICH THEY APPEAR.

13. AN INITIAL VALUE EXPRESSION MUST BE OF THE SAME TYPE AS THE DECLARATION.

14. IF AN INITIAL VALUE IS NOT SPECIFIED FOR ANY TYPE IDENTIFIER, THE INITIAL VALUE OF THAT IDENTIFIER IS ZERO.

EXAMPLES:

TYPE DECLARATIONS:

```
OWN REAL V, Q;
OWN INTEGER B,A;
INTEGER Q←1, R, S=T, V=(3,4);
```

TYPE LIST:

```
Q
C=W
GENERAL =(1,2),F12T←5,Z=(3)
```

TYPE PART:

```
ZERO←1
QUOTE = (3,4)
TERIF=GRT
```

ADDRESS:

```
ZQW
```

(3,4)

INITIAL VALUE:

```
3
IF B THEN XxQ ELSE BOD+NOTHING
```

7.16. VALUE ARRAY DECLARATIONS

SYNTAX:

<VALUE ARRAY DECLARATION> ::= <ARRAY SAVE PART> <VALUE TYPE>
VALUE ARRAY <VALUE ARRAY PART LIST>
<ARRAY SAVE PART> ::= <EMPTY> / SAVE
<VALUE TYPE> ::= REAL / INTEGER / DOUBLE / BOOLEAN
<VALUE ARRAY PART LIST> ::= <VALUE ARRAY IDENTIFIER>
<REPLACEMENT OPERATOR> (<CONSTANT LIST>) /
<VALUE ARRAY IDENTIFIER> (<CONSTANT LIST>)
<CONSTANT LIST> ::= <CONSTANT> / <CONSTANT LIST> , <CONSTANT>
<CONSTANT> ::= <NUMBER> / <LOGICAL VALUE> / <STRING> /
<CONSTANT EXPRESSION> / <UNSIGNED INTEGER> (<CONSTANT LIST>)
<CONSTANT EXPRESSION> ::= ⊿AN EXPRESSION WHICH CAN BE ENTIRELY
EVALUATED BY THE ESPOL COMPILER AT COMPILE TIME⊾

SEMANTICS:

THE VALUE ARRAY DECLARATION DEFINES A ONE-DIMENSIONAL ARRAY OF VALUES.

A <CONSTANT EXPRESSION> CAN CONSIST OF ANYTHING WHICH THE ESPOL COMPILER CAN COMPLETELY EVALUATE AT COMPILE TIME. THIS INCLUDES CONCATENATION, VARIOUS <INTRINSIC>S SUCH AS FIRSTONE, ONES, ETC.

EXAMPLES:

```
SAVE BOOLEAN VALUE ARRAY DISPLAYOPT←(TRUE, FALSE, FALSE, TRUE)
VALUE ARRAY EPSILON(8"2.7182818284590452353602874713526 6")
```

8. STATEMENTS

8.1. GENERAL

THE UNITS OF OPERATION WITHIN THE LANGUAGE ARE CALLED STATEMENTS.

SYNTAX:

```
<STATEMENT> ::= <CONDITIONAL STATEMENT> / <UNCONDITIONAL STATEMENT>
<CONDITIONAL STATEMENT> ::= <IF CLAUSE> <STATEMENT> /
      <IF CLAUSE> <UNCONDITIONAL STATEMENT> ELSE
      <CONDITIONAL STATEMENT> / <CONDITIONAL ITERATION> /
      <LABEL> : <CONDITIONAL STATEMENT>
<IF CLAUSE> ::= IF <BOOLEAN EXPRESSION> THEN
<LABEL> ::= <LABEL IDENTIFIER>
<UNCONDITIONAL STATEMENT> ::= <EMPTY> / <IF CLAUSE>
      <UNCONDITIONAL STATEMENT> ELSE <UNCONDITIONAL STATEMENT> /
      <LABEL> : <UNCONDITIONAL STATEMENT> /
      <ASSIGNMENT STATEMENT> / <BASIC STATEMENT> /
      <BLOCK> / <COMPOUND STATEMENT> / <DEFINE INVOCATION> /
      <EVENT STATEMENT> / <FORK STATEMENT> /
      <FUNCTION DESIGNATOR> / <I/O STATEMENT> /
      / <INTERRUPT STATEMENT> / <ON STATEMENT> /
      <PROCEDURE STATEMENT> / <REPLACE STATEMENT> /
      <SCAN STATEMENT> / <UNCONDITIONAL ITERATION>
```

SEMANTICS:

1. STATEMENTS CAN BE LABELED.

2. A CONDITIONAL STATEMENT CAUSES CERTAIN STATEMENTS TO BE EXECUTED OR SKIPPED, DEPENDING ON THE VALUE PRODUCED BY A BOOLEAN EXPRESSION.

EXAMPLES:

STATEMENTS:

```
IF X>B THEN X←X+1 ELSE GO TO B2
X←A+B
```

CONDITIONAL STATEMENTS:

```
B2 : IF X>0 THEN N←N+1
IF TREU THEN V:Q←N+M ELSE IF NOT BOO THEN Q:= M/N
IF B←F(A) THEN GO TO START
WHILE TRUE DO IF X←Q+M>Z THEN GO TO L6
```

UNCONDITIONAL STATEMENTS:

```
LBL: GO TO NEXT
IF Z>X THEN GO TO FLAS ELSE GO SCND
BEGIN Y←X+1;Z←Y+2 END
BEGIN REAL X; LABEL Q; IF Z>P THEN GO TO Q ELSE X←FC(C);
Q:END
LBL:
```

IF CLAUSES:

```
IF B>A THEN
IF GATE[1,2] AND GATE[1,3]THEN
```

8.2. ASSIGNMENT STATEMENTS

SYNTAX:

```
<ASSIGNMENT STATEMENT> ::= <ARITHMETIC ASSIGNMENT STATEMENT> /
      <ARRAY ASSIGNMENT STATEMENT> /
      <BOOLEAN ASSIGNMENT STATEMENT> / <QUEUE ASSIGNMENT> /
      <REFERENCE ASSIGNMENT> / <WORD ASSIGNMENT>
<ARITHMETIC ASSIGNMENT STATEMENT> ::= <ARITHMETIC ASSIGNMENT> /
      <ARITHMETIC FIELD ASSIGNMENT>
<ARITHMETIC FIELD ASSIGNMENT> ::=
      <ARITHMETIC VARIABLE> . <FIELD IDENTIFIER>
      <REPLACEMENT OPERATOR> <ARITHMETIC EXPRESSION>
<ARRAY ASSIGNMENT STATEMENT> ::= <ARRAY ASSIGNMENT> /
      <ARRAY FIELD ASSIGNMENT>
<ARRAY FIELD ASSIGNMENT> ::=
      <ARRAY DESIGNATOR> . <FIELD IDENTIFIER>
      <REPLACEMENT OPERATOR> <ARITHMETIC EXPRESSION>
<BOOLEAN ASSIGNMENT STATEMENT> ::= <BOOLEAN ASSIGNMENT> /
      <BOOLEAN FIELD ASSIGNMENT>
<BOOLEAN FIELD ASSIGNMENT> ::=
      <BOOLEAN VARIABLE> . <FIELD IDENTIFIER>
      <REPLACEMENT OPERATOR> <BOOLEAN EXPRESSION>
<QUEUE ASSIGNMENT> ::= <QUEUE DESIGNATOR> <REPLACEMENT OPERATOR>
      <REFERENCE EXPRESSION>
```

SEMANTICS:

1. THE ASSIGNMENT STATEMENT CAUSES THE EXPRESSION TO THE RIGHT OF THE REPLACEMENT OPERATOR TO BE EVALUATED. THE VALUE IS THEN ASSIGNED (OR TRANSFERRED) TO THE VARIABLE OR FIELD ON THE LEFT

2. A QUEUE ASSIGNMENT CAUSES THE ENTRY DENOTED BY THE VALUE OF THE REFERENCE EXPRESSION TO BE MADE AVAILABLE TO THE INSERTION PART OF THE DESIGNATED QUEUE: IT THEN CAUSES THIS INSERTION

PART TO BE EXECUTED.

PRAGMATICS:

1. <WORD ASSIGNMENT>S INVOKE OVRD ACTION.

2. THERE IS NO GUARANTEE THAT A CORRECTLY COMPILED <WORD ASSIGNMENT> WILL PRODUCE VALID B6700/B7700 CODE.

EXAMPLES:

ARITHMETIC ASSIGNMENT STATEMENTS:

```
V←BxQ-R
JOY.BOY← V←Q
```

ARITHMETIC ASSIGNMENTS:

```
ALTHO ← N+1
S[V,K+2]:=3-FUNC(Q+2)
```

ARITHMETIC FIELD ASSIGNMENTS:

```
IOQUEUE.ERR← 6
V[7].LNK := QUED-GONEx7
```

ARRAY ASSIGNMENT:

```
NEXT←IF A[1,2]AND B[Q,3] THEN VARY[2,*] ELSE A253-A234
```

BOOLEAN ASSIGNMENT STATEMENTS:

```
RDY←TRUE
```

```
PARTWAY.NXT ← B AND GONE > 6
```

QUEUE ASSIGNMENT:

```
QUED ←QUEB(HERE,THERE,EVYWHR)
```

8.3. BASIC STATEMENTS

SYNTAX:

```
<BASIC STATEMENT> ::= <GO TO STATEMENT> /
      <DUMMY STATEMENT> / <CASE STATEMENT>
<GO TO STATEMENT> ::= GO <TO PART> <DESIGNATIONAL EXPRESSION> /
      GO <TO PART> <CASE HEAD> (<DESIGNATIONAL EXPRESSION LIST>)
      GO <TO PART> <WORD VARIABLE>
<TO PART> ::= <EMPTY> / TO
<DUMMY STATEMENT> ::= <EMPTY> / <DUMMY STATEMENT> <LABEL> :
<CASE STATEMENT> ::= <CASE HEAD> <COMPOUND STATEMENT>
<CASE HEAD> ::= CASE <ARITHMETIC EXPRESSION> OF
```

SEMANTICS:

1. THE GO TO STATEMENT TRANSFERS CONTROL TO THE LABEL WHICH IS THE VALUE OF THE DESIGNATIONAL EXPRESSION OR THE DESIGNATIONAL EXPRESSION LIST. IF A <WORD VARIABLE> IS USED, IT IS PLACED IN THE TOP OF STACK AND AN UNCONDITIONAL DYNAMIC BRANCH IS EXECUTED.

2. IN THE <CASE STATEMENT>, THE <ARITHMETIC EXPRESSION> IN THE <CASE HEAD> IS EVALUATED AND IS USED TO SELECT ONE OF THE STATEMENTS IN THE <COMPOUND STATEMENT> FOLLOWING THE <CASE HEAD>. THE SELECTED STATEMENT AND ONLY THE SELECTED STATEMENT WILL BE EXECUTED.

   THE STATEMENTS IN THE <COMPOUND STATEMENT> ARE NUMBERED STARTING WITH ZERO, AND THE <ARITHMETIC EXPRESSION> IN THE <CASE HEAD> IS INTERPRETED AS THE NUMBER OF THE STATEMENT TO BE EXECUTED. DUMMY STATEMENTS MAY BE USED TO ARRANGE CONVENIENT NUMBERING OF THE STATEMENTS.

3. IF THE VALUE OF OF THE <ARITHMETIC EXPRESSION> IS GREATER THAN

THE NUMBER OF STATEMENTS. ANYTHING CAN HAPPEN. FOR PURPOSES OF EFFICIENCY, THE ESPOL COMPILER DOES NOT EMIT "RANGE CHECKING CODE" TO DETERMINE VALIDITY OF THE EVALUATED <ARITHMETIC EXPRESSION>.

EXAMPLES:

GO TO STATEMENTS:

```
GO TO START
GO NEXT
GO TO CASE ARITHEXP-1 OF (LABEL1, NEXT, EXIT, START)
```

DUMMY STATEMENTS:

```
L1:
EXIT: NEXT:
```

CASE STATEMENT:

```
CASE V OF BEGIN X←X+1;Z←Z+1 END
```

CASE HEAD:

```
CASE Z←QxV-B MOD 7 OF
```

8.4. CONDITIONAL AND UNCONDITIONAL ITERATION STATEMENTS

SYNTAX:

```
<CONDITIONAL ITERATION> ::= <ITERATION CLAUSE>
      <CONDITIONAL STATEMENT>
<UNCONDITIONAL ITERATION> ::= <DO STATEMENT> /
      <ITERATION CLAUSE> <UNCONDITIONAL STATEMENT>
<DO STATEMENT> ::= DO <STATEMENT> UNTIL <BOOLEAN EXPRESSION>
<ITERATION CLAUSE> ::= <WHILE PART> DO / <FOR CLAUSE> DO /
      <THRU CLAUSE> DO
<WHILE PART> ::= WHILE <BOOLEAN EXPRESSION>
<FOR CLAUSE> ::= FOR <CONTROLLED VARIABLE>
      <REPLACEMENT OPERATOR> <FOR PART>
<CONTROLLED VARIABLE> ::= <SIMPLE VARIABLE> /
      <SUBSCRIPTED VARIABLE>
<FOR PART> ::= <INITIAL PART> <STEP PART> <FINAL PART>
<INITIAL PART> ::= <ARITHMETIC EXPRESSION>
<STEP PART> ::= STEP <ARITHMETIC EXPRESSION> / BY
      <ARITHMETIC EXPRESSION>
<FINAL PART> ::= UNTIL <ARITHMETIC EXPRESSION> / <WHILE PART>
<THRU CLAUSE> ::= THRU <ARITHMETIC EXPRESSION>
```

SEMANTICS:

ITERATION PROVIDES A MEANS OF FORMING LOOPS IN A PROGRAM. IF "BY <ARITHMETIC EXPRESSION>" IS USED INSTEAD OF "STEP <ARITHMETIC EXPRESSION>", THE LOOP WILL BE CONSTRUCTED USING THE STEP AND BRANCH OPERATOR TO OPTIMIZE EXECUTION TIME.

THE <THRU CLAUSE> FORM OF ITERATION OPERATES AS FOLLOWS: THE <ARITHMETIC EXPRESSION> IS EVALUATED AND THE <STATEMENT> FOLLOWING "DO" EXECUTED THE NUMBER OF TIMES INDICATED BY THAT VALUE. THE HARDWARE OPERATOR WHICH IMPLEMENTS THIS FORM OF ITERATION RESTRICTS THE BINARY REPRESENTATION OF THE INTEGERIZED <ARITHMETIC

EXPRESSION> TO 16 BITS THUS THIS VALUE MUST LIE IN THE RANGE 0 THROUGH 65,535.

EXAMPLES:

ITERATION CLAUSES:

```
FOR V := Q STEP 1 UNTIL GONE DO
FOR FIRST ← 1 BY 1 UNTIL LAST DO
```

WHILE PARTS:

```
WHILE NOT A ≠ C EQV GATE [1,2]
WHILE TRUE
```

FOR CLAUSES:

```
FOR ATLST ← J-2 STEP A ← Q ← J-K UNTIL F(A)
```

CONTROLLED VARIABLES:

```
ATLST
V[2,3]
```

FOR PARTS:

```
X←2 STEP X←Y+FUNC(A)UNTIL X←Y MOD 9
```

INITIAL PARTS:

```
P MOD 2
+3
Q
```

STEP PARTS:

```
STEP IF B =0 THEN X ELSE Y+2
BY 29
```

FINAL PARTS:

```
UNTIL X-7
WHILE B>8
```

8.5. EVENT STATEMENTS

SYNTAX:

```
<EVENT STATEMENT> ::= <SET STATEMENT> / <RESET STATEMENT> /
      <CAUSE STATEMENT> / <CAUSEANDRESET STATEMENT> /
      <WAIT STATEMENT> / <WAITANDRESET STATEMENT> /
      <DSWAIT STATEMENT> / <DSWAITANDRESET STATEMENT> /
      <PROCURE STATEMENT> / <FIX STATEMENT> /
      <LIBERATE STATEMENT> / <FREE STATEMENT>
<SET STATEMENT> ::= SET (<EVENT DESIGNATOR>)
<RESET STATEMENT> ::= RESET (<EVENT DESIGNATOR>)
<CAUSE STATEMENT> ::= CAUSE (<EVENT DESIGNATOR>)
<CAUSEANDRESET STATEMENT> ::= CAUSENRESET (<EVENT DESIGNATOR>)
<WAIT STATEMENT> ::= <WAITONEVENT STATEMENT> /
      <WAITONTIME STATEMENT>
<WAITONEVENT STATEMENT> ::= <WAITONEVENT-SIMPLE STATEMENT> /
      <WAITONEVENT-COMPLEX STATEMENT>
<WAITONEVENT-SIMPLE STATEMENT> ::= WAIT (<EVENT DESIGNATOR>)
<WAITONEVENT-COMPLEX STATEMENT> ::= WAIT (<WAIT PARAMETER LIST>)
<WAIT PARAMETER LIST> ::= <EVENT LIST> / (<TIME>), <EVENT LIST>
<EVENT LIST> ::= <EVENT DESIGNATOR> / <EVENT LIST>,
      <EVENT DESIGNATOR>
<TIME> ::= ⌐THE MINIMUM AMOUNT OF TIME IN SECONDS
      (FRACTIONAL SECONDS ALLOWED) THAT A TASK IS TO BE
      SUSPENDED⌙
<WAITONTIME STATEMENT> ::= WAIT ((<TIME>))
<WAITANDRESET STATEMENT> ::= <WAITANDRESET-SIMPLE STATEMENT>/
      <WAITANDRESET-COMPLEX STATEMENT>
<WAITANDRESET-SIMPLE STATEMENT> ::= WAITANDRESET
      (<EVENT DESIGNATOR>)
<WAITANDRESET-COMPLEX STATEMENT> ::= WAITANDRESET
      (<WAIT PARAMETER LIST>)
<DSWAIT STATEMENT> ::= <DSWAIT-SIMPLE STATEMENT> /
      <DSWAIT-COMPLEX STATEMENT>
```

<DSWAIT-SIMPLE STATEMENT> ::= DSWAIT (<EVENT DESIGNATOR>)

<DSWAIT-COMPLEX STATEMENT> ::= DSWAIT (<WAIT PARAMETER LIST>)

<DSWAITANDRESET STATEMENT> ::= <DSWAITANDRESET-SIMPLE STATEMENT> /
<DSWAITANDRESET-COMPLEX STATEMENT>

<DSWAITANDRESET-SIMPLE STATEMENT> ::= DSWAITANDRESET
(<EVENT DESIGNATOR>)

<DSWAITANDRESET-COMPLEX STATEMENT> ::= DSWAITANDRESET
(<WAIT PARAMETER LIST>)

<PROCURE STATEMENT> ::= PROCURE (<EVENT DESIGNATOR>)

<FIX STATEMENT> ::= FIX (<EVENT DESIGNATOR>)

<LIBERATE STATEMENT> ::= LIBERATE (<EVENT DESIGNATOR>)

<FREE STATEMENT> ::= FREE (<EVENT DESIGNATOR>)

SEMANTICS:

AN EVENT HAS TWO PROPERTIES, EACH PROPERTY HAVING TWO STATES. THE CONDITION-ORIENTED PROPERTY HAS EITHER A HAPPENED OR NOT HAPPENED STATE, AND THE RESOURCE-ORIENTED PROPERTY HAS EITHER AN AVAILABLE OR NOT AVAILABLE STATE. THE INITIAL STATE OF EACH PROPERTY IS NOT HAPPENED AND AVAILABLE. ASSOCIATED WITH EACH PROPERTY ARE STATEMENTS WHICH MAY BE USED TO CONTROL AND/OR INTERROGATE THE STATE OF THE PROPERTY.

THE <SET STATEMENT> WILL SET AN EVENT TO THE HAPPENED STATE. IT WILL NOT CAUSE ANY OTHER ACTION, I.E. THE <SET STATEMENT> WILL NOT ACTIVATE A TASK WAITING ON THE EVENT.

THE <RESET STATEMENT> WILL RESET AN EVENT TO THE NOT HAPPENED STATE IT WILL NOT CAUSE ANY OTHER ACTION.

THE <CAUSE STATEMENT> WILL ACTIVATE ALL TASKS WHICH ARE WAITING ON THE EVENT. NORMALLY THE CAUSE WILL ALSO SET THE EVENT TO THE HAPPENED STATE. (SEE THE <WAITANDRESET STATEMENT> FOR EXCEPTIONS).

IT MUST BE POINTED OUT THAT ACTIVATING A TASK DOES NOT IMPLY THE TASK GOES INTO IMMEDIATE EXECUTION. ACTIVATING A TASK CONSISTS OF

DELINKING THE TASK FROM AN EVENT QUEUE (EACH EVENT HAS ITS OWN QUEUE), AND LINKING THAT TASK IN PRIORITY ORDER INTO A SYSTEM QUEUE CALLED THE READYQ. THE READYQ IS A QUEUE OF ALL TASKS, IN PRIORITY ORDER, THAT ARE CAPABLE OF RUNNING. TASKS ARE TAKEN OUT OF THE READYQ WHEN EITHER A PROCESSOR IS ASSIGNED TO THE TASK, OR THE TASK MUST WAIT FOR SOMETHING SUCH AS AN I/O OPERATION OR AN EVENT TO BE CAUSED. A TASK WILL ONLY BE PLACED INTO ACTUAL EXECUTION WHEN IT IS THE TOP ITEM IN THE READYQ AND A PROCESSOR IS AVAILABLE.

THE <CAUSEANDRESET STATEMENT> IS SIMILAR TO THE <CAUSE STATEMENT> IN THAT IT ACTIVATES ALL TASKS WAITING ON THE EVENT. IT VARIES FROM THE <CAUSE STATEMENT> IN THAT THE RESULTANT STATE OF THE EVENT IS SET TO NOT HAPPENED.

FOR THE <WAITONEVENT STATEMENT>, THE HAPPENED STATE OF THE EVENT IS EXAMINED. IF THE EVENT STATE IS HAPPENED, THE STATEMENT IS ESSENTIALLY A "NO-OPERATION". IF THE EVENT STATE IS NOT HAPPENED, THE TASK WILL BE SUSPENDED UNTIL SUCH TIME SOME OTHER TASK EXECUTES A <CAUSE STATEMENT>.

THE <WAIT STATEMENT> ALLOWS FOR SUSPENDING A TASK EITHER FOR A TIME PERIOD OR UNTIL AN EVENT IS CAUSED.

FOR THE <WAITONTIME STATEMENT>, THE TIME FUNCTION GENERATES AN IMPLICIT EVENT WHICH IT WAITS ON. THIS EVENT IS CAUSED BY THE OPERATING SYSTEM WHEN IT DETECTS THE TIME PERIOD SPECIFIED BY <TIME> HAS ELAPSED. IT MIGHT BE NOTED THAT DEPENDING ON THE DEGREE OF MULTIPROCESSING BEING PERFORMED AND PROGRAM PRIORITIES, THE ACTUAL TIME A TASK IS SUSPENDED FOR <TIME> MAY VARY WIDELY IN RESPECT TO THE TIME INDICATED IN <TIME> WITH SMALLER INCREMENTS OF TIME HAVING THE GREATEST VARIATION.

THE <WAITANDRESET STATEMENT> ALLOWS FOR SUSPENDING A TASK UNTIL THE EVENT IS CAUSED. IT IS IDENTICAL TO THE <WAIT STATEMENT> EXCEPT THAT THE EVENT IS FORCED TO THE STATE NOT HAPPENED DURING THE SUBSEQUENT CAUSE PROCESSES.

THE <WAITONEVENT STATEMENT> AND <WAITANDRESET STATEMENT> ARE IDENTICAL EXCEPT FOR THE STATE TO WHICH THE CAUSED EVENT IS SET DURING THE CAUSE PROCESS. IF ALL TASKS ARE WAITING ON THE EVENT VIA THE <WAITONEVENT STATEMENT> (SIMPLE OR COMPLEX), THE STATE OF THE EVENT IS SET TO HAPPENED. IF ANY ONE TASK IS WAITING ON THE EVENT VIA THE <WAITANDRESET STATEMENT>, THE STATE OF THE EVENT IS RESET TO NOT HAPPENED.

THE <WAITONEVENT-COMPLEX STATEMENT> ALLOWS A TASK TO BE SUSPENDED UNTIL ANY ONE EVENT IN THE <EVENT LIST> IS CAUSED OR UNTIL THE TIME AS INDICATED BY <TIME> HAS ELAPSED. THE <WAITONEVENT-COMPLEX STATEMENT> MAY BE USED AS AN INTEGER FUNCTION WHICH RETURNS A VALUE (STARTING AT 1) WHICH REPRESENTS THE POSITION IN THE <WAIT PARAMETER LIST> OF THE PARAMETER WHICH CAUSED THE TASK TO BE ACTIVATED. FOR EXAMPLE, IN THE STATEMENT:

```
T ::= WAIT ((.0001), E1,E2)
```

THE VALUE OF T WOULD BE 1 IF ELAPSED TIME CAUSED THE TASK TO BE ACTIVATED, WHILE IN THE EXAMPLE:

```
T ::= WAIT (E1, E2, E3)
```

THE VALUE OF T WOULD BE 2 IF A CAUSE ON EVENT E2 ACTIVATED THE TASK

IT MIGHT BE NOTED THAT THE IMPLEMENTATION OF THIS MECHANISM CONTAINS INTERLOCKS TO GUARANTEE THAT ONE AND ONLY ONE PARAMETER CAN ACTIVATE A TASK.

IF TIME IS INCLUDED AS A PARAMETER, IT MUST THE FIRST PARAMETER IN THE LIST (AND ENCLOSED IN PARENTHESES).

THERE EXIST SOME SITUATIONS WHEREBY CERTAIN OPERATING SYSTEM FUNCTIONS CANNOT BE TERMINATED WHILE WAITING ON AN EVENT, SUCH AS WHILE WAITING FOR AN I/O COMPLETE. FOR THIS REASON, A REQUEST TO TERMINATE A SYSTEM FUNCTION WHICH IS IN TURN WAITING ON AN EVENT

WILL NOT BE HONORED. HOWEVER, THERE ARE OTHER OPERATING SYSTEM FUNCTIONS FOR WHICH NO SYSTEM PROBLEM OCCURS IF THEY ARE TERMINATED WHILE WAITING ON EVENT. FOR THESE FUNCTIONS, THE <DSWAIT STATEMENT> MAY BE USED.

TERMINATION OF OPERATING SYSTEM FUNCTIONS IS SOMEWHAT DIFFERENT THAN FOR OBJECT TASKS. FIRST OF ALL, AN INTERLOCK IS SET SO THAT CONTROL CANNOT RETURN TO THE OBJECT PROGRAM WHICH CALLED THE SYSTEM FUNCTION. THE SYSTEM FUNCTION THEN IS ACTIVATED AS IF THE EVENT HAD BEEN CAUSED. MEANS EXIST WHEREBY A SYSTEM FUNCTION CAN DETERMINE IF IT WAS ACTIVATED BY A CAUSE OF AN EVENT OR BY A TERMINATE REQUEST ("DS").

THE <DSWAIT-COMPLEX STATEMENT> IS SIMILAR TO THE <DSWAIT-SIMPLE STATEMENT> IN THAT IT ALLOWS A TERMINATE REQUEST AGAINST AN OPERATING SYSTEM FUNCTION TO BE HONORED. IT IS SIMILAR ALSO TO THE <WAITONEVENT-COMPLEX STATEMENT> IN THAT IT ALLOWS SUSPENSION FOR A SPECIFIED AMOUNT OF TIME OR FOR A CAUSE OF ANY ONE OF A LIST OF EVENTS. IT RETURNS AN INTEGER SIMILAR TO THE <WAITONEVENT-COMPLEX STATEMENT> EXCEPT THAT A ZERO (0) IS RETURNED IF THE FUNCTION WAS ACTIVATED DUE TO A TERMINAL REQUEST.

BY THIS TIME THE READER MAY BE CONCERNED AS TO SETTING OF THE HAPPENED STATE OF AN EVENT JUST AFTER AN EVENT IS CAUSED. ESSENTIALLY, ANY ONE TASK WAITING ON THE EVENT VIA THE <WAITANDRESET STATEMENT> OR THE EVENT CONTROL TASK EXECUTING A <CAUSEANDRESET STATEMENT> WILL RESULT IN THE STATE NOT HAPPENED. ALL TASKS MUST BE WAITING USING THE <WAIT STATEMENT> AND THE EVENT CONTROL TASK MUST USE THE <CAUSE STATEMENT> IN ORDER FOR THE EVENT TO HAVE THE HAPPENED STATE IMMEDIATELY AFTER THE CAUSE PROCESS. FOR THIS REASON, EXTREME CARE MUST BE TAKEN WHEN USING THESE STATEMENTS. AS A GUIDE TO THE PROPER USE OF THE WAIT, WAITANDRESET, CAUSE, AND CAUSEANDRESET STATEMENTS, CONSIDER THERE ARE TWO TYPES OF CONDITIONS, MOMENTARY AND ELAPSED.

A MOMENTARY CONDITION CAN BE DEFINED AS A CONDITION WHICH CAN EXIST

FOR ONLY AN INSTANT. THE RESET OPTIONS OF CAUSE AND WAIT STATEMENTS ALLOW THE IMPLEMENTATION OF MOMENTARY CONDITIONS. FOR THE SITUATION WHERE AN EVENT CONTROL TASK (THE ONE CAUSING THE EVENT) SOLELY DETERMINES THE CONDITION, THE DEPENDENT STACKS (THOSE WAITING) SHOULD USE THE <WAIT STATEMENT> WHILE THE EVENT CONTROL STACK SHOULD USE THE <CAUSEANDRESET STATEMENT>.

FOR THE CASE WHERE THE ACT OF A TASK BEING ACTIVATED MEANS THE CONDITION IS TO DISAPPEAR, THE DEPENDENT TASK SHOULD USE THE <WAITANDRESET STATEMENT> WHILE THE EVENT CONTROL TASK SHOULD USE THE <CAUSE STATEMENT>. ONE SHOULD AVOID MIXING WAIT, WAITANDRESET, CAUSE AND CAUSEANDRESET ALL ON THE SAME EVENT. THE RESULTANT CONFUSION OVER THE HAPPENED STATE OF THE EVENT CAN CAUSE A CONSIDERABLE PROBLEM.

AN ELAPSED CONDITION CAN BE DEFINED AS A CONDITION WHICH HOLDS OVER A LONG PERIOD. TO IMPLEMENT THIS TYPE OF CONDITION, ONE SHOULD USE ONLY THE WAIT, CAUSE, AND RESET STATEMENTS.

THE SECOND PROPERTY OF AN EVENT IS THE RESOURCE-ORIENTED PROPERTY. ONE CAN CONSIDER A RESOURCE AS SOMETHING WHICH CAN BE UTILIZED BY SEVERAL TASKS BUT ONLY ONE TASK AT A TIME. FOR EXAMPLE, LET US CONSIDER A COMPLEX LIST STRUCTURE BUILT INTO AN ARRAY AS A RESOURCE LET US ALSO ASSUME THAT THERE EXISTS A TASK WHICH ADDS DATA TO THE LIST STRUCTURE AND ANOTHER TASK WHICH DELETES FROM THE LIST STRUCTURE WE ALSO ASSUME THAT BOTH TASKS RUNNING SIMULTANEOUSLY COULD DESTROY THE LIST STRUCTURE. IT IS NECESSARY TO LET EITHER TASK RUN AT ANY TIME BUT NOT SIMULTANEOUSLY. THIS TYPE OF CONTROL CAN BE DONE VIA THE RESOURCE-ORIENTED PROPERTIES OF AN EVENT AND THE RESOURCE-ORIENTED STATEMENTS.

THE <PROCURE STATEMENT> TESTS THE AVAILABLE STATE OF AN EVENT. IF THE EVENT IS NOT AVAILABLE, THE EVENT STATE IS SET TO NOT AVAILABLE AND THE TASK CONTINUES IN SEQUENCE.

THE <FIX STATEMENT> (ALSO REFERRED TO AS A CONDITIONAL PROCURE

FUNCTION) IS A BOOLEAN FUNCTION WHICH EXAMINES THE AVAILABLE STATE OF AN EVENT. IF THE STATE IS AVAILABLE, THE EVENT IS PROCURED (SETS STATE TO NOT AVIALBLE) AND A "TRUE" IS RETURNED. IF THE AVAILABLE STATE WAS NOT AVAILABLE, THE FUNCTION RETURNS A "FALSE", LEAVING THE AVAILABLE STATE UNCHANGED.

THE <LIBERATE STATEMENT> WHEN EXECUTED, PRODUCES SEVERAL EFFECTS:

A. FIRST OF ALL, THE PROCURE LIST IS EXAMINED.

   1. IF THERE ARE NO OTHER TASKS WAITING TO PROCURE THE THE EVENT, THE EVENT STATE IS SET TO AVAILABLE.

   2. IF THERE ARE OTHER TASKS WAITING TO PROCURE THE EVENT, THE LAST TASK (THE IMPLEMENTATION IS LAST IN-FIRST OUT) IS ACTIVATED. THE EVENT STATE IS LEFT MARKED AS NOT AVAILABLE.

B. LASTLY, ALL TASKS WAITING ON THE EVENT ARE ACTIVATED (AN IMPLICIT CAUSE IS EXECUTED). THIS MAY RESULT IN A CHANGE TO THE HAPPENED STATE OF THE EVENT DEPENDING ON WHETHER THE TASKS WHICH WERE WAITING USED THE <WAITONEVENT STATEMENT> OR <WAITANDRESET STATEMENT>.

THE <FREE STATEMENT> (OF ARBITRARY VALUE AND DANGEROUS USE) IS A BOOLEAN FUNCTION WHICH EXAMINES THE AVAILABLILITY STATE OF AN EVENT IT RETURNS A "TRUE" IF THE EVENT IS AVAILABLE AND A "FALSE" IF THE EVENT IS NOT AVAILABLE. IN ADDITION, IT WILL RESET THE EVENT STATE UNCONDITIONALLY TO AVAILABLE BUT WILL NOT ACTIVATE ANY TASK SUSPENDED BY AN ATTEMPT TO PROCURE THE EVENT NOR WILL IT ACTIVATE ANY TASK WAITING ON THE EVENT.

8.6. FORK STATEMENT

SYNTAX:

<FORK STATEMENT> ::= FORK <PROCEDURE STATEMENT> [<FORK PARAMETERS>]
<FORK PARAMETERS> ::= <STACK SIZE>, <PRIORITY> <VISIBLE NAME INDEX>
<STACK SIZE> ::= <ARITHMETIC EXPRESSION>
<PRIORITY> ::= <ARITHMETIC EXPRESSION>
<VISIBLE NAME INDEX> ::= <EMPTY> / , <ARITHMETIC EXPRESSION>

SEMANTICS:

THE <FORK STATEMENT> PROVIDES THE MEANS OF INITIATING "INDEPENDENT RUNNERS" WITHIN THE MCP.

THE PROCEDURE REFERENCED MAY BE TYPED OR UNTYPED. IT MAY NOT BE FORMAL OR DYNAMIC. IF TYPED, THE VALUE IS LOST.

IF THERE ARE ACTUAL PARAMETERS, THEY MUST AGREE IN NUMBER AND TYPE WITH THE FORMAL PARAMETERS SPECIFIED FOR THE PROCEDURE.

THERE MAY BE TWO OR THREE <FORK PARAMETERS>. THE FIRST TWO SPECIFY THE STACK SIZE AND PRIORITY RESPECTIVELY, OF THE INITIATED PROCESS.

THE THIRD FORK PARAMETER, IF PRESENT, IS A CHARACTER INDEX INTO AN ARRAY OF INDEPENDENT RUNNER NAMES ("INDEPRUNNERNAMES"). A NON-ZERO VALUE INDICATES THE SPECIFIED NAME SHOULD BE DISPLAYED ON THE SPO WITH "BOJ" AND "EOJ" MESSAGES. AN <EMPTY> OR ZERO VALUE INDICATES SPO MESSAGES ARE NOT TO BE DISPLAYED.

EXAMPLES:

```
FORK STATUS (VEC) [IRSTACKSIZE, IRPRIORITY]
FORK LIEBNITZ (NAMEDARRAY)
```

[IRMAXSTACKSIZE, IRTOPPRIORITY, LIEBNAME]

8.7. I/O STATEMENTS

SYNTAX:

<I/O STATEMENT> ::= <CLOSE STATEMENT> / <LOCK STATEMENT> /
<READ STATEMENT> / <REWIND STATEMENT> / <SEEK STATEMENT> /
<SPACE STATEMENT> / <WRITE STATEMENT>

SEMANTICS:

INPUT/OUTPUT STATEMENTS CAUSE INFORMATION TO BE EXCHANGED BETWEEN MEMORY AREAS AND PERIPHERAL DEVICES, OR ALLOW THE USER TO PERFORM CERTAIN CONTROL FUNCTIONS. THE USER ASSUMES RESPONSIBILITY FOR HANDLING IRRECOVERABLE ERROR CONDITIONS (I.E., THE MCP I/O ROUTINES WILL TRY NORMAL AND APPROPRIATE ERROR RECOVERY; IF THESE FAIL, THE USER IS THEN RESPONSIBLE FOR THE SUBSEQUENT COURSE OF ACTION).

EXCEPTION CONDITIONS OCCURRING DURING A "READ", "WRITE", OR "SPACE" STATEMENT MAY BE HANDLED BY USING THE I/O RESULT WORD RETURNED BY THE I/O ROUTINES AS A <BOOLEAN PRIMARY>.

EXAMPLES:

```
IF BOOLVAR := READ(FILEID, 14, A[*]) THEN GO TO ERRORCOND;
BRSLT := WRITE (FYLE, AMOUNT, PID)
```

THIS METHOD MAY NOT BE USED FOR DIRECT I/O STATEMENTS. IF AN EXCEPTION CONDITION OCCURS DURING A "READ", "WRITE", OR "SPACE" STATEMENT, BIT ZERO OF THE I/O RESULT WORD WILL BE 1. IF NO EXCEPTION CONDITION OCCURS THAT BIT WILL BE (ZERO) 0. IF BIT ZERO IS 1 THEN ONE OF THE FOLLOWING BITS WILL ALSO BE 1:

BIT 9 INDICATING ON END-OF-FILE CONDITION;
BIT 7 INDICATING A PARITY ERROR OR
BIT 4 INDICATING A DATA ERROR.

IF THE "READ", "WRITE", OR "SPACE" STATMENT IS NOT USED AS AN EXPRESSION, THE COMPILER GENERATES A DELETE OPERATOR TO DISPOSE OF THE I/O RESULT WORD.

EXAMPLES:

```
CLOSE ( . . . )
LOCK ( . . . )
READ ( . . . )
REWIND ( . . . )
SEEK ( . . . )
SPACE ( . . . )
WRITE ( . . . )
```

8.7.1. CLOSE STATEMENT

SYNTAX:

<CLOSE STATEMENT> ::= CLOSE (<FILE DESIGNATOR>) /
CLOSE (<FILE DESIGNATOR> , <CLOSE OPTION>)
<CLOSE OPTION> ::= * / PURGE / REEL

SEMANTICS:

THE <CLOSE STATEMENT> WITH NO <CLOSE OPTION> CAUSES THE REFERENCED FILE TO BE CLOSED. THE FOLLOWING ACTIONS TAKE PLACE:

1. ON A CARD PUNCH FILE, A CARD CONTAINING AN ENDING LABEL IS PUNCHED.

2. ON A LINE PRINTER FILE, THE PRINTER IS SKIPPED TO CHANNEL 1, AN ENDING LABEL IS PRINTED, AND THE PRINTER IS AGAIN SKIPPED TO CHANNEL 1.

3. ON AN UNLABELED TAPE OUTPUT FILE, A DOUBLE TAPE MARK IS WRITTEN AFTER THE LAST BLOCK ON TAPE.

4. ON A LABELED TAPE OUTPUT FILE, A TAPE MARK IS WRITTEN AFTER THE LAST BLOCK ON THE TAPE THEN AN ENDING LABEL IS WRITTEN, FOLLOWED BY A DOUBLE TAPE MARK.

5. ON A DISK FILE, IF THE FILE IS A TEMPORARY FILE, THE DISK SPACE IS RETURNED.

FOR ALL TYPES OF FILES, THE BUFFER AREAS ARE RETURNED TO THE SYSTEM

IF ONLY THE <FILE DESIGNATOR> IS USED, THE I/O UNIT IS RELEASED TO THE SYSTEM. IF THE FILE IS A TAPE FILE, THE TAPE IS REWOUND.

IF "PURGE" IS USED, THE FILE IS CLOSED, PURGED, AND RELEASED TO THE SYSTEM. IF THE FILE IS A PERMANENT DISK FILE, IT IS REMOVED FROM THE DISK DIRECTORY AND THE DISK SPACE IS RETURNED.

IF THE SYMBOL "*" IS USED, THE FILE MUST BE A TAPE FILE. THE I/O UNIT REMAINS UNDER PROGRAM CONTROL AND THE TAPE IS NOT REWOUND. THIS CONSTRUCT IS USED TO CREATE MULTI-FILE REELS.

WHEN THE SYMBOL "*" IS USED ON MULTI-FILE INPUT TAPES, THE FOLLOWING ACTIONS CAN TAKE PLACE: IF THE VALUE OF THE ATTRIBUTE "DIRECTION" IS "FORWARD", THE TAPE IS POSITIONED FORWARD TO A POINT JUST FOLLOWING THE ENDING LABEL OF THE FILE; IF THE VALUE OF THE ATTRIBUTE "DIRECTION" IS "REVERSE", THE TAPE IS POSITIONED TO A POINT JUST IN FRONT OF THE BEGINNING LABEL FOR THE FILE; IF THE END-OF-FILE BRANCH HAS BEEN TAKEN, NO ACTION IS PERFORMED TO POSITION THE FILE. ON A SINGLE-FILE REEL, THE ACTION TAKEN IS THE SAME AS FOR A MULTI-FILE REEL. THE NEXT REFERENCE TO THIS FILE MUST BE "READ" IN THE OPPOSITE DIRECTION FROM THAT OF THE PRIOR "READ" ON THE FILE OR AN ERROR RESULT WORD WILL BE RETURNED.

IF "REEL" IS USED, THE FILE MUST BE A MULTI-REEL TAPE FILE. THE CURRENT REEL WILL BE CLOSED AND A SUBSEQUENT REFERENCE OF THE FILE WILL IMPLICITLY OPEN THE NEXT REEL. THIS IS PROVIDED PRIMARILY FOR THE USE OF DIRECT TAPE FILES, WHERE THE SYSTEM DOES NOT AUTOMATICALLY PERFORM REEL SWITCHING.

EXAMPLES:

```
CLOSE (FYLE)
CLOSE (FID, *)
CLOSE (LINES, PURGE)
CLOSE (FYLE, REEL)
```

8.7.2. LOCK STATEMENT

SYNTAX:

```
<LOCK STATEMENT> ::= LOCK ( <FILE DESIGNATOR> ) /
      LOCK (<FILE DESIGNATOR>, *)
```

SEMANTICS:

THE <LOCK STATEMENT> CAUSES THE REFERENCED FILE TO BE CLOSED. IF THE FILE IS TAPE, IT IS REWOUND AND A SYSTEM MESSAGE IS PRINTED TO NOTIFY THE OPERATOR TO REMOVE THE REEL AND SAVE IT. IF THE FILE IS NOT A DISK FILE, THE UNIT IS MADE INACCESSIBLE TO THE SYSTEM UNTIL THE OPERATOR RESETS IT AGAIN MANUALLY. IF THE FILE IS A DISK FILE, IT IS RETAINED AS A PERMANENT FILE ON DISK. THE FILE BUFFER AREAS ARE RETURNED TO THE SYSTEM.

THE "LOCK (<FILE DESIGNATOR> , *)" CONSTRUCT CAUSES AN ADJUSTMENT OF THE BLOCKING OF A DISK FILE TO ELIMINATE UNUSED SPACE IN THE DISK AREAS ALLOCATED FOR THE FILE. AS A RESULT OF THE ADJUSTMENT OF THE NUMBER AND SIZE OF PHYSICAL AREAS ON THE DISK, THE "CRUNCHED" FILE MAY NOT BE EXPANDED WITHOUT BEING COPIED INTO A NEW FILE.

EXAMPLES:

```
LOCK (FILEA)
LOCK (FILE, *)
```

8.7.3. READ STATEMENT

SYNTAX:

```
<READ STATEMENT> ::= READ (<FILE PART>, <WORD COUNT>,
      <I/O AREA>) <I/O FINISH EVENT>
<FILE PART> ::= <FILE DESIGNATOR> /
      <FILE DESIGNATOR> <RECORD NUMBER OR CARRIAGE CONTROL>
<RECORD NUMBER OR CARRIAGE CONTROL> ::= [<ARITHMETIC EXPRESSION>] /
      [LINE <ARITHMETIC EXPRESSION>] / [SPACE
      <ARITHMETIC EXPRESSION>] / [SKIP <ARITHMETIC EXPRESSION>] /
      [STACKER <ARITHMETIC EXPRESSION>] / [NO]
<I/O AREA> ::= <ARRAY ROW> / <POINTER EXPRESSION> /
      <SUBSCRIPTED VARIABLE>
<I/O FINISH EVENT> ::= <EMPTY> / [<EVENT DESIGNATOR>]
```

SEMANTICS:

THE <READ STATEMENT> CAUSES DATA TO BE TRANSFERRED FROM A PERIPHERAL DEVICE TO AN <I/O AREA>. THE INPUT DATA IS PROCESSED AS FULL WORDS AND IS ASSIGNED TO THE ELEMENTS OF THE SPECIFIED ARRAY WITHOUT EDITTING.

EXAMPLES:

```
READ (FILEID, 14, POINTERID);
BOOLVAR := READ (FILEID, WORDCOUNT, A[*]);
READ (FILEID, AMOUNT, ARRAYID) [COMPLETE]
```

<FILE PART> INDICATES WHERE THE DATA IS TO BE FOUND.

IF <RECORD NUMBER OR CARRIAGE CONTROL> IS AN <ARITHMETIC EXPRESSION>, ITS VALUE INDICATES THE RELATIVE ADDRESS OF THE RECORD

IN A FILE WHICH IS TO BE READ. THE RECORD POINTER IS SET TO THE SPECIFIED ADDRESS BEFORE THE "READ" IS PERFORMED. IF AN ADDRESS IS INDICATED AND THE FILE IS A RANDOM ACCESS FILE, THE RECORD POINTER IS NOT ADJUSTED AFTER THE READ.

IF THE <RECORD NUMBER OR CARRIAGE CONTROL> IS [NO], THE FILES BUFFER IS NOT RELEASED AFTER IT HAS BEEN READ.

IF THE <RECORD NUMBER OR CARRIAGE CONTROL> IS <EMPTY>, THE RECORD INDICATED BY THE RECORD POINTER IS READ. IF THE FILE IS SERIAL, THE RECORD POINTER IS ADJUSTED TO POINT TO THE NEXT RECORD IN THE FILE.

IF THE <RECORD NUMBER OR CARRIAGE CONTROL> IS "[SPACE <ARITHMETIC EXPRESSION>]" THE NUMBER OF RECORDS SPECIFIED IN <ARITHMETIC EXPRESSION> IS SKIPPED. SPACING IS FORWARD IF THE <ARITHMETIC EXPRESSION> IS POSITIVE; BACKWARD IF NEGATIVE.

THE NUMBER OF WORDS READ IS THE SMALLEST OF:

A. THE NUMBER OF ELEMENTS IN THE <ARRAY ROW>, OR ITEM REFERENCED BY THE <POINTER EXPRESSION>,

B. THE MAXIMUM RECORD LENGTH ("MAXRECSIZE"),

C. THE ABSOLUTE VALUE OF THE <WORD COUNT>.

(NOTE THAT IF THE UNITS ATTRIBUTE=1, AND INTMODE ≠ 0, THEN ALL COUNTS REPRESENT CHARACTERS, NOT WORDS.)

THE "[<EVENT DESIGNATOR>]" FORM OF <I/O FINISH EVENT> MAY BE USED ONLY FOR DIRECT I/O; THE EVENT IS CAUSED WHEN THE I/O OPERATION IS FINISHED.

8.7.4. REWIND STATEMENT

SYNTAX:

<REWIND STATEMENT> ::= REWIND (<FILE DESIGNATOR>)

SEMANTICS:

THE <REWIND STATEMENT> CAUSES THE REFERENCED FILE TO BE CLOSED. IF THE FILE IS A PAPER TAPE OR MAGNETIC TAPE FILE, IT IS REWOUND. FOR DISK FILES, THE RECORD POINTER IS RESET TO THE FIRST RECORD OF THE FILE. THE I/O UNIT WILL REMAIN UNDER PROGRAM CONTROL.

EXAMPLES:

```
REWIND (FILEA)
```

RESTRICTION:

ON PAPER TAPE FILES, THE <REWIND STATEMENT> MAY BE USED ONLY ON INPUT.

8.7.5. SEEK STATEMENT

SYNTAX:

<SEEK STATEMENT> ::= SEEK (<FILE DESIGNATOR> [<RECORD NUMBER>])
<RECORD NUMBER> ::= <ARITHMETIC EXPRESSION>

SEMANTICS:

THE <SEEK STATEMENT> IS USED WITH DISK FILES. IT PROVIDES THE MEANS BY WHICH A FILE"S BUFFER MAY BE FILLED IN ADVANCE OF AN ANTICIPATED READ OR WRITE ON THE RECORD TO WHICH <RECORD NUMBER> POINTS.

THE <FILE DESIGNATOR> MUST NOT BE A DIRECT FILE.

EXAMPLES:

```
SEEK (FILEA [X+2*Y])
```

8.7.6. SPACE STATEMENT

SYNTAX:

<SPACE STATEMENT> ::= SPACE (<FILE DESIGNATOR>,
<ARITHMETIC EXPRESSION>)

SEMANTICS:

THE <SPACE STATEMENT> IS USED TO BYPASS INPUT RECORDS WITHOUT READING THEM. THE VALUE OF THE <ARITHMETIC EXPRESSION> DETERMINES THE NUMBER OF RECORDS TO BE SPACED AND THE DIRECTION OF THE SPACING IF THE <ARITHMETIC EXPRESSION> IS POSITIVE, THE RECORDS ARE SPACED IN A FORWARD DIRECTION; IF NEGATIVE, IN THE REVERSE DIRECTION.

EXAMPLES:

```
SPACE (FID, 1)
BOOLVAR := SPACE (FYLE, - RECS)
SPACE (LINE, LINES - PAGEINX)
```

8.7.7. WRITE STATEMENT

SYNTAX:

<WRITE STATEMENT> ::= WRITE (<FILE PART>, <WORD COUNT>, <I/O AREA>)
<I/O FINISH EVENT>

SEMANTICS:

THE <WRITE STATEMENT> CAUSES DATA TO BE TRANSFERRED FROM AN <I/O AREA> TO A PERIPHERAL DEVICE. THE OUTPUT DATA IS PROCESSED AS FULL WORDS AND IS WRITTEN WITHOUT EDITTING.

<FILE PART> INDICATES WHERE THE DATA IS TO BE WRITTEN.

IF <RECORD NUMBER OR CARRIAGE CONTROL> IS "[LINE <ARITHMETIC EXPRESSION>]" AND THE FILE IS A LINE PRINTER FILE, THEN THE PRINTER WILL SKIP FORWARD TO THE SPECIFIED LINE BEFORE PRINTING. HOWEVER, THE FOLLOWING MUST BE OBSERVED:

1) THE PAGESIZE FILE ATTRIBUTE MUST BE SET TO THE NUMBER OF LINES ON A PAGE.

2) SINCE NORMAL DEFAULT ACTION FOR ALGOL IS PRINT BEFORE CARRIAGE ACTION, A SUBSEQUENT WRITE MAY OVERPRINT THE LINE.

3) THE LINE NUMBER IS NOT RESET WHEN [SKIP 1] IS USED SINCE THIS DOES NOT NECESSARILY EJECT A PAGE. IN THIS CASE, THE USER MUST RESET THE LINENUMBER ATTRIBUTE.

"[SPACE <ARITHMETIC EXPRESSION>]" CAUSES THE LINEPRINTER TO SPACE THE NUMBER OF LINES DENOTED BY THE <ARITHMETIC EXPRESSION> AFTER PRINTING THE CURRENT RECORD. ON OTHER TYPES OF DEVICES IT CAUSES

THE NUMBER OF RECORDS SIGNIFIED BY THE <ARITHMETIC EXPRESSION> TO BE SPACED. "[SKIP <ARITHMETIC EXPRESSION>]" CAUSES THE LINEPRINTER TO SKIP TO THE CHANNEL INDICATED BY THE <ARITHMETIC EXPRESSION>. AFTER PRINTING THE CURRENT RECORD.

IF THE FILE IS NOT A PRINTER FILE, <RECORD NUMBER OR CARRIAGE CONTROL> IS INTERPRETED AS A RECORD NUMBER AS DESCRIBED UNDER <READ STATEMENT>.

IF THE <RECORD NUMBER OR CARRIAGE CONTROL> IS "[STACKER <ARITHMETIC EXPRESSION>]", THE PRIMARY (NORMAL) STACKER IS CHOSEN WITH A VALUE OF ZERO (0), AND THE ALTERNATE (AUXILLIARY) STACKER IS CHOSEN WITH A VALUE OF ONE (1).

THE NUMBER OF WORDS WRITTEN IS THE SMALLEST OF:

A. THE NUMBER OF ELEMENTS IN THE <ARRAY ROW> OR ITEM REFERENCED BY THE <POINTER EXPRESSION>,

B. THE MAXIMUM RECORD LENGTH ("MAXRECSIZE"),

C. THE ABSOLUTE VALUE OF THE <WORD COUNT>.

WHEN UNBLOCKED RECORDS ARE BEING USED, THE BUFFER SIZE IS THE MAXIMUM RECORD LENGTH. (NOTE THAT IF THE UNITS ATTRIBUTE=1, AND INTMODE ≠ 0, THEN ALL COUNTS REPRESENT CHARACTERS NOT WORDS.)

DIRECT I/O BRINGS THE USER CLOSER TO THE ACTUAL INPUT/OUTPUT OPERATION. ESSENTIALLY IT PERMITS I/O OPERATIONS TO BE PERFORMED ASYNCHRONOUSLY - IN PARALLEL WITH PROGRAM EXECUTION - AND ALLOWS THE I/O TIME AND PROCESSING TIME TO BE OVERLAPPED WITHIN A SINGLE PROGRAM.

TO PERFORM DIRECT I/O ON A FILE (LET US CALL IT "FID") ONE MUST FIRST DECLARE THE FILE AS A DIRECT FILE.

THE SYNTAX FOR A DIRECT I/O READ OR WRITE EMPLOYS THE "<ARITHMETIC EXPRESSION>, <ARRAY ROW>" FORM, AND THE "[<EVENT DESIGNATOR>]" CAN BE USED. THE <ARRAY ROW> IS CALLED THE USERS I/O AREA AND THE DIRECT ARRAY IDENTIFIER MUST BE USED IN THE <ARRAY ROW> CONSTRUCT. THUS TO READ DIRECT TEN WORDS FROM FID INTO DIRECT ARRAY "A" USING THE EVENT EVT WE WRITE: "READ(FID,10,A[*])[EVT];". WHEN EXECUTING THIS <STATEMENT>, THE MCP ESTABLISHES A RELATIONSHIP BETWEEN THE I/O AREA AND THE EVENT EVT.

HOWEVER, BEFORE ANY SUBSEQUENT USE OF THE I/O AREA CAN BE MADE IN THE PROGRAM, EITHER FOR CALCULATIONS OR FOR FURTHER I/O, WE MUST BE SURE THAT THE DIRECT I/O OPERATION IS COMPLETE. THE EVENT MECHANISM TELLS US THIS BY SETTING THE "HAPPENED" BIT ASSOCIATED WITH EVT WHEN THE READ OPERATION IS FINISHED. THIS BIT MAY BE INSPECTED BY MEANS OF THE BOOLEAN INTRINSIC FUNCTION "HAPPENED". THE USER MAY ALSO USE A <WAIT STATEMENT> TO DEACTIVATE THE PROCESS UNTIL THE "HAPPENED" BIT IS SET. ONCE THE OPERATION IS COMPLETED, THE "HAPPENED" BIT SHOULD BE RESET BEFORE REUSING EVT.

EXAMPLES:

```
WRITE (LINE, 14, POINTERID)
BOOLVAR := WRITE (FIB, WORDCOUNT, A[*])
WRITE (FYLE, AMOUNT, ARRAYID) [ALLOVER]
```

8.8. INTERRUPT STATEMENTS

SYNTAX:

```
<INTERRUPT STATEMENT> ::= <ATTACH STATEMENT> / <DETACH STATEMENT> /
      <ENABLE STATEMENT> / <DISABLE STATEMENT>
<ATTACH STATEMENT> ::= ATTACH <INTERRUPT IDENTIFIER> TO
      <EVENT DESIGNATOR>
<DETACH STATEMENT> ::=  DETACH <INTERRUPT IDENTIFIER>
<ENABLE STATEMENT> ::= ENABLE <INTERRUPT IDENTIFIER>
<DISABLE STATEMENT> ::=  DISABLE <INTERRUPT IDENTIFIER>
```

SEMANTICS:

THE <ATTACH STATEMENT> IS USED TO ASSOCIATE AN INTERRUPT WITH AN EVENT. ATTACHING AN INTERRUPT TO AN EVENT WITH AN <ATTACH STATEMENT> SERVES TO IMPLICITLY ENABLE THE INTERRUPT. AN INTERRUPT MUST BE "ENABLED" AND "ATTACHED" TO AN EVENT BEFORE IT CAN HAVE ANY EFFECT.

AN INTERRUPT MAY NOT BE ATTACHED TO MORE THAN ONE EVENT AT A TIME. HOWEVER, SEVERAL INTERRUPTS MAY BE ATTACHED TO THE SAME EVENT AT THE SAME TIME.

THE <DETACH STATEMENT> SEVERS THE ASSOCIATION BETWEEN AN INTERRUPT AND THE EVENT TO WHICH IT IS ATTACHED. THIS STATEMENT IMPLICITLY DISABLES THE INTERRUPT.

THE <ENABLE STATEMENT> AND THE <DISABLE STATEMENT> ARE USED TO EXPLICITLY ENABLE OR DISABLE AN INTERRUPT. THEY DIFFER FROM THE <ATTACH STATEMENT> AND <DETACH STATEMENT> IN THAT THEY DO NOT AFFECT THE ASSOCIATION BETWEEN THE INTERRUPT AND THE EVENT.

EXAMPLES:

```
ATTACH I1 TO EVT;
DISABLE I1;
ENABLE I1;
DETACH I1;
```

8.9. ON STATEMENT

SYNTAX:

```
<ON STATEMENT> ::= ON <FAULT NAME> /
      ON <FAULT NAME> , <STATEMENT>
<FAULT NAME> ::= ANYFAULT / BOTTOMOFSTACK /
      EXPONENTOVERFLOW / EXPONENTUNDERFLOW / INACTIVEQUEUE /
      INTEGEROVERFLOW / INVALIDADDRESS / INVALIDINDEX /
      INVALIDOP / INVALIDPROGRAMWORD / LOOP / MEMORYPARITY /
      MEMORYPROTECT / PROGRAMMEDOPERATOR / SCANPARITY /
      SEQUENCE / STACKOVERFLOW / STACKUNDERFLOW /
      STRINGPROTECT / ZERODIVIDE
```

SEMANTICS:

THE <ON STATEMENT> IS USED TO ENABLE (OR DISABLE) AN INTERRUPT FOR ONE OF 19 FAULT CONDITIONS. "ANYFAULT" SPECIFIES THAT ANY ONE OF THE 19 FAULTS INVOKES THE <STATEMENT>.

"ON <FAULT NAME>,<STATEMENT>" ENABLES THAT <STATEMENT> AS THE INTERRUPT CODE. "ON <FAULT NAME>" DISABLES ANY PREVIOUSLY ENABLED INTERRUPT FOR THAT FAULT.

ONCE ENABLED, THE INTERRUPT REMAINS ENABLED UNTIL:

A. THE PROCEDURE OR <BLOCK> CONTAINING THE <ON STATEMENT> IS EXITED;

B. THE INTERRUPT IS EXPLICITLY DISABLED;

C. A NEW INTERRUPT IS ENABLED FOR THE SAME FAULT CONDITION.

WHENEVER AN EXIT IS MADE FROM THE <BLOCK> CONTAINING AN <ON

STATEMENT>, THE INTERRUPT STATUS FOR THAT FAULT CONDITION REVERTS TO WHATEVER WAS ENABLED WHEN THE <BLOCK> WAS ENTERED.

THE INTERRUPT <STATEMENT> MUST EITHER BE AN UNCONDITIONAL GO TO (OR A <COMPOUND STATEMENT> OR <BLOCK> CONTAINING AN UNCONDITIONAL GO TO), SINCE THE INTERRUPTED CODE CANNOT BE RESUMED.

UPON ENTRY INTO THE <STATEMENT>, THE TOP OF THE STACK CONTAINS A "FAULT NUMBER" WHICH SPECIFIES THE NATURE OF THE FAULT. (SEE THE "TOPOFSTACK" INTRINSIC.) THIS IS A PARTICULAR IMPORTANCE IF "ANYFAULT" IS USED. THE VALUES AND MEANINGS ARE AS FOLLOWS:

1	ZERODIVIDE
2	EXPONENTOVERFLOW
3	EXPONENTUNDERFLOW
4	INVALIDINDEX
5	INTEGEROVERFLOW
6	INACTIVEQUEUE
7	MEMORYPROTECT
8	INVALIDOP
9	LOOP
10	MEMORYPARITY
11	SCANPARITY
12	INVALIDADDRESS
13	STACKOVERFLOW
14	STRINGPROTECT
15	PROGRAMMEDOPERATOR
16	BOTTOMOFSTACK
17	SEQUENCE
18	INVALIDPROGRAMWORD
19	STACKUNDERFLOW

8.10. PROCEDURE STATEMENTS AND FUNCTION DESIGNATORS

SYNTAX:

```
<PROCEDURE STATEMENT> ::= <PROCEDURE IDENTIFIER>
      <ACTUAL PARAMETER PART>
<FUNCTION DESIGNATOR> ::= <FUNCTION IDENTIFIER>
      <ACTUAL PARAMETER PART>
<FUNCTION IDENTIFIER> ::= <IDENTIFIER>
<ACTUAL PARAMETER PART> ::= <EMPTY> / (<ACTUAL PARAMETER LIST>)
<ACTUAL PARAMETER LIST> ::= <ACTUAL PARAMETER> /
      <ACTUAL PARAMETER LIST> <PARAMETER DELIMITER>
      <ACTUAL PARAMETER>
<ACTUAL PARAMETER> ::= <EXPRESSION> / <PROCEDURE IDENTIFIER> /
      <EVENT DESIGNATOR>
<PARAMETER DELIMITER> ::= , /
      )"⊿ANY SEQUENCE OF LETTERS AND SINGLE SPACES⊾"(
```

SEMANTICS:

1. A PROCEDURE STATEMENT CAUSES A PREVIOUSLY DECLARED PROCEDURE TO BE EXECUTED.

2. A FUNCTION DESIGNATOR RETURNS A VALUE. HOWEVER, WHEN A FUNCTION DESIGNATOR IS USED AS A PROCEDURE STATEMENT, THIS VALUE IS DELETED.

3. THE ACTUAL PARAMETER LIST OF THE PROCEDURE STATEMENT MUST HAVE THE SAME NUMBER OF ENTRIES AS THE FORMAL PARAMETER LIST OF THE PROCEDURE DECLARATION HEADING.

4. FORMAL AND ACTUAL PARAMETERS MUST CORRESPOND IN TYPE AND KIND OF QUANTITIES. THE CORRESPONDENCE IS OBTAINED BY TAKING THE ENTRIES OF THESE TWO LISTS IN THE SAME ORDER.

5. THE FOLLOWING REMARKS CONCERN PARAMETERS:

A. CONSTANTS ARE PASSED BY VALUE, NO MATTER HOW THE CORRESPONDING FORMAL PARAMETER IS SPECIFIED.

B. SUBSCRIPTED VARIABLES, WHEN PASSED BY NAME, WILL NOT CAUSE ACCIDENTAL ENTRY, EVEN IF THE SUBSCRIPTS ARE NOT CONSTANT.

C. AS A RESULT OF B, ABOVE, ANY CALL-BY-NAME PARAMETER MAY NOT BEGIN WITH A SUBSCRIPTED VARIABLE UNLESS IT IS JUST A SUBSCRIPTED VARIABLE. (NOTE: PARENTHESES AROUND THIS VARIABLE WILL REMEDY THIS.)

D. SUBEXPRESSIONS OF THE FORM X+X ARE OPTIMIZED TO THE FOLLOWING:

```
VALC X
DUPL
ADD
```

EVEN IF X IS A FORMAL CALL-BY-NAME PARAMETER WITH SIDE EFFECTS.

EXAMPLES:

PROCEDURE STATEMENTS:

```
ALGORITHM123 (A+2)
ALGORITHM546 (A+2)"AVERAGE PLUS TWO"(CALCRULE)
GETESPDISK
```

FUNCTION DESIGNATORS:

```
J(A,B+2,Q[I,L])
GASVOL(K)"TEMPERATURE"(T)"PRESSURE"(P)
RANDOMNO
```

ACTUAL PARAMETER PARTS:

```
(A,B+2,Q[I,J])
(A+2)
(K)"TEMPERATURE"(T)"PRESSURE"(P)
```

ACTUAL PARAMETERS:

```
A+2
A
CHECKOUT
A[2]
```

EVENT DESIGNATORS:

```
E1
XERXES
ZEUS[2,3]
```

PARAMETER DELIMITER:

```
)"TEMPERATURE"(
```

8.11. REPLACE AND SCAN STATEMENTS

SYNTAX:

```
<REPLACE STATEMENT> ::= REPLACE <DESTINATION> BY <SOURCE LIST>
<SCAN STATEMENT> ::= SCAN <SOURCE> <SCAN PART>
<DESTINATION> ::= <UPDATE POINTER> <POINTER EXPRESSION>
<SOURCE LIST> ::= <SOURCE PART> / <SOURCE PART> , <SOURCE LIST>
<SOURCE> ::= <POINTER SOURCE> / <ARITHMETIC SOURCE>
<SCAN PART> ::= <SCAN COUNT> <CONDITION> / <CONDITION>
<SOURCE PART> ::= <SOURCE> <TRANSFER PART> / <STRING>
      <TRANSFER PART> / <ARITHMETIC SOURCE>
      <ARITHMETIC TRANSFER PART> / <STRING>
<TRANSFER PART> ::= <SCAN COUNT> <CONDITION> / <CONDITION>
      <FINAL COUNT> <UNITS> / WITH <PICTURE DESIGNATOR> /
      <FINAL COUNT> WITH <TRANSLATE TABLE>
<ARITHMETIC TRANSFER PART> ::= <DIGIT COUNT> / <CORRECT COUNT> /
      <CORRECT COUNT> <CONDITION>
<SCAN COUNT> ::= FOR <UPDATE COUNT> <ARITHMETIC EXPRESSION>
<FINAL COUNT> ::= FOR <ARITHMETIC EXPRESSION>
<DIGIT COUNT> ::= FOR <ARITHMETIC EXPRESSION> DIGITS
<CORRECT COUNT> ::= <SCAN COUNT> CORRECTLY
<POINTER SOURCE> ::= <UPDATE POINTER> <POINTER EXPRESSION>
<ARITHMETIC SOURCE> ::= <UPDATE VARIABLE> <ARITHMETIC EXPRESSION>
<CONDITION> ::= WHILE <RELATIONAL OPERATOR>
      <ARITHMETIC EXPRESSION> / UNTIL <RELATIONAL OPERATOR>
      <ARITHMETIC EXPRESSION> / UNTIL IN <TABLE> / WHILE IN <TABLE>
<TABLE> ::= <ARRAY ROW> / <SUBSCRIPTED VARIABLE>
<TRANSLATE TABLE> ::= <TABLE>
<PICTURE DESIGNATOR> ::= <PICTURE IDENTIFIER> <REPEAT PARAMETERS>
<REPEAT PARAMETERS> ::= <EMPTY> / (<UNSIGNED INTEGER LIST>)
<UNSIGNED INTEGER LIST> ::= <ARITHMETIC EXPRESSION> /
      <ARITHMETIC EXPRESSION> , <UNSIGNED INTEGER LIST>
<UPDATE POINTER> ::= <EMPTY> / <POINTER IDENTIFIER> :
<UPDATE VARIABLE> ::= <EMPTY> / <SIMPLE VARIABLE> :
```

<UPDATE COUNT> ::= <EMPTY> / <SIMPLE VARIABLE> :
<UNITS> ::= <EMPTY> / WORDS / OVERWRITE

SEMANTICS:

IN THE FOLLOWING DISCUSSION, THE TERMS "REPLACE" AND "TRANSFER" ARE SYNONYMOUS.

1. A <REPLACE STATEMENT> CAUSES INFORMATION TO BE TRANSFERRED FROM THE <SOURCE> TO THE <DESTINATION>. THE UNIT OF INFORMATION MAY BE WORDS OR CHARACTERS, AND THE NUMBER OF UNITS TRANSFERRED MAY BE SPECIFIED BY A COUNT OR DETERMINED BY A <CONDITION>. IF THE UNIT OF INFORMATION IS A WORD, THEN BOTH THE <SOURCE> AND THE <DESTINATION> WILL BE RIGHT-ADJUSTED TO A WORD BOUNDARY BEFORE THE TRANSFER BEGINS.

2. A <SCAN STATEMENT> CAUSES THE INFORMATION IN THE SOURCE TO BE SCANNED. THE UNIT OF INFORMATION IS A CHARACTER, AND A <CONDITION> MUST BE PRESENT. THE NUMBER OF CHARACTERS SCANNED MAY BE DETERMINED BY THE COUNT OR THE <CONDITION>.

3. THE DEFAULT UNIT FOR THE <REPLACE STATEMENT> IS CHARACTERS. "WORDS" AND "OVERWRITE" BOTH MEAN UNITS OF ONE WORD. IN ADDITION, "OVERWRITE" IGNORES MEMORY PROTECTION ON THE DESTINATION AND THE SOURCE, AND IT ALSO IGNORES MEMORY PARITY ERRORS ON THE DESTINATION.

4. AT THE END OF A REPLACE OR SCAN STATEMENT, CERTAIN UPDATED INFORMATION IS AVAILABLE.

    A. THE DESTINATION <UPDATE POINTER> POINTS TO THE NEXT POSITION TO BE FILLED. IF THE UNIT OF TRANSFER IS A WORD, THE DESTINATION <UPDATE POINTER> WILL POINT TO THE LEFTMOST CHARACTER OF THE NEXT WORD TO BE FILLED

    B. THE SOURCE <UPDATE POINTER> POINTS TO THE NEXT UNIT

TO BE SCANNED OR TRANSFERRED. IF THE UNIT IS A WORD, THE <UPDATE POINTER> WILL POINT TO THE LEFTMOST CHARACTER OF THE WORD.

C. FOR SCANS AND TRANSFERS, AN <ARITHMETIC SOURCE> SHOULD BE THOUGHT OF AS BEING CIRCULAR, WITH THE HIGH-ORDER AND LOW-ORDER ENDS CONTIGUOUS. THE SOURCE <UPDATE VARIABLE> RETURNS THE ORIGINAL EXPRESSION ROTATED IN SUCH A WAY THAT THE NEXT CHARACTER TO BE USED IS IN THE HIGH-ORDER POSITION.

D. EACH TIME A UNIT OF INFORMATION IS SCANNED OR TRANSFERRED, THE ORIGINAL COUNT AS GIVEN BY AN <ARITHMETIC EXPRESSION> IS DECREMENTED BY 1. THIS CONINUES UNTIL THE COUNT REACHES 0 IF NO <CONDITION> IS IMPOSED. IF A <CONDITION> IS IMPOSED, THE COUNT MAY NOT REACH 0 AND THE <UPDATE COUNT> RETURNS THE VALUE OF THE COUNT AT THE END OF THE TRANSFER. THE RESERVED WORD "TOGGLE" WILL BE TRUE IF THE <UPDATE COUNT> IS 0.

5. THE <DIGIT COUNT> CAUSES THE SOURCE <ARITHMETIC EXPRESSION> THE DESIGNATED NUMBER OF LOW-ORDER DECIMAL DIGITS IS TRANSFERRED TO THE <DESTINATION>, AND THE SOURCE <UPDATE VARIABLE> RETURNS THE ORIGINAL EXPRESSION "DIV" THE DESIGNATED POWER OF 10.

6. THE <CORRECT COUNT> CAUSES THE SOURCE <ARITHMETIC EXPRESSION> TO BE ROTATED IN SUCH A WAY THAT THE APPROPRIATE NUMBER OF LOW-ORDER CHRACTERS APPEAR IN THE HIGH-ORDER ORDER END OF THE SOURCE. AFTER THIS ROTATION, A NORMAL TRANSFER WILL OCCUR. IF, FOR EXAMPLE, ONE CHARACTER WERE TO BE TRANSFERRED CORRECTLY, THE LOW-ORDER CHARACTER WOULD BE MOVED TO THE HIGH-ORDER POSITION BEFORE THE TRANSFER OCCURS. SINCE TRANSFERS WORK FROM LEFT TO RIGHT, THIS HAS THE EFFECT OF ALLOWING THE TRANSFER OF RIGHT-JUSTIFIED CHARACTERS IN AN <ARITHMETIC

EXPRESSION>.

7. SCANS AND TRANSFERS ALWAYS WORK FROM LEFT TO RIGHT ON BOTH <DESTINATION> AND <SOURCE>, INCLUDING <ARITHMETIC SOURCE>. THE <CORRECT COUNT> AND <DIGIT COUNT> MAY BE CONSIDERED EXCEPTIONS.

8. THE <TRANSLATE TABLE> WORKS IN THE FOLLOWING WAY. EACH SOURCE CHARACTER IS USED TO FIND AN EIGHT-BIT TRANSLATION CHARACTER IN AN <ARRAY ROW>. THE HIGH-ORDER PART OF THIS CHARACTER IS DISCARDED TO MAKE IT FIT THE <DESTINATION> CHARACTER SET, AND IT IS THEN STORED IN THE DESTINATION, THE EIGHT-BIT TRANSLATION CHARACTER IS FOUND IN THE FOLLOWING WAY. THE LOW-ORDER TWO BITS OF THE SOURCE CHARACTER ARE USED AS THE CHARACTER INDEX, AND THE REMAINING HIGH-ORDER BITS ARE USED AS THE WORD INDEX. THE WORD INDEX IS EITHER USED AS A SUBSCRIPT TO THE <ARRAY ROW> OR IS ADDED TO THE RIGHTMOST SUBSCRIPT OF THE <SUBSCRIPTED VARIABLE>. IN THE RESULTING WORD, CHARACTER NUMBER (2+ (CHARACTER INDEX)) IS THE TRANSLATION CHARACTER. AS USUAL, THE CHARACTERS ARE NUMBERED LEFT TO RIGHT, SO THAT THE TRANSLATION CHARACTER WILL BE ONE OF THE FOUR LOW-ORDER EIGHT-BIT CHARACTERS IN THE WORD.

9. THE WHILE <RELATIONAL OPERATOR> FORM OF <CONDITION> CAUSES TERMINATION OF THE SCAN OR TRANSFER WHEN THE SOURCE CHARACTER CEASES TO HAVE THE DESIGNATED RELATION TO THE LOW-ORDER CHARACTER OF THE CONDITION <ARITHMETIC EXPRESSION>. THE CHARACTER WHICH CAUSES TERMINATION OF THE SCAN OR TRANSFER WILL NOT BE SCANNED OR TRANSFERRED, THUS THE SOURCE <UPDATE POINTER> IS LEFT POINTING AT THAT CHARACTER.

10. THE UNTIL <RELATIONAL OPERATOR> FORM OF <CONDITION> CAUSES TERMINATION OF THE SCAN OR TRANSFER WHEN THE SOURCE CHARACTER HAS THE DESIGNATED RELATION TO THE LOW-ORDER CHARACTER OF THE CONDITION <ARITHMETIC EXPRESSION>. THE CHARACTER WHICH CAUSES TERMINATION OF THE SCAN OR TRANSFER WILL NOT BE SCANNED OR

TRANSFERRED.

11. <CONDITION>S INVOLVING THE <TABLE> CONSTRUCT USE THE BITS OF THE SOURCE CHARACTER TO FIND A TEST BIT IN AN <ARRAY ROW>. THE CHARACTER IS "IN" THE <TABLE> IFF THE TEST BIT IS ON. THE TEST BIT IS FOUND IN THE FOLLOWING WAY. THE LOW-ORDER FIVE BITS OF THE SOURCE CHARACTER (FOUR BITS IF THE SOURCE CHARACTER ONLY HAS FOUR) ARE USED AS THE BIT INDEX, AND THE REMAINING BITS, IF ANY, ARE USED AS THE WORD INDEX. THE WORD INDEX IS EITHER USED AS A SUBSCRIPT TO THE <ARRAY ROW> OR IS ADDED TO THE RIGHTMOST SUBSCRIPT OF THE <SUBSCRIPTED VARIABLE> IN THE RESULTING WORD, BIT NUMBER (31 - (BIT INDEX)) IS THE TEST BIT. AS USUAL, BITS ARE NUMBERED RIGHT TO LEFT, SO THAT THE TEST BIT WILL BE ONE OF THE LOW-ORDER 32 BITS IN THE WORD.

12. IF THE <SOURCE> IS A <STRING>, THE <STRING> IS TRANSFERRED TO THE <DESTINATION> UNDER THE CONTROL OF A COUNT. IF THE COUNT, C, IS NOT GREATER THAT THE STRING LENGTH, L, THEN C CHARACTERS ARE COPIED INTO THE DESTINATION. IF THE COUNT, C, IS GREATER THAN THE LENGTH, L, AND THE STRING IS MORE THAN 48 BITS LONG, THE BEHAVIOR IS UNPREDICTABLE. IF C IS GREATER THAN L AND THE STRING IS SHORTER THAN 48 BITS, THEN THE STRING IS CONCATENATED WITH ITSELF UNTIL THE TOTAL LENGTH IS GREATER THAN 48 BITS. THE FIRST 48 BITS OF THIS STRING ARE USED AS A SOURCE AND ARE REPEATEDLY COPIED INTO THE DESTINATION UNTIL THE COUNT IS EXHAUSTED. FOR EXAMPLE, 8 "ABCD" FOR 10 TRANSFERS "ABCDABABCD", NOT "ABCDABCDAB".

13. <SOURCE> WITH <PICTURE IDENTIFIER> TRANSFERS CHARACTERS UNDER THE CONTROL OF THE PICTURE. (SEE <PICTURE DECLARATION>.)

9. EXPRESSIONS

9.1. GENERAL

AN EXPRESSION IS USED TO OBTAIN VALUES.

SYNTAX:

<EXPRESSION> ::= <GENERAL COMPONENT> / <ARITHMETIC EXPRESSION> /
<ARRAY EXPRESSION> / <BOOLEAN EXPRESSION> /
<CASE EXPRESSION> / <CONDITIONAL EXPRESSION> /
<DESIGNATIONAL EXPRESSION> / <POINTER EXPRESSION> /
<REFERENCE EXPRESSION> / <WORD EXPRESSION>

SEMANTICS:

EXPRESSIONS ARE RULES BY WHICH VALUES MAY BE OBTAINED BY EXECUTING VARIOUS OPERATIONS ON THE COMPONENTS OF WHICH EXPRESSIONS ARE COMPOSED. CONDITIONAL AND CASE EXPRESSIONS ARE MORE COMPLICATED BECAUSE ONE OF SEVERAL ALTERNATIVE EXPRESSIONS MUST FIRST BE SELECTED FOR EVALUATION.

9.2. ARITHMETIC EXPRESSIONS

SYNTAX:

```
<ARITHMETIC EXPRESSION> ::= <CONDITIONAL ARITHMETIC EXPRESSION> /
      <ARITHMETIC ASSIGNMENT> / <SIMPLE ARITHMETIC EXPRESSION> /
      <WORD EXPRESSION>
<CONDITIONAL ARITHMETIC EXPRESSION> ::= <IF CLAUSE>
      <ARITHMETIC EXPRESSION> ELSE <ARITHMETIC EXPRESSION>
<ARITHMETIC ASSIGNMENT> ::=
      <ARITHMETIC VARIABLE> <REPLACEMENT OPERATOR>
      <ARITHMETIC EXPRESSION>
<SIMPLE ARITHMETIC EXPRESSION> ::=
      <SIMPLE ARITHMETIC EXPRESSION> <ADDING OPERATOR> <TERM> /
      <UNARY OPERATOR> <TERM> / <TERM>
<ADDING OPERATOR> ::= + / -
<UNARY OPERATOR> ::= <ADDING OPERATOR>
<TERM> ::= <FACTOR> / <TERM> <MULTIPLYING OPERATOR> <FACTOR>
<FACTOR> ::= <PRIMARY> / <PRIMARY> * <INTEGER>
<MULTIPLYING OPERATOR> ::= x / <SLASH> / DIV / MOD / MUX
<PRIMARY> ::= <UNSIGNED NUMBER> / <STRING> /
      <FIELD DESIGNATOR> / <OPERAND> / <PRIMARY> &
      <ARITHMETIC EXPRESSION> <CONCATENATION> / <PRIMARY>
      & <LAYOUT>
<FIELD DESIGNATOR> ::= <OPERAND> . <FIELD IDENTIFIER> /
      <ARRAY PRIMARY> . <FIELD IDENTIFIER>
<OPERAND> ::= <ARITHMETIC VARIABLE>
      (<ARITHMETIC EXPRESSION>) /
      <ARITHMETIC FUNCTION DESIGNATOR> /
      <CASE HEAD> (<ARITHMETIC EXPRESSION LIST>)
<CONCATENATION> ::= [<LEFT BIT-TO>:<NUMBER OF BITS>] /
      [<LEFT BIT-TO>:<LEFT BIT-FROM>:<NUMBER OF BITS>]
<LEFT BIT-TO> ::= <ARITHMETIC EXPRESSION>
<NUMBER OF BITS> ::= <ARITHMETIC EXPRESSION>
<LEFT BIT-FROM> ::= <ARITHMETIC EXPRESSION>
```

<LAYOUT> ::= <LAYOUT IDENTIFIER> (<FIELD VALUE LIST>)

<FIELD VALUE LIST> ::= <FIELD VALUE> / <FIELD VALUE LIST>,
<FIELD VALUE>

<FIELD VALUE> ::= <EMPTY> / <ARITHMETIC EXPRESSION> / *

<ARITHMETIC VARIABLE> ::= <VARIABLE>

<ARITHMETIC FUNCTION DESIGNATOR> ::= <FUNCTION DESIGNATOR> /
<ARITHMETIC INTRINSIC>

<ARITHMETIC EXPRESSION LIST> ::= <ARITHMETIC EXPRESSION> /
<ARITHMETIC EXPRESSION LIST> , <ARITHMETIC EXPRESSION>

SEMANTICS:

1. AN ARITHMETIC EXPRESSION OBTAINS A NUMERIC VALUE.

2. A VARIABLE, VALUE DESIGNATOR, OR FUNCTION DESIGNATOR USED AS A PRIMARY IN AN ARITHMETIC EXPRESSION MUST BE OF AN ARITHMETIC TYPE: INTEGER, REAL, OR DOUBLE.

3. EACH EXPRESSION IN A FIELD VALUE LIST OR EXPRESSION LIST USED IN AN ARITHMETIC EXPRESSION MUST ALSO BE OF AN ARITHMETIC TYPE

4. THE VALUE OF AN ARITHMETIC EXPRESSION MAY BE EXPRESSED IN SINGLE OR DOUBLE PRECISION.

    A. THE PRECISION OF A CASE EXPRESSION VALUE IS DOUBLE IF ANY ONE OF THE EXPRESSIONS OF ITS EXPRESSION LIST IS DOUBLE. OTHERWISE THE PRECISION IS SINGLE.

    B. THE PRECISION OF THE VALUE OF A CONDITIONAL ARITHMETIC EXPRESSION IS DOUBLE IF ANY OF THE EXPRESSIONS IN THE STATEMENT IS DOUBLE PRECISION, OTHERWISE THE PRECISION OF THE VALUE IS SINGLE.

    C. EXTENDED PRECISION VALUES MAY NOT BE USED IN A FIELD DESIGNATOR AS A BASE.

5. THE OPERATOR "DIV" DENOTES INTEGER DIVISION. Y DIV Z = SIGN (Y/Z)xENTIER(ABS(Y/Z))

6. THE OPERATOR "MOD" DENOTES REMAINDER DIVISION. Y MOD Z = Y - (Zx(SIGN(Y/Z)x ENTIER (ABS(Y/Z))))

7. THE OPERATOR "MUX" DENOTES DOUBLE-PRECISION MULTIPLICATION. ITS OPERANDS MAY BE SINGLE OR DOUBLE PRECISION.

8. THE EXPONENTIATION OPERATOR, * , IS DEFINED FOR INTEGER-CONSTANT EXPONENTS ONLY.

9. THE SEQUENCE IN WHICH OPERATIONS ARE PERFORMED IS DETERMINED BY THE PRECEDENCE OF THE OPERATORS.

10. THE ORDER OF PRECEDENCE OF OPERATORS IS:

FIRST: *
SECOND: x, /, MOD, DIV, MUX
THIRD: +, -

11. WHEN OPERATORS ARE OF THE SAME ORDER OF PRECEDENCE, THE SEQUENCE OF OPERATION IS DETERMINED FROM THE LEFT-TO-RIGHT ORDER OF APPEARANCE OF THE OPERATORS.

12. AN EXPRESSION BETWEEN PARENTHESES IS EVALUATED BY ITSELF AND THIS VALUE IS USED IN SUBSEQUENT CALCULATIONS. THAT IS, THE NORMAL ORDER OF PRECEDENCE OF OPERATORS CAN BE OVERRIDDEN BY THE JUDICIOUS PLACEMENT OF PARENTHESES. THEREFORE, THE DESIRED ORDER OF EXECUTION WITHIN AN EXPRESSION CAN ALWAYS BE ARRANGED BY THE APPROPRIATE POSITIONING OF PARENTHESES.

13. NO TWO OPERATORS MAY BE ADJACENT.

14. THE CONCATENATION FORM OF <ARITHMETIC EXPRESSION> PROVIDES AN EFFICIENT METHOD OF FORMING A <PRIMARY> FROM

SELECTED PARTS OF TWO OR MORE <PRIMARY>S. A CONCATENATION <PRIMARY> IS FORMED BY LINKING PART OF A <PRIMARY> WITH THE SPECIFIED PORTION OF AN <ARITHMETIC EXPRESSION> VALUE. SINCE <ARITHMETIC EXPRESSION> IS RECURSIVE WITH RESPECT TO CONCATENATION, ANY NUMBER OF <CONCATENATION> TERMS MAY BE USED IN CONSTRUCTING A <PRIMARY>.

THE <LEFT BIT-TO> PART OF A <CONCATENATION> TERM DEFINES THE LEFTMOST BIT LOCATION OF THE DATA FIELD IN THE DESTINATION WORD. THE <LEFT BIT-FROM> PART DEFINES THE LEFTMOST BIT LOCATION OF THE DATA FIELD IN THE SOURCE WORD. THE <NUMBER OF BITS> PART SPECIFIES THE LENGTH OF THE DATA FIELD TO BE MOVED FROM THE SOURCE FIELD TO THE DESTINATION FIELD.

IF THE <LEFT BIT-TO> : <NUMBER OF BITS> FORM OF <CONCATENATION> IS SPECIFIED, THE SOURCE FIELD IS ASSUMED TO START AT <NUMBER OF BITS>-1, THAT IS, THE SOURCE FIELD IS ASSUMED TO BE THE LOW-ORDER <NUMBER OF BITS> IN THE SOURCE WORD.

IF MORE THAN ONE <CONCATENATION> TERM IS USED IN AN <EXPRESSION>, THEN THESE ARE EVALUATED FROM LEFT TO RIGHT.

A <FIELD DESIGNATOR> ALLOWS OPERATIONS TO BE PERFORMED ON ANY CONTIGUOUS FIELD WITHIN A WORD RATHER THAN THE WHOLE WORD. THE <LEFT BIT-FROM> PART DEFINES THE LEFTMOST BIT LOCATION OF THE FIELD. THE <NUMBER OF BITS> PART SPECIFIES THE LENGTH OF THE FIELD.

THE <LEFT BIT-FROM> PARTS OF <CONCATENATION> AND <FIELD DESIGNATOR>S MUST LIE IN THE RANGE OF 0 THRU 47, WHERE BIT 0 IS THE RIGHTMOST (OR LEAST SIGNIFICANT) BIT IN THE WORD. <NUMBER OF BITS> MUST LIE IN THE RANGE 0 THROUGH 48. IF IT EXCEEDS THE NUMBER OF BITS REMAINING IN EITHER THE SOURCE OR DESTINATION WORDS, THESE FIELDS ARE CONTINUED AT BIT NUMBER 47 (LEFTMOST) OF THE SAME WORD.

15. AN EMPTY FIELD VALUE CAUSES THE DEFAULT VALUE, AS SPECIFIED BY THE <FIELD VALUE PART> IN THE DECLARATION, TO BE ASSIGNED TO THE FIELD. A <FIELD VALUE> OF "*" CAUSES THE FIELD TO BE IGNORED. IF NO INITIAL VALUE IS SPECIFIED, THEN <EMPTY> IS EQUIVALENT TO "*".

16. IF THE <FIELD VALUE LIST> CONTAINS FEWER <FIELD VALUE>S THAN DECLARED IN THE CORRESPONDING <LAYOUT ITEM LIST>, THE COMPILER WILL EMIT CODE AS NECESSARY FOR THE REMAINING <FIELD VALUE PART>S WHICH ARE NOT <EMPTY>.

EXAMPLES:

ARITHMETIC EXPRESSIONS:

```
3
+3
Q
Q-V
HO ← (IF GONE THEN 2 ELSE Z/3)
IF JOY THEN X ELSE 4+Q
WxU-Qx(S+CU)
IF Q>0 THEN S+3xQ/A ELSE ZxS+3xA
IF A>0 THEN U+V ELSE IF AxB>17 THEN U/V ELSE IF K≠Y
THEN V/U
.57@12xA[Nx(N-1)/2,0]
QxV*2
P MOD 2
A[2, SIZ*2 DIV QUANT] ←IF BOOEX THEN Q←Q+1 ELSE 4
```

SIMPLE ARITHMETIC EXPRESSIONS:

```
Q+V
Q-V
```

```
-Q
3
+3
Q
P MOD 2
Y*3
4xR DIV S
A[I]-B[J]+5.3
```

TERMS:

```
Q
Q MOD V
7.394@-8
SUM
W[I+2,8]
2x(X+Y)
Y*3
Q MOD V DIV 2
```

PRIMARIES:

```
7
J.K
J
Q + R
VARINAME & LOOK (6, IF BOOEXP THEN Q←Q+1 ELSE 2,
AN[3,5])
2 & SEET (X←F(A+B), 12, VO) & NAW (27, TRUE)
@-72
7
O&CONCAT()
Q
```

FIELD DESIGNATORS:

```
SIGNIFY(X).Z
VARINAME.FIELDNAME
FUNC(A,TRUE).F6
(X←ARITHEXP(Z7)+Q MOD 2).F711
```

FIELD OPERANDS:

```
CASE X←X+U OF (V, +27xF(Z,7),ARY[2,BOOQ])
Q
TALLYHO(TFX)
(Q+RxZ-T)
```

LAYOUTS:

```
CHAS (Q+R*6, F(A))
GHT(ZY,7)
```

9.3. ARRAY EXPRESSIONS

SYNTAX:

```
<ARRAY EXPRESSION> ::= <ARRAY ASSIGNMENT> / <ARRAY PRIMARY> /
      <IF CLAUSE> <ARRAY EXPRESSION> ELSE <ARRAY EXPRESSION>
<ARRAY ASSIGNMENT> ::=
      <ARRAY DESIGNATOR> <REPLACEMENT OPERATOR> <ARRAY EXPRESSION>
<ARRAY DESIGNATOR> ::= <ARRAY IDENTIFIER> <SUBARRAY DESIGNATOR> /
      <ARRAY VARIABLE>
<SUBARRAY DESIGNATOR> ::= <EMPTY> /
      [<SUBSCRIPT PART> <SUBARRAY PART>]
<SUBSCRIPT PART> ::= <EMPTY> / <SUBSCRIPT LIST> ,
<SUBARRAY PART> ::= * / <SUBARRAY PART> , *
<ARRAY VARIABLE> ::= <SIMPLE VARIABLE> / <ARRAY ITEM>
<ARRAY ITEM> ::= <ITEM>
<ARRAY PRIMARY> ::= <ARRAY DESIGNATOR> / (<ARRAY EXPRESSION>) /
      <ARRAY PRIMARY> & <LAYOUT> / <WORD EXPRESSION>
```

SEMANTICS:

1. AN <ARRAY EXPRESSION> IS REPRESENTED BY AN UNINDEXED DATA DESCRIPTOR.

2. AN <ARRAY DESIGNATOR> REFERENCES A DATA DESCRIPTOR.

3. AN <ARRAY ASSIGNMENT> INITIALIZES OR CHANGES THE VALUES OF THE VARIOUS FIELDS IN THE CORRESPONDING DATA DESCRIPTOR.

EXAMPLES:

ARRAY EXPRESSIONS:

```
IF BOOVAR THEN ARRVAR [2,3,*] ELSE A2S3 ← A1S3
```

```
      A2S2←A3S3
      ARVR[3,*]

ARRAY PRIMARIES:
----- ----------

      QVAR [3,*]
      (A2S2←A3S3)
      QVAR [2,*] & C2S

ARRAY DESIGNATORS:
----- ------------

      NEXT
      NXTON[2,3,*]

ARRAY VARIABLES:
----- ----------

      DELTA
      ARRAYNAME @ ARRAYDECQUE

SUBARRAY DESIGNATORS:
-------- ------------

      [*]
      [2,*]
```

9.4. BOOLEAN EXPRESSIONS

SYNTAX:

```
<BOOLEAN EXPRESSION> ::= <CONDITIONAL BOOLEAN EXPRESSION> /
      <BOOLEAN ASSIGNMENT> / <SIMPLE BOOLEAN EXPRESSION> /
      <WORD EXPRESSION>
<CONDITIONAL BOOLEAN EXPRESSION> ::=
      <IF CLAUSE> <BOOLEAN EXPRESSION> ELSE <BOOLEAN EXPRESSION>
<BOOLEAN ASSIGNMENT> ::= <BOOLEAN VARIABLE> <REPLACEMENT OPERATOR>
      <BOOLEAN EXPRESSION>
<SIMPLE BOOLEAN EXPRESSION> ::= <IMPLICATION> /
      <SIMPLE BOOLEAN EXPRESSION> EQV <IMPLICATION>
<IMPLICATION> ::= <BOOLEAN TERM> / <IMPLICATION> IMP <BOOLEAN TERM>
<BOOLEAN TERM> ::= <BOOLEAN FACTOR> / <BOOLEAN TERM> OR
      <BOOLEAN FACTOR>
<BOOLEAN FACTOR> ::= <BOOLEAN SECONDARY> / <BOOLEAN FACTOR> AND
      <BOOLEAN SECONDARY>
<BOOLEAN SECONDARY> ::= <BOOLEAN PRIMARY> / NOT <BOOLEAN PRIMARY>
<BOOLEAN PRIMARY> ::= <LOGICAL VALUE> / <RELATION> /
      <BOOLEAN ITEM> / <BOOLEAN FIELD DESIGNATOR> /
      <BOOLEAN OPERAND> /
      <BOOLEAN PRIMARY> & <BOOLEAN EXPRESSION> <CONCATENATION> /
      <BOOLEAN PRIMARY> & <BOOLEAN LAYOUT> / <TABLE MEMBERSHIP> /
      <VALUE DESIGNATOR>
<BOOLEAN ITEM> ::= <ITEM>
<BOOLEAN FIELD DESIGNATOR> ::=
      <BOOLEAN OPERAND> . <FIELD IDENTIFIER>
<BOOLEAN OPERAND> ::= (<BOOLEAN EXPRESSION>) /
      <BOOLEAN VARIABLE> / <BOOLEAN FUNCTION DESIGNATOR> /
      <CASE HEAD> (<BOOLEAN EXPRESSION LIST>)
<BOOLEAN EXPRESSION LIST> ::= <BOOLEAN EXPRESSION> /
      <BOOLEAN EXPRESSION LIST> , <BOOLEAN EXPRESSION>
<BOOLEAN LAYOUT> ::= <LAYOUT>
<BOOLEAN VARIABLE> ::= <VARIABLE>
```

<BOOLEAN FUNCTION DESIGNATOR> ::= <FUNCTION DESIGNATOR> /
<BOOLEAN INTRINSIC>
<RELATION> ::= <ARITHMETIC RELATION> / <REFERENCE RELATION> /
<STRING RELATION> / <POINTER RELATION>
<ARITHMETIC RELATION> ::= <ARITHMETIC EXPRESSION>
<ARITHMETIC RELATIONAL> <ARITHMETIC EXPRESSION>
<ARITHMETIC RELATIONAL> ::= <RELATIONAL OPERATOR> / IS
<REFERENCE RELATION> ::= <REFERENCE EXPRESSION>
<REFERENCE RELATIONAL> <REFERENCE EXPRESSION>
<REFERENCE RELATIONAL> ::= = / ≠
<STRING RELATION> ::= <POINTER EXPRESSION> <RELATIONAL OPERATOR>
<POINTER EXPRESSION> FOR <ARITHMETIC EXPRESSION>
<POINTER RELATION> ::= <POINTER EXPRESSION> <REFERENCE RELATIONAL>
<POINTER EXPRESSION>
<TABLE MEMBERSHIP> ::= <ARITHMETIC EXPRESSION> IN <TABLE POINTER> /
<POINTER EXPRESSION> IN <TABLE POINTER>
<TABLE POINTER> ::= ALPHA / ALPHA6 / ALPHA8 /
<SUBSCRIPTED VARIABLE>

SEMANTICS:

1. A BOOLEAN EXPRESSION OBTAINS A LOGICAL VALUE.

2. A VARIABLE, VALUE DESIGNATOR, OR FUNCTION DESIGNATOR USED AS A "BOOLEAN" PRIMARY MUST BE OF TYPE "BOOLEAN".

3. THE SEQUENCE IN WHICH OPERATIONS ARE PERFORMED IS DETERMINED BY THE PRECEDENCE OF THE OPERATORS.

4. THE ORDER OF PRECEDENCE OF OPERATORS IS AS FOLLOWS:

FIRST: ARITHMETIC EXPRESSIONS
SECOND: RELATIONS
THIRD: NOT
FOURTH: AND
FIFTH: OR

SIXTH: IMP

SEVENTH: EQV

WHEN OPERATIONS ARE OF THE SAME ORDER OF PRECEDENCE, THE EXPRESSION IS EVALUATED FROM LEFT TO RIGHT.

5. THE LOGICAL OPERATORS ARE DEFINED BY THE FOLLOWING TRUTH TABLE:

OPERAND A	OPERAND B	NOT A	A AND B	A OR B	A IMP B	A EQV B
TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE

THE ABOVE DEFINED BOOLEAN OPERATIONS ARE PERFORMED ON ALL 48 BITS OF THE OPERAND OR OPERANDS INVOLVED. FOR EXAMPLE, REAL (NOT TRUE) IS NOT EQUIVALENT TO REAL (FALSE) BECAUSE "NOT" COMPLEMENTS ALL 48 BITS OF THE OPERAND IN THE TOP OF THE STACK, WHEREAS "FALSE" ONLY IMPLIES THAT BIT ZERO OF THAT OPERAND IS OFF.

6. RELATIONS DEFINE THE MANNER IN WHICH THE VARIOUS RELATIONAL OPERATORS ARE USED WITH THE VARIOUS EXPRESSION TYPES.

7. THE "IS" OPERATOR COMPARES ALL THE BITS (INCLUDING TAG BITS) OF TWO B6700/B7700 WORDS. IF ALL BITS ARE EQUAL, THE RESULT OF THE COMPARISON IS TRUE. WHEN USED WITH ARITHMETIC OPERANDS THE "IS" RELATIONAL OPERATOR DIFFERS FROM "=" IN THAT IT COMPARES BIT PATTERNS FOR EQUALITY, AND DOES NO NORMALIZATION.

8. THE <TABLE MEMBERSHIP> CONSTRUCT ALLOWS THE PROGRAMMER TO TEST WHETHER A GIVEN CHARACTER IS A MEMBER OF A PREDEFINED TABLE REFERENCED BY THE <TABLE POINTER>. THE CHARACTER IN QUESTION MAY BE EITHER A <STRING CHARACTER> OR A CHARACTER IN AN <ARRAY ROW> REFERENCED BY A <POINTER EXPRESSION>. THE <TABLE MEMBERSHIP> TEST REFERENCES A BIT IN MEMORY IN THE FOLLOWING

WAY: THE BINARY REPRESENTATION OF THE CHARACTER BEING TESTED IS SPLIT INTO TWO PARTS. THE LOW-ORDER 5 BITS BECOME A BIT INDEX, AND THE REMAINING HIGH-ORDER BITS BECOME A WORD INDEX. THE WORD INDEX IS ADDED TO THE <SUBSCRIPT> IN THE <TABLE POINTER>. THIS INDEXING OPERATION THUS SELECTS A WORD FROM THE TABLE. THE BIT INDEX IS THEN SUBTRACTED FROM 31 AND THE RESULT IS USED TO REFERENCE ONE OF THE LOW-ORDER 32 BITS IN THE SELECTED WORD. FINALLY, THE CHARACTER IS IN THE TABLE IF AND ONLY IF THIS REFERENCED BIT IS ON.

THE <TABLE POINTER> CONSTRUCT ALLOWS SEVERAL TABLES TO BE CONTAINED IN ONE <ARRAY ROW>. THE <SUBSCRIPTED VARIABLE> ALWAYS INDICATES THE BEGINNING OF THE DESIRED TABLE. ALPHA, ALPHA6, AND ALPHA8 MAY BE THOUGHT OF AS RESERVED <SUBSCRIPTED VARIABLE>S. ALPHA8 IS A <TABLE POINTER> FOR EBCDIC LETTERS AND DIGITS; ALPHA6 FUNCTIONS SIMILARLY FOR BCL LETTERS AND DIGITS.

EXAMPLES:

BOOLEAN EXPRESSIONS:

```
BOOLE ← A EQV B[J,1]
A OR B
IF K>1 THEN S>W ELSE L<C
M[I]
```

SIMPLE BOOLEAN EXPRESSION:

```
UGO EQV IGO
NOT SO
WEGO OR HEGO EQV IGO EQV UGO
```

IMPLICATIONS:

```
ITISRAINING IMP GROUNDISWET
BOOVAR AND THIS IMP TOMOR[1,2]
THIS IMP THAT IMP THOSE
NOT SO
```

BOOLEAN TERMS:

```
(B>C) OR (D>E)
BOOV AND BOON OR BOOK
A[1,2] AND BVAR OR (NOT THT) OR YEST
NOT SO
```

BOOLEAN FACTORS:

```
BOOV AND BOON
NOT (J>2) AND TRUE
A[J+1,Z-3] AND VARB AND NEXTM
NOT SO
```

BOOLEAN SECONDARIES:

```
TRUE
NOT SO
```

BOOLEAN PRIMARIES:

```
FALSE
X>V
(NOT SO)
BOOVAR.F2
TRUE & CONGLOM(TRUE,FALSE,Z+010,TRUE)
```

BOOLEAN FIELD DESIGNATORS:

```
BAHK. LEFTMOST
FRNT. FORTYON
```

BOOLEAN FIELD OPERANDS:

```
(IF K>1 THEN V<2 ELSE (V2←DYNAM))
BOOLVARB
BOOFUNC(Q>V)
CASE ZYGLOT OF (TRUE, NOT SO, ETCETERA<Q)
BOOLEAN(FLAGWORD)
```

9.5. CASE EXPRESSIONS

SYNTAX:

<CASE EXPRESSION> ::= <CASE HEAD> (<EXPRESSION LIST>)
<EXPRESSION LIST> ::= <ARITHMETIC EXPRESSION LIST> /
<BOOLEAN EXPRESSION LIST> / <DESIGNATIONAL EXPRESSION LIST> /
<POINTER EXPRESSION LIST>

SEMANTICS:

CASE EXPRESSIONS PROVIDE A CONVENIENT MEANS OF SELECTING ONE OF MANY ALTERNATIVE EXRESSIONS OF THE SAME KIND TO BE EVALUATED AT A PARTICULAR POINT DURING EXECUTION. THE <EXPRESSION> TO BE EVALUATED IS SELECTED AS FOLLOWS: THE <ARITHMETIC EXPRESSION> IN THE CASE HEAD> IS EVALUATED AND INTEGERIZED BY ROUNDING IF ITS VALUE IS NOT INTEGRAL. THIS VALUE IS THEN USED AS AN INDEX INTO THE <EXPRESSION LIST>. THE COMPONENT EXPRESSIONS OF THE <EXPRESSION LIST> ARE INDEXED SEQUENTIALLY FROM 0 THROUGH N-1, WHERE N IS THE NUMBER OF EXPRESSIONS IN THE LIST. THE INDEXED <EXPRESSION> IS THEN EVALUATED AND ITS VALUE IS THE VALUE OF THE <CASE EXPRESSION>. IF THE VALUE OF THE INDEX LIES OUTSIDE THE RANGE 0 TO N-1, ANYTHING CAN HAPPEN. THE ESPOL COMPILER DOES NOT EMIT "RANGE CHECKING CODE" WHICH VALIDATES THE EVALUATED <ARITHMETIC EXPRESSION>.

9.6. CONDITIONAL EXPRESSIONS

SYNTAX:

<CONDITIONAL EXPRESSION> ::= <CONDITIONAL ARITHMETIC EXPRESSION> /
<CONDITIONAL BOOLEAN EXPRESSION> /
<CONDITIONAL DESIGNATIONAL EXPRESSION> /
<CONDITIONAL POINTER EXPRESSION>

SEMANTICS:

EXPRESSIONS OF THE FORM "<IF CLAUSE> <EXPRESSION> ELSE <EXPRESSION>" ARE CALLED CONDITIONAL EXPRESSIONS. DEPENDING UPON EITHER THE VALUE OF THE <BOOLEAN EXPRESSION> IN THE <IF CLAUSE> OR, IF ONE OR BOTH OF THE ALTERNATIVE EXPRESSIONS ARE THEMSELVES CONDITIONAL, THE VALUES OF THE BOOLEAN EXPRESSIONS IN SEVERAL IF CLAUSES, AN <EXPRESSION> IS SELECTED FOR EVALUATION. ALL THE ALTERNATIVE EXPRESSIONS MUST BE OF THE SAME TYPE.

THE SELECTION PROCESS PROCEEDS AS FOLLOWS: FIRST, THE <BOOLEAN EXPRESSION> FOLLOWING THE FIRST "IF" IS EVALUATED; IF THE RESULTING VALUE IS TRUE, THE <EXPRESSION> FOLLOWING THE "THEN" IS EVALUATED AND THE <EXPRESSION> FOLLOWING "ELSE" IGNORED, OTHERWISE THE <EXPRESSION> FOLLOWING THE "ELSE" IS EVALUATED. IF EITHER OF THE ALTERNATIVE EXPRESSIONS IS CONDITIONAL, THE PROCESS IS REPEATED UNTIL AN UNCONDITIONAL <EXPRESSION> IS SELECTED FOR EVALUATION.

9.7. DESIGNATIONAL EXPRESSIONS

SYNTAX:

```
<DESIGNATIONAL EXPRESSION> ::= <LABEL DESIGNATOR> /
      <CASE HEAD> (<DESIGNATIONAL EXPRESSION LIST>) /
      <CONDITIONAL DESIGNATIONAL EXPRESSION>
<CONDITIONAL DESIGNATIONAL EXPRESSION> ::=
      <IF CLAUSE> <DESIGNATIONAL EXPRESSION> ELSE
      <DESIGNATIONAL EXPRESSION>
<DESIGNATIONAL EXPRESSION LIST> ::= <DESIGNATIONAL EXPRESSION> /
      <DESIGNATIONAL EXPRESSION LIST>,
      <DESIGNATIONAL EXPRESSION>
<LABEL DESIGNATOR> ::= <LABEL IDENTIFIER>
```

SEMANTICS:

A DESIGNATIONAL EXPRESSION SPECIFIES A LABEL.

9.8. POINTER EXPRESSIONS

SYNTAX:

```
<POINTER EXPRESSION> ::= <CONDITIONAL POINTER EXPRESSION> /
      <SIMPLE POINTER EXPRESSION>
<CONDITIONAL POINTER EXPRESSION> ::=
      <IF CLAUSE> <POINTER EXPRESSION> ELSE <POINTER EXPRESSION>
<SIMPLE POINTER EXPRESSION> ::= <POINTER PRIMARY> <SKIP> /
      <POINTER ASSIGNMENT> / <WORD ARRAY ROW> /
      <SUBSCRIPTED WORD VARIABLE>
<POINTER PRIMARY> ::= <POINTER IDENTIFIER> / (<POINTER EXPRESSION>)
      / <CASE HEAD> ( <POINTER EXPRESSION LIST> ) /
      <POINTER DESIGNATOR>
<SKIP> ::= <EMPTY> / <ADDING OPERATOR> <PRIMARY>
<POINTER IDENTIFIER> ::= <IDENTIFIER>
<POINTER DESIGNATOR> ::= POINTER ( <POINTER PARAMETERS> )
<POINTER PARAMETERS> ::= <ARRAY PART> / <ARRAY PART>,
      <CHARACTER SIZE>
<CHARACTER SIZE> ::= 4 / 6 / 8 / *
<ARRAY PART> ::= <ARRAY ROW> / <SUBSCRIPTED VARIABLE> /
      <ARRAY IDENTIFIER>
<SUBSCRIPTED WORD VARIABLE> ::= <SUBSCRIPTED VARIABLE>
<ARRAY ROW> ::= <ARRAY IDENTIFIER> [ <ROW DESIGNATOR> ]
<ROW DESIGNATOR> ::= * / <ROW>, *
<WORD ARRAY ROW> ::= <ARRAY ROW>
<ROW> ::= <ARITHMETIC EXPRESSION> / <ROW>, <ARITHMETIC EXPRESSION>
<POINTER EXPRESSION LIST> ::= <POINTER EXPRESSION> /
      <POINTER EXPRESSION LIST> , <POINTER EXPRESSION>
<POINTER ASSIGNMENT> ::= <POINTER VARIABLE>
      <REPLACEMENT OPERATOR> <POINTER EXPRESSION>
<POINTER VARIABLE> ::= <VARIABLE>
```

SEMANTICS:

1. A POINTER EXPRESSION ADDRESSES A CHARACTER POSITION WITHIN AN ARRAY ROW.

2. AN IDENTIFIER USED AS A POINTER PRIMARY MUST BE OF TYPE POINTER.

3. IF A POINTER EXPRESSION IS ENCLOSED IN PARENTHESES, IT IS EVALUATED FIRST AND ITS VALUE IS USED AS A PRIMARY.

4. IF SKIP IS NOT EMPTY, THE POINTER VALUE IS ADJUSTED BY "L" CHARACTERS TO THE RIGHT OR LEFT, WHERE "L" IS THE ABSOLUTE VALUE OF THE ARITHMETIC EXPRESSION. IF THE ADDING OPERATOR IS +, SKIPPING IS TO THE RIGHT. IF THE OPERATOR IS -, SKIPPING IS TO THE LEFT.

5. A POINTER DESIGNATOR MAY BE USED TO CREATE A POINTER VALUE WHICH REFERENCES A SPECIFIC CHARACTER POSITION IN AN ARRAY. THE POINTER DESIGNATOR HAS THREE FORMS:

POINTER(A, L)	YIELDS A POINTER VALUE "POINTING" TO A. A IS AN ARRAY IDENTIFIER WHICH MAY EITHER BE SUBSCRIPTED OR UNSUBSCRIPTED. L IS A CHARACTER LENGTH IN BITS (FOUR, SIX, OR EIGHT, OR ANOTHER POINTER, IN WHICH CASE THE SIZE OF THAT POINTER IS USED.
POINTER(A)	YIELDS A POINTER WITH A CHARACTER SIZE FIELD SET TO ZERO. EBCDIC OR BCL DEFAULT CHARACTER SIZE WILL BE DETERMINED BY THE HARDWARE, BASED ON HOW THE POINTER IS USED.
POINTER(A,*)	SAME AS POINTER(A).
POINTER (P1,P2)	THIS FORM PUTS THE SIZE FIELD OF POINTER P2 INTO POINTER P1.

6. A POINTER MAY BE INITIALIZED EITHER BY A POINTER ASSIGNMENT OR BY APPEARING AS AN UPDATE POINTER IN A SCAN OR REPLACE STATEMENT OR A STRING COMPARISON.

9.9. REFERENCE EXPRESSIONS

SYNTAX:

```
<REFERENCE EXPRESSION> ::= NULL / <REFERENCE ASSIGNMENT> /
      <IF CLAUSE> <REFERENCE EXPRESSION> ELSE
      <REFERENCE EXPRESSION> / <REFERENCE VARIABLE> /
      <ENTRY EXPRESSION> / <QUEUE DESIGNATOR> /
      REFERENCE (<VARIABLE>) / <FUNCTION DESIGNATOR> /
      <WORD EXPRESSION> / (<REFERENCE EXPRESSION>) /
      <REFERENCE ITEM> / <CASE HEAD> (<REFERENCE EXPRESSION LIST>)
<ENTRY EXPRESSION> ::= <QUEUE NAME> (<ACTUAL ITEM LIST>)
<QUEUE NAME> ::= <QUEUE IDENTIFIER> / <QUEUE ARRAY IDENTIFIER>
<ACTUAL ITEM LIST> ::= <ACTUAL PARAMETER LIST>
<REFERENCE ASSIGNMENT> ::= <REFERENCE VARIABLE>
      <REPLACEMENT OPERATOR> <REFERENCE EXPRESSION>
<QUEUE DESIGNATOR> ::= <QUEUE IDENTIFIER> /
      <QUEUE ARRAY IDENTIFIER> [<ARITHMETIC EXPRESSION>]
<REFERENCE VARIABLE> ::= <VARIABLE>
<REFERENCE ITEM> ::= <ITEM>
<REFERENCE EXPRESSION LIST> ::= <REFERENCE EXPRESSION> /
      <REFERENCE EXPRESSION LIST>, <REFERENCE EXPRESSION>
```

SEMANTICS:

1. A <REFERENCE EXPRESSION> POINTS TO A CONSTRUCT OR LOCATION. A "NULL" REFERENCE EXPRESSION POINTS AT NOTHING AND TO NOWHERE.

2. A <QUEUE DESIGNATOR> POINTS TO THE ENTRY AT THE HEAD OF THE DESIGNATED QUEUE.

3. AN <ENTRY EXPRESSION> CAUSES THE CREATION OF AN ENTITY HAVING THE FORMAT OF AN ENTRY IN THE NAMED QUEUE OR QUEUE ARRAY. A POINTER TO THIS POTENTIAL ENTRY IS RETURNED AS A <REFERENCE EXPRESSION>.

4. THE "REFERENCE" TRANSFER FUNCTION GENERATES A <REFERENCE EXPRESSION> POINTING TO A VARIABLE.

EXAMPLES:

REFERENCE EXPRESSIONS:

```
IF A > B THEN NEXTONE ← LASTONE ELSE NULL
NULL
NEXTONE ←LASTONE
NEXTONE
QUEUP (LOCAT,SIZEX)
ARY[2,3]
REFERENCE (BOOVARB)
REFFUNC(ARAY[2,7])
CASE Q[V←7+ATT[NUM]] OF (FIRSTONE, NEXTONE, LASTONE)
IF ARRAYROW IS NULL THEN
```

ENTRY EXPRESSION:

```
QUNM (HERE, THERE, EVY)
```

QUEUE DESIGNATORS:

```
QUNM
DYNAM[3]
```

9.10. WORD EXPRESSIONS

SYNTAX:

```
<WORD EXPRESSION> ::= <WORD ASSIGNMENT> / <WORD VARIABLE> /
      <WORD ITEM> / <IF CLAUSE> <WORD EXPRESSION> ELSE
      <WORD EXPRESSION> / WORD (<MOST EXPRESSIONS>) /
      <FUNCTION DESIGNATOR> / <CASE HEAD>
      (<WORD EXPRESSION LIST>) / (<WORD EXPRESSION>)
<WORD VARIABLE> ::= <VARIABLE>
<MOST EXPRESSIONS> ::= <ARITHMETIC EXPRESSION> /
      <BOOLEAN EXPRESSION> / <REFERENCE EXPRESSION> /
      <ARRAY EXPRESSION>
<WORD EXPRESSION LIST>::= <WORD EXPRESSION> /
      <WORD EXPRESSION LIST>, <WORD EXPRESSION>
<WORD ASSIGNMENT> ::=
      <WORD VARIABLE> <REPLACEMENT OPERATOR> <EXPRESSION>
```

SEMANTICS:

1. A WORD EXPRESSION OBTAINS A WORD VALUE.

2. A WORD VALUE IS REGARDED AS A 48-BIT FIELD WITH NO TYPE SIGNIFICANCE.

3. A WORD TRANSFER FUNCTION BEHAVES IN MUCH THE SAME MANNER AS THE REAL AND BOOLEAN TRANSFER FUNCTIONS; I.E., IT SUPPRESSES SYNTAX CHECKING WHICH WOULD OTHERWISE BE INVOKED.

PRAGMATICS:

1. A WORD VARIABLE IS ACCESSED VIA AN LODT. HOWEVER NO SUCH ACTION MUST BE EXPECTED FOR THE EXPRESSION ASSOCIATED WITH THE WORD TRANSFER FUNCTION; THE CODE APPLICABLE TO THE EXPRESSION

WILL BE COMPILED.

2. THERE IS NO GUARANTEE THAT A CORRECTLY COMPILED WORD EXPRESSION WILL PRODUCE VALID B6700/B7700 CODE.

APPENDIX A. COMPILER ERROR MESSAGES

THE LEFTMOST DIGIT OF THE ERROR NUMBERS IS USUALLY THE NUMBER OF THE SECTION OF THE COMPILER IN WHICH THE ERROR WAS DETECTED.

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
100	TABLE	UNKNOWN IDENTIFIER
101	STATEMENT	SCANNER ERROR
102	PRIMARY	SCANNER ERROR
103	BOOPRIM	SCANNER ERROR
110	PASSFILE	ILLEGAL FILE DESIGNATOR
199		PROCEDURE HAS NOT YET BEEN CODED
300	SCAN	IDENTIFIER OR NUMBER OF >63 CHARACTERS
301	TABLE	ILLEGAL CONSTRUCT
302	HOOK	TOO MUCH NESTING OF DEFINES
303	TABLE	NUMBER IS TOO LARGE
304	UNHOOK	EXTRANEOUS CROSSHATCH
305	ASSOCIATE	MISSING "(" OR "["
306	ASSOCIATE	MISSING ")" OR "]" OR TOO MANY PARAMETERS
307	GOBBLE	INVALID STRING CHARACTER
308	GOBBLE	INVALID STRING CODE OR ILLEGAL STRING SYNTA
309	SCAN	MISSING """ IN STRING
310	DOLLARCARD	UNKNOWN DOLLAR CARD OPTION
311	DOLLARCARD	INVALID IDENTIFIER
312	DOLLARCARD	"BUMP TO" DOLLAR OPTION NOT FOLLOWED BY NUM
313	DOLLARCARD	>7 DIGITS IN ARGUMENT TO "BUMP" DOLLAR OPT
314	READACARD	SEQUENCE ERROR.
315	FINDUSER	USER OPTION MUST BE REFERENCED PREVIOUSLY
316	DOLLARCARD	INVALID GO ON A DOLLAR CARD
325	STARTINCLUDING	INCLUDES TOO DEEPLY NESTED.
326	STARTINCLUDE	MISSING ENDING SEQUENCE NUMBER ON INCLUDE C
327	STARTINCLUDING	MISSING INCLUDE-FILE OPTION.
328	STARTINCLUDE	INVALID INCLUDE-OPTION--IGNORED

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
340	LOADINFO	LOADINFO MUST PRECEDE GLOBALS
350	ATTRIBMNEREF	MNEMONIC FOR ATTRIBUTE VALUE EXPECTED
400	EMITTERS	TOO MUCH CODE IN THIS SEGMENT
401	EMITV OR EMITN	DISPLACEMENT TOO BIG
402	SUBSCRIBER	MISSING BRACKET OR SUBSCRIPT
403	DOTTER	EXPRESSION NOT ARITHMETIC
404	DOTIT	"."PERIOD NOT FOLLOWED BY FIELD IDENTIFIER
405	PURGE	LABEL DECLARED FORWARD NOT SEEN
406	PURGE	PROCEDURE DECLARED FORWARD NOT SEEN
407	GETSPACE	ILLEGAL ADDRESS-PART VALUE
408	GETSPACE	ILLEGAL ADDRESS-PART SYNTAX
410	GETSPACE	TOO MANY STACK CELLS AT THIS LEVEL
415	GETSTACK	TOO MANY TEMPORARIES IN USE AT ONE TIME
420	PURGE	COMPILER ERROR INVOLVING BAD GO TO LABEL
501	IFCLAUSE	MISSING "THEN"
502	CASEHEAD	EXPRESSION NOT ARITHMETIC
503	CASEHEAD	MISSING "OF"
504	IFEXP	MISSING "ELSE"
505	EXPRESSION	EXPRESSION IS NOT OF REQUIRED TYPE
506	CASEXP	MISSING "("
507	CASEXP	MISSING ")"
508	RELATION	THIS EXPRESSION MAY NOT APPEAR IN A RELATIO
509	RELATION	THIS RELATION MAY USE ONLY "=" OF "≠"
510	RELATION	MISSING "FOR" IN STRING RELATION
511	RELATION	ILLEGAL EXPRESSION TYPE
512	BEXP	EXPRESSION NOT BOOLEAN TYPE
513	AEXP	IF EXPRESSION NOT ARITHMETIC TYPE
514	SIMPARITH	ARRAY EXPRESSION MAY NOT BE SIGNED
515	TERM	ARRAY AND WORD EXPRESSIONS MAY NOT BE OPERATED UPON
516	BOOSEC	CANNOT NEGATE AN EXPR UNLESS ITS BOOLEAN
517	BOOCOMP	EXPRESSION NOT BOOLEAN
518	REXP	VARIABLE NOT REFERENCE TYPE

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
519	REXP	NOT ENOUGH SUBSCRIPTS ON REFERENCE ARRAY
520	REXP	REF EXPR CANNOT START WITH IDENTIFIER OF THIS TYPE
521	REXP	EXPRESSION NOT OF TYPE REFERENCE
522	REXP	MISSING ")"
523	REXP	CASE EXPR NOT OF TYPE REFERENCE
524	REXP	IF EXPRESSION NOT OF TYPE REFERENCE
525	REXP	REF EXPR CANT START WITH THIS QUANTITY
526	REXP	MISSING "("
527	REXP	CANNOT TRANSFER THIS TO TYPE REFERENCE
528	REXP	NOT ENOUGH SUBSCRIPTS
529	VARIABLE	REGISTER ASSIGNMENT MUST BE LEFT-MOST ASSIGNMENT
530	VARIABLE	LEFT BRACKET DOES NOT FOLLOW ARRAY IDENTIFIER
531	VARIABLE	FIELD DESIGNATOR IS NOT LEFT-MOST IN LEFT-PART LIST
532	VARIABLE	A PROCEDURE IDENTIFIER IS USED OUTSIDE OF ITS SCOPE
533	LAYITOUT	"&" NOT FOLLOWED BY LAYOUT
534	LAYITOUT	MISSING "("
535	LAYITOUT	MISSING ")"
537	ENTRYEXP	NOT YET CODED
538	VARIABLE	A VARIABLE IS NOT FOLLOWED BY A REPLACEMENT OPERATOR
539	VARIABLE	A READ ONLY ARRAY MAY NOT BE STORED INTO
540	PEXP	EXPRESSION NOT POINTER TYPE
541	PTRCOMP	SKIP PART NOT INTEGER OR REAL EXPRESSION
542	ITEMREFERENCE	INCORRECT SYNTAX FOR AN ITEM REFERENCE
543	ENTRYEXPR	INCORRECT SYNTAX FOR ENTRY EXPRESSION
544	LAYITOUT	EXPRESSION NOT OF REQUIRED TYPE
545	VARIABLE	POINTER IDENTIFIER MAY NOT BE SUBSCRIPTED
546	SETVARIABLE	CANNOT MIX SETS OF VARYING SIZES

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
		IN SAME EXP
547	SETVARIABLE	MISSING REPLACEMENT OPERATOR
548	VARIABLE	REGISTERS MAY NOT BE PARTIAL FIELDED
549	ARRAYROW	MISSING LEFT BRACKET
550	ARRAYROW	ARRAY ROW MUST HAVE 1 ASTERISK
551	LONGSTRING	TOO MANY STRING CHARACTERS OR MISSING QUOTE
552	STRINGSOURCE	STRING MUST BE 4-, 6-, OR 8-BIT CHARACTERS
553	ARITHCOMP	DOUBLE EXPRESSION MAY NOT BE DOTTED
554	TERM SIMPARITH	WORD EXPRESSIONS CONSIST OF WORD PRIMARY ONLY
555	PARTIALFIELD OR CONCATENATE	ILLEGAL BIT DESIGNATION EXPRESSN TYPE
556	PARTIALFIELD OR CONCATENATE	INVALID LEFT BIT VALUE SPECIFIED
557	PARTIALFIELD OR CONCATENATE	MISSING COLON
558	PARTIALFIELD OR CONCATENATE	INVALID NUMBER OF BITS SPECIFIED
559	PARTIALFIELD OR CONCATENATE	NUMBER OF BITS SPECIFIED IS TOO LARGE FOR SPECIFIED LEFT BIT VALUE
560	PARTIALFIELD OR CONCATENATE	MISSING LEFT BRACKET
561	PARTIALFIELD OR CONCATENATE	MISSING RIGHT BRACKET
567	VARIABLE	EVENTS MAY NOT BE PARTIAL FIELDS
573	REPLACESTMT	MISSING COMMA OR SEMICOLON
574	SUBSCRIBER	ACTUAL SUBSCRIPT(S) MUST PRECEDE ROW DESIGNATOR(S)
575	VARIABLE	FIELDS MAY NOT BE DUPLICATED AND LOADED
576	BITFIDDLE	MISSING "("
577	BITFIDDLE	MISSING ","
578	BITFIDDLE	BAD BIT NUMBER.

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
579	BITFIDDLE	MISSING ")"
580	VARIABLE	IDENTIFIER EXPECTED
581	VARIABLE	PARTIAL FIELD FOR COMPILETIME VARIABLE MUST BE A CONSTANT
582	VARIABLE	COMPILETIME VARIABLE MUST BE CONSTANT
600	CASESTMT	MISSING SEMICOLON
601	CASESTMT	MISSING "BEGIN"
602	CASESTMT	TOO MANY STATEMENTS
603	PROCALL	ILLEGAL USE OF PROCEDURE IDENTIFIER
604	PROCALL	EITHER ACTUAL OR FORMAL PARAMETERS DO NOT AGREE AS TO NUMBER, OR EXTRA ")"
605	ACTUAL PARAPART	ACTUAL AND FORMAL ARRAYS DO NOT HAVE SAME NUMBER OF DIMENSIONS
607	ACTUAL PARAPART	NO ACTUAL PARAMETERS MAY START WITH A QUANTITY OF THIS TYPE
608	FORSTMT	IMPROPER FOR INDEX VARIABLE
609	FORSTMT	MISSING UNTIL OR WHILE IN STEP ELEMENT
610	FORSTMT	MISSING DO IN FOR CLAUSE
611	FORSTMT	MISSING LEFT ARROW FOLLOWING INDEX VARIABLE
612	LABELR	MISSING COLON
613	LABELR	THE LABEL WAS NOT DECLARED IN THIS BLOCK
614	LABELR	THE LABEL HAS ALREADY OCCURRED
615	GOSTMT	LABEL OR CASE DOES NOT FOLLOW GO TO
616	GOSTMT	IMPROPER GO TO WITH CASE
617	GOSTMT	MISSING "("
618	GOSTMT	ONLY A LABEL MAY APPEAR IN THE LIST
619	GOSTMT	MISSING ")"
620	ACTUAL PARAPART	ACTUAL PARAMETER IS INTRINSIC PROCEDURE (NOT ERROR)
621	ACTUAL PARAPART	TEMPORARY ABSENCE OF CODE FOR THIS (NOT ERROR)
622	ACTUALPARAPART	TYPE OF ACTUAL AND FORMAL PARAMS DONT AGREE
623	ACTUAL PARAPART	ILLEGAL PARAMETER DELIMITER

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
624	COMPOUNDTAIL	MISSING SEMICOLON OR END
625	COMPOUNDTAIL	EXTRA END
626	COMPOUNDTAIL	MISSING END
627	QUEUEDEC	INTERRUPTEDEC TRYING TO GO BEYOND LEVEL 31
628	QALGORITHM	THIS ALGORITHM IS NOT VALID FOR THIS QUEUE
629	QALGORITHM	MISSING ACTUAL PARAMETER PART IN EXPLICIT CALL
630	QALGORITHM	FIRST ACTUAL PARAMETER OF QALGORITHM IS ILLEGAL
631	FORSTMT	ILLEGAL FOR CLAUSE
632	FORSTMT	ILLEGAL EXPRESSION TYPE
633	QALGORITHM	TOO MANY ACTUAL PARAMETERS
634	QALGORITHM	ILLEGAL PARAMETER DELIMITER
635	QALGORITHM	TOO MANY ACTUAL PARAMETERS
636	REPLACESTMT	POINTER IDENTIFIER REQUIRED
637	REPLACESTMT	MISSING KEY WORD "BY"
638	REPLACESTMT	UPDATE SOURCE MUST BE POINTER OR SIMPLE VARIABLE
639	REPLACESTMT	POINTER SOURCE MUST BE UPDATED BY A POINTER OR WORD
640	REPLACESTMT	SOURCE MUST BE POINTER OR ARITHMETIC EXPRESSION
541	REPLACESTMT	CONDITION OR COUNT NEEDED
642	REPLACESTMT	UPDATED COUNT MUST BE FOLLOWED BY A CONDITION
643	SCANSTMT	POINTER IDENTIFIER REQUIRED
644	SCANSTMT	SIMPLE ARITHMETIC VARIABLE REQUIRED
645	SCANSTMT	CONDITION MISSING
646	SCANSTMT	RELATIONAL OPERATOR OR IN EXPECTED
647	SCANSTMT	STATEMENT TERMINATOR REQUIRED
648	QALGORITHM	BUSY AND SIZE MUST BE USED AS PRIMARIES
649	QALGORITHM	INCORRECT USAGE OF UNTYPED QUEUE ALGORITHM
650	SCANSTMT	IN EXPECTED

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
651	REPLACESTMT	TOO MANY (OR WRONG ORDER OF) ELEMENTS:SOURC OR ELEMENTS IN WRONG ORDER
652	REPLACESTMT	UPDATE COUNT MUST BE REAL OR INTEGER
653	REPLACESTMT	THIS CONSTRUCT REQUIRES AN ARITHMETIC SOURC
654	REPLACESTMT	RELATIONAL OP OR IN EXPECTED
655	REPLACESTMT	PICTURE IDENTIFIER REQUIRED
656	FILLSTMT	TOO MANY WORDS OF INITIAL VALUES
657	FILLSTMT	INITIAL VALUE MUST BE NUMBER OR STRING
658	FILLSTMT	MISSING RIGHT PARENTHESIS
659	FILLSTMT	ARRAY ROW REQUIRED
660	FILLSTMT	MISSING "WITH"
661	SWAPSTMT	MISSING LEFT PARENTHESIS
662	SWAPSTMT	ARRAYID OR SUBARRAY DESIGNATOR REQUIRED
663	SWAPSTMT	MISSING COMMA
664	SWAPSTMT	MISSING RIGHT PARENTHESIS
665	SWAPSTMT	NUMBER OF UNSPECIFIED SUBSCRIPTS MUST AGREE
666	THRUSTMT	MISSING DO IN A THRU CLAUSE
667	FORSTMT	MISSING UNTIL FOLLOWING A BY ELEMENT
668	FORSTMT	CONTROL VARIABLE IS NOT SIMPLE IN BY ELEMENT LIST
669	QALGORITHM	BUZZ IS UNTYPED
670	EVENTINTRINSIC	THE PARAMETER MUST BE AN EVENT DESIGNATOR
671	EVENTINTRINSIC	THE PARAMETER MUST BE AN INTERRUPT ID
672	EVENTINTRINSIC	MISSING LEFT OR RIGHT PARENTHESIS
673	CLOSESTMT	UNRECOGNIZED PARAMETER
674	FORMATANDLIST	ILLEGAL CONSTRUCT OF DIRECT READ-WRITE STMT
675	READWRITESTMT	EVENT LEVEL GTR THAN DIRECT ARRAY LEVEL
676	READWRITESTMT	SEEK STMT MUST HAVE ADDRESS PART
677	READWR-SPACE	SEEK OR SPACE MUST HAVE NON-DIRECT FILE
678	READWRITESTMT	SEEK MAY NOT HAVE FORMAT
679	READWRITESTMT	DIRECT I-O CANNOT BE USED AS PRIMARY
680	EVENTINTRINSIC	SECONDWORD EXPECTS EVENT OR DP PARAMETER
681	BLOCK	LEVEL 1 BLOCK NOT ALLOWED

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
682	SPACESTMT	SPACE MAY NOT HAVE BRACKET PART
683	EMITDB16	LABEL NOT ON HALF WORD BOUND(USE "::")
684		MISSING "(" IN CALL ON TRANSFER FUNCTION, INTRINSIC FUNCTION, OR PROCEDURE
685	INTRNSC	MISSING COMMA AFTER PARAMETER
686		MISSING ")" IN AN EXPRESSION OR A FUNCTION FUNCTION CALL PARAMETER LIST
687	ONSTMT	UNRECOGNIZED FAULT IDENTIFIER
690		ATTACHSTMT-DETACHSTMT-INTERRUPTDEC) INTERRUPT IDENTIFIER REQUIRED
691	ATTCHSTMT	MISSING "TO"
692	ATTCHSTMT	EVENT DESIGNATOR REQUIRED
694	FILEPART	ILLEGAL FILE PART
700	IDLIST	THIS ID ALREADY DECLARED IN THIS BLOCK
701	MERRIMAC	PARENTHESES PROBLEM
702	MERRIMAC	INCORRECT TYPE OR PROCEDURE OR MONITORED ITEM
703	FIELDPART	INCORRECT "FIELD PART"
704	FIELDER	INCORRECT "FIELD"-VALUE OR SYNTAX
705	LAYOUTDEC	INCORRECT "LAYOUT PART"
706	LAYOUTDEC	"FIELD VALUE" IS NOT UNSIGNED INTEGER
707	LAYOUTDEC	ILLEGAL LAYOUT PART
708	PUTOGETHER	TOTAL ALPHA LONGER THAN 2047 CHARACTERS
710	DEFINEDEC	EQUAL SIGN EXPECTED
711	QUEUEDEC	ILLEGAL MULTIPLE USE OF IDENTIFIER
712	QUEUEDEC	MISSING ENTRY DESCRIPTION
713	QUEUEDEC	MULTIPLE USE OF SAME ALGORITHM ID IN QUEUE DEC
714	QUEUEDEC	INCORRECT USAGE OF STANDARD QUEUE ALGORITHM
715	QUEUEDEC	THIS CONTRUCT NOT CODED AS YET (NOT ERROR)
717	ARRAY DEC	THIS CONTRUCT NOT CODED AS YET (NOT ERROR)
718	READONLYARRAYDEC	ARRAY WORD MISSING
719	READONLYARRAYDEC	IDENTIFIER DECLARED BEFORE

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
720	INITIALIZEARRAY	MISSING LEFT PARENTHESIS
721	INITIALIZEARRAY	NOT A NUMBER OR A LOGICAL VALUE
722	INITIALIZEARRAY	TOO BIG AN INTEGER
723	INITIALIZEARRAY	MISSING RIGHT PARENTHESIS
724	ARRAYDEC	IDENTIFIER DECLARED BEFORE
725	ARRAYDEC	MORE THAN ONE IDENTIFIER DECLARED PREVIOUSL
726	ARRAYDEC	MISSING LEFT BRACKET
727	ARRAYDEC	ANY OF SEVERAL SYNTAX ERRORS IN AN ARRAY DECLARATION
728	ARRAYDEC	MISSING RIGHT BRACKET
729	ARRAYDEC	MISSING SEMICOLON
730	PROCEDUREDEC	PROCEDURE TYPE DIFFERS FROM FORWARD DECLARATION
731	PROCEDUREDEC	PROCEDURE INDENTIFIER ALREADY USED IN THIS BLOCK
732	PROCEDUREDEC	A PARAMETER WAS NOT SPECIFIED
733	PROCEDUREDEC	NUMBER OF PARAMETERS DIFFERS FROM FORWARD DEC
734	PROCEDUREDEC	SPECIFICATION DIFFERS FROM FORWARD DECLARATION
735	PROCEDUREDEC	THIS PROCEDURE WAS ALREADY DECLARED FORWARD
736	PROCEDUREDEC	MISSING SEMICOLON OR "(" AFTER PROCEDURE IDENTIFIER
737	FMLPARAPART	ILLEGAL PARAMETER DELIMITER
738	FMLPARAPART	MISSING SEMICOLON AFTER FORMAL PARAMETER LIST
739	FMLPARAPART	NOT VALID PARAMETER IDENTIFIER
740	FMLPARAPART	MISSING SEMICOLON IN SPECIFICATION PART
741	FMLPARAPART	ILLEGAL BOUND SPECIFIER
742	FMLPARAPART	TOO MANY ":"S
743	FMLPARAPART	ID NOT FORMAL, OR ALREADY SPECIFIED
744	FMLPARAPART	MISSING "[" IN ARRAY SPECIFICATION
745	FMLPARAPART	ILLEGAL BOUND SPECIFIER

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
746	FMLPARAPART	MISSING "]" IN ARRAY SPECIFICATION
747	DEFINEDEC	PARAMETER ERROR
748	DEFINEDEC	MORE THAN 9 DEFINE PARAMETERS
749	DEFINEDEC	MISSING ")"
750	QUEUEDEC	MORE THAN 32767 ITEMS
751	QUEUEDEC	INCORRECT SYNTAX FOR LOCKING SPECIFICATION
752	QUEUEDEC	MISSING RIGHT BRACKET IN SIZE SPECIFICATION
753	QUEUEDEC	ILLEGAL SYNTAX FOR QUEUE ARRAY BOUND
756	PICTUREDEC	ILLEGAL REPEAT PART VALUE
757	PICTUREDEC	MISSING ")" IN REPEAT PART
758	PICTUREDEC	PICTURE ID ALREADY USED IN THIS BLOCK
759	PICTUREDEC	MISSING "(" AFTER PICTURE ID
760	PICTUREDEC	ILLEGAL PICTURE CHARACTER
761	TRUTHSETDEC	INVALID TRUTHSET EXPRESSION
762	TRANSTBLDEC	INVALID TRANSLATE TABLE
770	FILEDEC	ILLEGAL FILE ATTRIBUTE
771	FILEDEC	INTRINSIC MAY NOT HAVE A FILE DECLARATION
772	FILEDEC	FILES MAY NOT BE DECLARED AT LEVEL 0
773	FILEDEC	STRING OR CONSTANT REQUIRED FOR THIS ATTRIB
774	FILEDEC	CONSTANT EXPRESSION REQUIRED FOR THIS ATTRI
775	FILEDEC	ILLEGAL VALUE FOR THIS ATTRIBUTE
782	IDLIST	INITIAL VALUE OWN VARIABLES NOT IMPLEMENTED
783	IDLIST	EVENT MAY NOT HAVE INITIAL VALUE
784	FMLPARAPART	THIS ID HAS APPEARED IN THE VALUE PART
785	INTERRUPTDEC	MISSING EVENT DESIGNATOR
786	INTERRUPTDEC	MISSING "ON" OR COLON
787	INTERRUPTDEC	MISSING COMMA
788	PROCEDUREDEC	PROCEDURE FOR SEPARATED COMPILING CANNOT BE EXTERNAL
789	PROCEDUREDEC	PROCEDURE FOR SEPARATED COMPILING CANNOT BE FORWARD
790	PROCEDUREDEC	GLOBAL PROCEDURE SHOULD NOT HAVE A BODY
791	PROCEDUREDEC	TOO MANY GLOBAL DECLARATIONS

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
792	ENTER	SYMBOLIC NOT IN NORMAL FORM
800	DECLARATIONS	MISSING SEMICOLON AFTER DECLARATION
801	DECLARATIONS	ILLEGAL COMBINATION OF DECLARATORS
802	DECLARATIONS	"SAVE" MAY NOT BE USED THIS WAY
803	DECLARATIONS	"OWN" MAY NOT BE USED THIS WAY
804	STATEMENT	UNRECOGNIZABLE STATEMENT STARTER
805	STATEMENT	DECLARATIONS SHOULD PRECEDE STATEMENTS
806	STATEMENT	MISSING "UNTIL" IN DO STATEMENT
807	STATMENT	MISSING "DO" IN WHILE STATEMENT
808	PRIMARY	NO PRIMARY MAY START THIS WAY
809	PRIMARY	MISSING ")"
810	PRIMARY	COMPILER ERROR
811	PRIMARY	ILLEGAL TRANSFER TYPE
812	PRIMARY	MISSING "("
813	PRIMARY	ILLEGAL EXPR TYPE IN TRANSFER FUNCTION
814	PRIMARY	ILLEGAL SECOND EXPRESSION IN TRANSFER FUNCTION
815	PRIMARY	TIMER AND XSIGN MAY NOT BE READ
816	PRIMARY	WRONG TYPE EXPRESSION IN REGISTER ASSIGNMEN
817	BOOPRIM	NO PRIMARY MAY START LIKE THIS
818	BOOPRIM	COMPILER ERROR
819	BOOPRIM	MISSING ")"
820	BOOPRIM	MISSING "("
821	BOOPRIM	ILLEGAL EXPRESSION TYPE IN TRANSFER FUNCTIO
822	PTRPRIM	COMPILER ERROR
823	PTRPRIM	MISSING ")"
824	PTRPRIM	CASE EXPRESSION NOT POINTER TYPE
825	PTRPRIM	POINTER PRIMARY CANNOT START WITH THIS
826	MAKEPOINTER	MISSING "("
827	MAKEPOINTER	MUST BE ARRAY IDENTIFIER
828	MAKEPOINTER	MUST BE ARRAY ROW
829	MAKEPOINTER	MUST BE ONE-DIMENSIONAL ARRAY
830	MAKEPOINTER	MISSING ")"

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
831	SETPRIM	MISSING RIGHT PARENTHESIS
833	MAKEPOINTER	CHARACTER SIZE MUST BE LITERAL 4, 6, OR 8
834	BOOPRIM	EXPRESSION BEFORE "IN" MUST BE ARITHMETIC
835	BOOPRIM	SET IDENTIFIER REQUIRED AFTER "IN"
836	STATEMENT	HAPPENED AND AVAILABLE ARE BOOLEAN INTRINSICS
837	BOOPRIM	THIS EVENT INTRINSIC IS UNTYPED
838	DECLARATIONS	NOTHING BUT PROCEDURES CAN BE SEPARATELY COMPILED
839	DECLARATIONS	NO SAVE PROCEDURE CAN BE SEPARATELY COMPILED
840	DECLARATIONS	NO "OWN" DEC IS PERMITTED IN SEPAR. COMPILATION
841	DECLARATIONS	NO "VALUE" DEC IS PERMITTED IN SEPAR. COMPILATION
842	DECLARATIONS	SAVE N ALLOWED ONLY FOR N = 1
843	STATEMENT	FAULT STATEMENTS MAY NOT BE AT LEVEL 0.
850	GLOBALDECLARATIONS	ILLEGAL GLOBAL DECLARATION FOR SEPARATED COMPILE
851	GLOBALDECLARATIONS	MISSING SEMICOLON ON GLOBAL DECLARATION
852	GLOBALDECLARATIONS	ARRAY ID ALREADY DECLARED OR MISSING
853	GLOBALDECLARATIONS	MISSING "[" IN ARRAY DECLARATION
854	GLOBALDECLARATIONS	ILLEGAL BOUND SPECIFIER
855	GLOBALDECLARATIONS	MISSING "]" IN ARRAY DECLARATION
856	GLOBALDECLARATIONS	GLOBAL ID ALREADY DECLARED OR MISSING
857	GLOBALDECLARATIONS	TOO MANY GLOBAL DECLARATIONS
858	SEPARATEDCOMPILING	MISSING SEMICOLON AFTER LAST "END" OF PROCEDURE
860	STATEMENT	ILLEGAL VECTORMODE STATEMENT
861	PROCESSVECTORMODE	INVALID VECTORMODE CONSTRUCT
862	PROCESSVECTORMODE	VECTORMODE FOR COUNT REQUIRED
863	EMITTERS	INVALID VECTOR MODE REFERENCE
900	QUEUEDEC	MISSING SPECIFICATION IN FORMAL ITEM LIST

ERR NO.	ESPOL PROCEDURE	ERROR MESSAGE
921	GENMICRO	REPEAT > 65535
922	PICTUREDEC	EXTRANEOUS IN PICTURE (INVALID REPEAT PART)
923	REPLACEMENT	MISSING ")"
924	PICTUREGEN	MISSING ","
925	PICTUREGEN	MISSING EXPRESSION
926	PICTUREGEN	MISSING ")"

```
                BEGIN
                REAL X, Y, Z;
X=(00,00C0)
Y=(00,00C1)
Z=(00,00C2)
                WORD WRD;
WRD=(00,00C3)
                    NAME REAL NAMEREAL = WRD;
NAMEREAL=(00,00C3)
                    NAME ARRAY NAMEARRAY = WRD [*];
NAMEARRAY=(00,00C3)
                WORD ARRAY XRAY [*];
XRAY=(00,00C4)
                EVENT EVNT1, EVNT2;
EVNT1=(00,00C5)
EVNT2=(00,00C7)
                    ARRAY BUF1[*],
BUF1=(00,00C9)
                          BUF2[24];
BUF2=(00,00CA)
                    SAVE ARRAY BUF3[5],
BUF3=(00,00CB)
                                BUF4[24];
BUF4=(00,00CC)
                    POINTER POINT, POINTX;
POINT=(00,00CD)
POINTX=(00,00CE)
                XRAY [3] := NULL;
                                005:0000:0  PUSH                    B4
                                005:0000:1  PUSH                    B4
                                005:0000:2  PUSH                    B4
                                005:0000:3  LTA                     B203
                                005:0000:5  NAMC   (00,000C4)       40C4
                                005:0001:1  INDX                    A6
                                005:0001:2  NAMC   (00,00091)       4091
                                005:0001:4  LOAD                    BD
                                005:0001:5  EXCH                    B6
                                005:0002:0  OVRD                    BA
                X := NAMEREAL;
                                005:0002:1  VALC   (00,000C3)       00C3
                                005:0002:3  NAMC   (00,000C0)       40C0
                                005:0002:5  STOD                    B8
                X := NAMEARRAY [3];
                                005:0003:0  LTA                     B203
                                005:0003:2  NAMC   (00,000C3)       40C3
                                005:0003:4  EVAL                    AC
                                005:0003:5  LOAD                    BD
                                005:0004:0  NXLV                    AD
                                005:0004:1  NAMC   (00,000C0)       40C0
                                005:0004:3  STOD                    B8
                M[3] := WRD;
                                005:0004:4  LTA                     B203
                                005:0005:0  NAMC   (00,00004)       4004
                                005:0005:2  INDX                    A6
                                005:0005:3  NAMC   (00,000C3)       40C3
                                005:0005:5  LODT                    95BC
                                005:0006:1  EXCH                    B6
                                005:0006:2  OVRD                    BA
                WRD := M[3];
                                005:0006:3  LTA                     B203
                                005:0006:5  LODT                    95BC
                                005:0007:1  NAMC   (00,000C3)       40C3
                                005:0007:3  OVRD                    BA
                ************* ITERATION CLAUSES AND STATEMENTS ****
                THRU 255 DO ;
                                005:0007:4  LT48                    BE
                                005:0008                            00100FF00000 0001001774000000

                                005:0009:0  LTA                     B204
                                005:0009:2  STAG                    95B4
                                005:0009:4  NAMC   (00,000CF)       40CF
                                005:000A:0  OVRD                    BA
```

```
                        005:000A:1   NAMC  (00,000CF)    40CF
                        005:000A:3   LINK   000A:3       A40000
                        005:000B:0   DLET                B5
                        005:000B:1   BRUN   000A:1       A2200A
                        005:000A:3   STBR   000B:4       A4800B
                THRU X - 1 DO;
                        005:000B:4   VALC  (00,000C0)    00C0
                        005:000C:0   ONE                 B1
                        005:000C:1   SUBT                81
TEMP   =(00,00CF)
                        005:000C:5   VALC  (00,000CF)    00CF
                        005:000D:1   ONE                 B1
                        005:000D:2   SUBT                81
                        005:000C:2   BRUN   000D:3       A2600D
                        005:000D:3   NAMC  (00,000CF)    40CF
                        005:000D:5   OVRN                B8
                        005:000E:0   ZERO                B0
                        005:000E:1   LSEQ                B8
                        005:000E:2   BRFL   000C:5       A0A00C
                % ****** INTRINSICS ***************************************************
                BEGIN
                X := ABS (X);                                                        2
                        005:000E:5   VALC  (00,000C0)    00C0
                        005:000F:1   BRST                9E2E
                        005:000F:3   NAMC  (00,000C0)    40C0
                        005:000F:5   STOD                B8
                IF AVAILABLE (EVNT1) THEN;
                        005:0010:0   VALC  (00,000C5)    00C5
                        005:0010:2   LNOT                92
                        005:0010:3   ISOL                9A0201
                        005:0011:0   DLET                B5
                IF BUSY (X) THEN;
                        005:0011:1   VALC  (00,000C0)    00C0
                        005:0011:3   DLET                B5
                BUZZ(X);
                        005:0011:4   DEXI                9547
                        005:0012:0   ZERO                B0
                        005:0012:1   DLET                B5
                        005:0012:2   NAMC  (00,000C0)    40C0
                        005:0012:4   ONE                 B1
                        005:0012:5   RDLK                958A
                        005:0013:1   DUPL                B7
                        005:0013:2   BRFL   0015:2       A04015
                        005:0013:5   ZERO                B0
                        005:0014:0   LOG2                958B
                        005:0014:2   LOG2                958B
                        005:0014:4   DLET                B5
                        005:0014:5   BRUN   0012:1       A22012
                        005:0015:2   DLET                B5
                 $ SET READLOCK
                BUZZ(X);
                        005:0015:3   DEXI                9547
                        005:0015:5   LTB                 B220
                        005:0016:1   RPRR                95BA
                        005:0016:3   ISOL                9A0F24
                        005:0017:0   LTB                 B236
                        005:0017:2   RPRR                95B8
                        005:0017:4   FLTR                98130F10
                        005:0018:2   LTB                 B235
                        005:0018:4   RPRR                95B8
                        005:0019:0   FLTR                982F0B0C
                        005:0019:4   ONE                 B1
                        005:0019:5   LOR                 91
                        005:001A:0   DUPL                B7
                        005:001A:1   NAMC  (00,000C0)    40C0
                        005:001A:3   DUPL                B7
                        005:001A:4   RSUP                95B6
                        005:001B:0   RDLK                95BA
                        005:001B:2   DUPL                B7
                        005:001B:3   BRFL   0023:3       A06023
                        005:001C:0   RSDN                95B7
                        005:001C:2   EXCH                B6
                        005:001C:3   DLET                B5
                        005:001C:4   DUPL                B7
                        005:001C:5   LTB                 B220
                        005:001D:1   RPRR                95B8
                        005:001D:3   ISOL                9A0F24
                        005:001E:0   LTB                 B236
                        005:001E:2   RPRR                95B8
                        005:001E:4   FLTR                98130F10
                        005:001F:2   LTB                 B235
```

```
                005:001F:4  RPRR                    958A
                005:0020:0  FLTR                    982F0B0C
                005:0020:4  ONE                     B1
                005:0020:5  LOR                     91
                005:0021:0  RDLK                    95BA
                005:0021:2  DUPL                    B7
                005:0021:3  BRFL    0023:3          A06023
                005:0022:0  ZERO                    B0
                005:0022:1  LOG2                    958B
                005:0022:3  LOG2                    958B
                005:0022:5  DLET                    B5
                005:0023:0  BRUN    001C:3          A2601C
                005:0023:3  RSDN                    9587
                005:0023:5  DLET                    B5
                005:0024:0  DLET                    B5
                005:0024:1  DLET                    B5
 $ POP READLOCK
X := 1 + DUPLICATE;
                005:0024:2  ONE                     B1
                005:0024:3  DUPL                    B7
                005:0024:4  ADD                     80
                005:0024:5  NAMC    (00,000C0)      40C0
                005:0025:1  STOD                    B8
X := ENTIER (X);
                005:0025:1  STON                    B9
                005:0025:2  XTND                    CE
                005:0025:3  LTR                     B29B
                005:0025:5  ISOL                    9A0930
                005:0026:2  SUBT                    81
                005:0026:3  NTGR                    B7
                005:0026:4  NAMC    (00,000C0)      40C0
                005:0027:0  STOD                    B8
EXCHANGE(X, WRD);
                005:0027:0  STON                    B9
                005:0027:1  NAMC    (00,000C3)      40C3
                005:0027:3  RDLK                    95BA
                005:0027:5  DLET                    B5
X := EXCHANGE (X, WRD);
                005:0028:0  VALC    (00,000C0)      00C0
                005:0028:2  NAMC    (00,000C3)      40C3
                005:0028:4  RDLK                    95BA
                005:0029:0  NAMC    (00,000C0)      40C0
                005:0029:2  STOD                    B8
WRD := EXCHANGE(X, WRD);
                005:0029:2  STON                    B9
                005:0029:3  NAMC    (00,000C3)      40C3
                005:0029:5  RDLK                    95BA
                005:002A:1  NAMC    (00,000C3)      40C3
                005:002A:3  OVRD                    BA
EXCHANGE(WRD, X);
                005:002A:3  OVRN                    BB
                005:002A:4  NAMC    (00,000C0)      40C0
                005:002B:0  RDLK                    95BA
                005:002B:2  DLET                    B5
X := EXCHANGE (WRD, X);
                005:002B:3  NAMC    (00,000C3)      40C3
                005:002B:5  LODT                    95BC
                005:002C:1  NAMC    (00,000C0)      40C0
                005:002C:3  RDLK                    95BA
                005:002C:5  NAMC    (00,000C0)      40C0
                005:002D:1  STOD                    B8
WRD := EXCHANGE(WRD, X);
                005:002D:2  NAMC    (00,000C3)      40C3
                005:002D:4  LODT                    95BC
                005:002E:0  NAMC    (00,000C0)      40C0
                005:002E:2  RDLK                    95BA
                005:002E:4  NAMC    (00,000C3)      40C3
                005:002F:0  OVRD                    BA
EXCHANGE (X);
                005:002F:1  VALC    (00,000C0)      00C0
                005:002F:3  EXCH                    B6
                005:002F:4  DLET                    B5
EXCHANGE (1);
                005:002F:5  ONE                     B1
                005:0030:0  EXCH                    B6
                005:0030:1  DLET                    B5
X := EXCHANGE(1);
                005:0030:2  ONE                     B1
                005:0030:3  EXCH                    B6
                005:0030:4  NAMC    (00,000C0)      40C0
                005:0031:0  STOD                    B8
```

```
IF HAPPENED (EVNT1) THEN;
                005:0031:1  VALC  (00,000C5)    00C5
                005:0031:3  ISNL                9A0001
                005:0031:3  DLET                B5
X := INTEGER (PNINT, 3);
                005:0031:4  NAMC  (00,000CD)    40CD
                005:0032:0  LOAD                BD
                005:0032:1  LT8                 B203
                005:0032:3  ICVD                CA
                005:0032:4  NTGR                87
                005:0032:5  NAMC  (00,000C0)    40C0
                005:0033:1  STND                B8
X := LISTLOOKUP (X, XRAY[*], X);
                005:0033:1  STON                B9
                005:0033:2  NAMC  (00,000C4)    40C4
                005:0033:4  LOAD                BD
                005:0033:5  VALC  (00,000C0)    00C0
                005:0034:1  LLLU                95BD
                005:0034:3  NAMC  (00,000C0)    40C0
                005:0034:5  STND                B8
X := MASKSEARCH (X, X, XRAY[*]);
                005:0034:5  STON                B9
                005:0035:0  DUPL                B7
                005:0035:1  NAMC  (00,000C4)    40C4
                005:0035:3  LOAD                BD
                005:0035:4  SRCH                95BE
                005:0036:0  NAMC  (00,000C0)    40C0
                005:0036:2  STND                B8
Z := MAX(X, Y, Z);
                005:0036:2  STON                B9
                005:0036:3  DUPL                B7
                005:0036:4  VALC  (00,000C1)    00C1
                005:0037:0  DUPL                B7
                005:0037:1  RSON                95B7
                005:0037:3  GREQ                B9
                005:0037:4  BRTR   003A:2       A14038
                005:0038:1  EXCH                B6
                005:0038:2  DLET                B5
                005:0038:3  DUPL                B7
                005:0038:4  VALC  (00,000C2)    00C2
                005:0039:0  DUPL                B7
                005:0039:1  RSON                95B7
                005:0039:3  GREQ                B9
                005:0039:4  BRTR   003A:2       A1403A
                005:003A:1  EXCH                B6
                005:003A:2  DLET                B5
                005:003A:3  NAMC  (00,000C2)    40C2
                005:003A:5  STND                B8
Z := MIN(X, Y, Z);
                005:003B:0  VALC  (00,000C0)    00C0
                005:003B:2  DUPL                B7
                005:003B:3  VALC  (00,000C1)    00C1
                005:003B:5  DUPL                B7
                005:003C:0  RSON                95B7
                005:003C:2  LSEQ                BA
                005:003C:3  BRTR   003D:1       A1203D
                005:003D:0  EXCH                B6
                005:003D:1  DLET                B5
                005:003D:2  DUPL                B7
                005:003D:3  VALC  (00,000C2)    00C2
                005:003D:5  DUPL                B7
                005:003E:0  RSON                95B7
                005:003E:2  LSEQ                BA
                005:003E:3  BRTR   003F:1       A1203F
                005:003F:0  EXCH                B6
                005:003F:1  DLET                B5
                005:003F:2  NAMC  (00,000C2)    40C2
                005:003F:4  STND                B8
X := NABS (X);
                005:003F:5  VALC  (00,000C0)    00C0
                005:0040:1  BSET                962E
                005:0040:3  NAMC  (00,000C0)    40C0
                005:0040:5  STND                B8
X := NORMALIZE (X);
                005:0040:5  STON                B9
                005:0041:0  SNGL                CD
                005:0041:1  NAMC  (00,000C0)    40C0
                005:0041:3  STND                B8
Y := OCRX (1, X);
                005:0041:4  ONE                 B1
                005:0041:5  VALC  (00,000C0)    00C0
```

```
                005:0042:1  OCRX                    9585
                005:0042:3  NAMC  (00,000C1)        40C1
                005:0042:5  STOD                    B8
X := REAL (POINT, 1);
                005:0043:0  NAMC  (00,000CD)        40CD
                005:0043:2  LOAD                    BD
                005:0043:3  ONE                     B1
                005:0043:4  SISD                    D5
                005:0043:5  NAMC  (00,000C0)        40C0
                005:0044:1  STOD                    B8
STOP;
                005:0044:2  HALT                    DF
STOP (3);
                005:0044:3  LTA                     B203
                005:0044:5  ZERO                    B0
                005:0045:0  EXCH                    B6
                005:0045:1  HALT                    DF
                005:0045:2  DLET                    B5
                005:0045:3  DLET                    B5
STOP (3, 3);
                005:0045:4  LTA                     B203
                005:0046:0  LTA                     B203
                005:0046:2  EXCH                    B6
                005:0046:3  HALT                    DF
                005:0046:4  DLET                    B5
                005:0046:5  DLET                    B5
UNLOCK (X);
                005:0047:0  NAMC  (00,000C0)        40C0
                005:0047:2  ZERO                    B0
                005:0047:3  STOD                    B8
ZAP;
                005:0047:4  PTPA                    954D
END OF INTRINSICSTUFF;
  ****** EVENT STATEMENTS **********************
BEGIN
SET (EVNT1);
                005:0048:0  MKST                    AE
                005:0048:1  NAMC  (00,00099)        4099
                005:0048:3  NAMC  (00,000C5)        40C5
                005:0048:5  STFF                    AF
                005:0049:0  ONE                     B1
                005:0049:1  ENTR                    A8
RESET (EVNT1);
                005:0049:2  MKST                    AE
                005:0049:3  NAMC  (00,00099)        4099
                005:0049:5  NAMC  (00,000C5)        40C5
                005:004A:1  STFF                    AF
                005:004A:2  ZERO                    B0
                005:004A:3  ENTR                    A8
CAUSE (EVNT1);
                005:004A:4  MKST                    AE
                005:004A:5  NAMC  (00,00096)        4096
                005:004B:1  NAMC  (00,000C5)        40C5
                005:004B:3  STFF                    AF
                005:004B:4  ONE                     B1
                005:004B:5  ENTR                    A8
CAUSENRESET (EVNT1);
                005:004C:0  MKST                    AE
                005:004C:1  NAMC  (00,00096)        4096
                005:004C:3  NAMC  (00,000C5)        40C5
                005:004C:5  STFF                    AF
                005:004D:0  ZERO                    B0
                005:004D:1  ENTR                    A8
WAIT (EVNT1);
                005:004D:2  MKST                    AE
                005:004D:5  NAMC  (00,000C5)        40C5
                005:004E:1  STFF                    AF
                005:004E:2  ZERO                    B0
                005:004D:3  NAMC  (00,00098)        4098
                005:004E:3  ENTR                    A8
WAIT ((5));
                005:004E:4  MKST                    AE
                005:004F:1  LTA                     B205
                005:004E:5  NAMC  (00,00013)        4013
                005:004F:3  ZERO                    B0
                005:004F:4  ENTR                    A8
WAIT ((3), EVNT1, EVNT2);
                005:004F:5  MKST                    AE
                005:0050:2  LTA                     B203
                005:0050:4  ZERO                    B0
                005:0050:0  NAMC  (00,0009F)        409F
```

```
                005:0050:5  NAMC  (00,000C5)   40C5
                005:0051:1  STFF               AF
                005:0051:2  ZERO               B0
                005:0051:3  NAMC  (00,000C7)   40C7
                005:0051:5  STFF               AF
                005:0052:0  ZERO               B0
                005:0052:1  ENTR               AB
                005:0052:2  DLET               B5
WAITANDRESET (EVNT1);
                005:0052:3  MKST               AE
                005:0053:0  NAMC  (00,000C5)   40C5
                005:0053:2  STFF               AF
                005:0053:3  ONE                B1
                005:0052:4  NAMC  (00,00098)   4098
                005:0053:4  ENTR               AB
WAITANDRESET ((3), EVNT1, EVNT2);
                005:0053:5  MKST               AE
                005:0054:2  LTA                B203
                005:0054:4  ONE                B1
                005:0054:0  NAMC  (00,0009F)   409F
                005:0054:5  NAMC  (00,000C5)   40C5
                005:0055:1  STFF               AF
                005:0055:2  ONE                B1
                005:0055:3  NAMC  (00,000C7)   40C7
                005:0055:5  STFF               AF
                005:0056:0  ONE                B1
                005:0056:1  ENTR               AB
                005:0056:2  DLET               B5
DSWAIT (EVNT1);
                005:0056:3  MKST               AE
                005:0057:0  NAMC  (00,000C5)   40C5
                005:0057:2  STFF               AF
                005:0057:3  ZERO               B0
                005:0057:4  CHSN               8E
                005:0056:4  NAMC  (00,00098)   4098
                005:0057:5  ENTR               AB
DSWAIT ((3));
                005:0058:0  MKST               AE
                005:0058:3  LTA                B203
                005:0058:1  NAMC  (00,00013)   4013
                005:0058:5  ZERO               B0
                005:0059:0  ENTR               AB
DSWAIT ((3), EVNT1, EVNT2);
                005:0059:1  MKST               AE
                005:0059:4  LTA                B203
                005:005A:0  ZERO               B0
                005:005A:1  CHSN               8E
                005:0059:2  NAMC  (00,0009F)   409F
                005:005A:2  NAMC  (00,000C5)   40C5
                005:005A:4  STFF               AF
                005:005A:5  ZERO               B0
                005:005B:0  NAMC  (00,000C7)   40C7
                005:005B:2  STFF               AF
                005:005B:3  ZERO               B0
                005:005B:4  ENTR               AB
                005:005B:5  DLET               B5
DSWAITNRESET (EVNT1);
                005:005C:0  MKST               AE
                005:005C:3  NAMC  (00,000C5)   40C5
                005:005C:5  STFF               AF
                005:005D:0  ONE                B1
                005:005D:1  CHSN               8E
                005:005C:1  NAMC  (00,00098)   4098
                005:005D:2  ENTR               AB
DSWAITNRESET ((3), EVNT1, EVNT2);
                005:005D:3  MKST               AE
                005:005E:0  LTA                B203
                005:005E:2  ONE                B1
                005:005E:3  CHSN               8E
                005:005D:4  NAMC  (00,0009F)   409F
                005:005E:4  NAMC  (00,000C5)   40C5
                005:005F:0  STFF               AF
                005:005F:1  ONE                B1
                005:005F:2  NAMC  (00,000C7)   40C7
                005:005F:4  STFF               AF
                005:005F:5  ONE                B1
                005:0060:0  ENTR               AB
                005:0060:1  DLET               B5
PROCURE (EVNT1);
                005:0060:2  MKST               AE
                005:0060:3  NAMC  (00,0009A)   409A
```

```
                005:0060:5  NAMC  (00,000C5)    40C5
                005:0061:1  STFF                AF
                005:0061:2  ENTR                AB
FIX (EVNT1);
                005:0061:3  MKST                AE
                005:0061:4  NAMC  (00,0008E)    408E
                005:0062:0  NAMC  (00,000C5)    40C5
                005:0062:2  STFF                AF
                005:0062:3  ONE                 B1
                005:0062:4  ENTR                AB
                005:0062:5  DLFT                B5
LIBERATE (EVNT1);
                005:0063:0  MKST                AE
                005:0063:1  NAMC  (00,0009C)    409C
                005:0063:3  NAMC  (00,000C5)    40C5
                005:0063:5  STFF                AF
                005:0064:0  ENTR                AB
FREE (EVNT1);
                005:0064:1  MKST                AE
                005:0064:2  NAMC  (00,0008E)    408E
                005:0064:4  NAMC  (00,000C5)    40C5
                005:0065:0  STFF                AF
                005:0065:1  ZERO                B0
                005:0065:2  ENTR                AB
                005:0065:3  DLFT                B5
END OF EVENTSTATEMENTS;
  ****** POINTER EXPRESSIONS **********************
BEGIN
    POINT := POINTER(BUF1);
                005:0065:4  NAMC  (00,000C9)    40C9
                005:0066:0  LOAD                BD
                005:0066:1  NAMC  (00,000CD)    40CD
                005:0066:3  OVRD                BA
    POINT := POINTER(BUF2[3]);
                005:0066:4  LTB                 B203
                005:0067:0  NAMC  (00,000CA)    40CA
                005:0067:2  INDX                A6
                005:0067:3  NAMC  (00,000CD)    40CD
                005:0067:5  OVRD                BA
    POINT := POINTER(BUF3);
                005:0068:0  NAMC  (00,000CB)    40CB
                005:0068:2  LOAD                BD
                005:0068:3  NAMC  (00,000CD)    40CD
                005:0068:5  OVRD                BA
    POINT := POINTER(BUF4[3]);
                005:0069:0  LTB                 B203
                005:0069:2  NAMC  (00,000CC)    40CC
                005:0069:4  INDX                A6
                005:0069:5  NAMC  (00,000CD)    40CD
                005:006A:1  OVRD                BA
    POINT := POINTER(BUF1, *);
                005:006A:2  NAMC  (00,000C9)    40C9
                005:006A:4  LOAD                BD
                005:006A:5  NAMC  (00,000CD)    40CD
                005:006B:1  OVRD                BA
    POINT := POINTER(BUF1, 4);
                005:006B:2  NAMC  (00,000C9)    40C9
                005:006B:4  LOAD                BD
                005:006B:5  BSET                9629
                005:006C:1  NAMC  (00,000CD)    40CD
                005:006C:3  OVRD                BA
    POINT := POINTER(BUF2[3], 6);
                005:006C:4  LTB                 B203
                005:006D:0  NAMC  (00,000CA)    40CA
                005:006D:2  INDX                A6
                005:006D:3  BSET                9629
                005:006D:5  BSET                9628
                005:006E:1  NAMC  (00,000CD)    40CD
                005:006E:3  OVRD                BA
    POINT := POINTER(BUF3[3], 8);
                005:006E:4  LTB                 B203
                005:006F:0  NAMC  (00,000CB)    40CB
                005:006F:2  INDX                A6
                005:006F:3  BSET                962A
                005:006F:5  NAMC  (00,000CD)    40CD
                005:0070:1  OVRD                BA
POINT := POINT + 1;
                005:0070:2  NAMC  (00,000CD)    40CD
                005:0070:4  LOAD                BD
                005:0070:5  ONE                 B1
                005:0071:0  EXPU                DD
```

```
                005:0071:1   SFDC                    DA
                005:0071:2   NAMC   (00,000CD)       40CD
                005:0071:4   OVRD                    BA
POINT :=* + 1;
                005:0071:5   NAMC   (00,000CD)       40CD
                005:0072:1   DUPL                    B7
                005:0072:2   LOAD                    BD
                005:0072:3   ONE                     B1
                005:0072:4   EXPU                    DD
                005:0072:5   SFDC                    DA
                005:0073:0   EXCH                    B6
                005:0073:1   OVRD                    BA
POINT := POINTX;
                005:0073:2   NAMC   (00,000CE)       40CE
                005:0073:4   LOAD                    BD
                005:0073:5   NAMC   (00,000CD)       40CD
                005:0074:1   OVRD                    BA
END OF POINTEREXPRESSIONS;
  ******* MISCELLANEOUS CONSTRUCTS **************
CASE X OF
                005:0074:2   VALC   (00,000C0)       00C0
    BEGIN
    Y := 0;
                005:0075:4   ZERO                    B0
                005:0075:5   NAMC   (00,000C1)       40C1
                005:0076:1   STOD                    B8
                005:0076:2   LINK    0076:2          0002C2
    Y := 0;
                005:0076:5   ZERO                    B0
                005:0077:0   NAMC   (00,000C1)       40C1
                005:0077:2   STOD                    B8
                005:0077:3   LINK    0077:3          0102C9
    Y := 0;
                005:0078:0   ZERO                    B0
                005:0078:1   NAMC   (00,000C1)       40C1
                005:0078:3   STOD                    B8
                005:0078:4   LINK    0078:4          0202D0
    Y := 0;
                005:0079:1   ZERO                    B0
                005:0079:2   NAMC   (00,000C1)       40C1
                005:0079:4   STOD                    B8
                005:0079:5   LINK    0079:5          0302D7
    ;
    ;
    ;
    Z := 1;
                005:007A:2   ONE                     B1
                005:007A:3   NAMC   (00,000C2)       40C2
                005:007A:5   STOD                    B8
                005:007B:0   LINK    007B:0          0702DE
    Z := 1;
                005:007B:3   ONE                     B1
                005:007B:4   NAMC   (00,000C2)       40C2
                005:007C:0   STOD                    B8
                005:007C:1   LINK    007C:1          0802E5
    Z := 1;
                005:007C:4   ONE                     B1
                005:007C:5   NAMC   (00,000C2)       40C2
                005:007D:1   STOD                    B8
                005:007D:2   LINK    007D:2          0902EC
    END;
                005:007D:5   NVLD                    FF
                005:0074:4   LTA                     B2FC
                005:0075:0   ADD                     B0
                005:0075:1   DBUN                    AA
                005:007E:0   BRUN    0083:0          A20083
                005:007E:3   BRUN    0083:0          A20083
                005:007F:0   BRUN    0083:0          A20083
                005:007F:3   BRUN    0083:0          A20083
                005:0080:0   BRUN    0083:0          A20083
                005:0080:3   BRUN    0083:0          A20083
                005:0081:0   BRUN    0083:0          A20083
                005:0081:3   BRUN    0083:0          A20083
                005:0082:0   BRUN    0083:0          A20083
                005:0082:3   BRUN    0083:0          A20083
                005:007D:2   BRUN    0083:0          A20083
                005:0082:3   BRUN    007C:4          A2807C
                005:007C:1   BRUN    0083:0          A20083
                005:0082:0   BRUN    0078:3          A2607B
                005:007B:0   BRUN    0083:0          A20083
                005:0081:3   BRUN    007A:2          A2407A
```

```
005:0079:5  BRUN   0083:0    A20083
005:007F:3  BRUN   0079:1    A22079
005:0078:4  BRUN   0083:0    A20083
005:007F:0  BRUN   0078:0    A20078
005:0077:3  BRUN   0083:0    A20083
005:007E:3  BRUN   0076:5    A2A076
005:0076:2  BRUN   0083:0    A20083
005:007E:0  BRUN   0075:4    A28075
FND.
```

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

THIS APPENDIX CONTAINS IN LOGICAL ORDER THE COMPLETE SYNTAX FOR THE ESPOL LANGUAGE. THE NUMBER TO THE LEFT OF EACH SYNTACTICAL FORMULA IS THE LINE NUMBER REFERRED TO BY APPENDIX D. THE NUMBER(S) TO THE RIGHT REFERS TO OTHER LINE NUMBER(S) IN THIS APPENDIX THAT DEFINE THE VARIOUS ITEMS OF THE FORMULA.

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

```
1  <PROGRAM> ::= <BLOCK> . <SPACE>                                    2  3
                / <COMPOUND STATEMENT> . <SPACE>                     4  3
2  <BLOCK> ::= <BLOCK HEAD> ; <COMPOUND TAIL>                         5  6
3  <SPACE> ::= <SINGLE SPACE>                                         7
             / <SPACE> <SINGLE SPACE>                                3  7
4  <COMPOUND STATEMENT> ::= BEGIN <COMPOUND TAIL>                     6
5  <BLOCK HEAD> ::= BEGIN <DECLARATION>                               8
                  / <BLOCK HEAD> ; <DECLARATION>                     5  8
6  <COMPOUND TAIL> ::= <STATEMENT> END                                9
                     / <STATEMENT> ; <COMPOUND TAIL>                 9  6
7  <SINGLE SPACE> ::= SONE HORIZONTAL BLANK POSITION2
8  <DECLARATION> ::= <ARRAY DECLARATION>                              10
                   / <DEFINE DECLARATION>                           11
                   / <DEFINE INVOCATION>                            12
                   / <DIRECT ARRAY DECLARATION>                     13
                   / <EVENT DECLARATION>                            14
                   / <EVENT ARRAY DECLARATION>                      15
                   / <FIELD DECLARATION>                            16
                   / <FILE DECLARATION>                             17
                   / <INTERRUPT DECLARATION>                        18
                   / <LABEL DECLARATION>                            19
                   / <LAYOUT DECLARATION>                           20
                   / <MONITOR DECLARATION>                          21
                   / <PICTURE DECLARATION>                          22
                   / <PROCEDURE DECLARATION>                        23
                   / <QUEUE ARRAY DECLARATION>                      24
                   / <QUEUE DECLARATION>                            25
                   / <TYPE DECLARATION>                             26
                   / <VALUE ARRAY DECLARATION>                      27
9  <STATEMENT> ::= <CONDITIONAL STATEMENT>                            28
                 / <UNCONDITIONAL STATEMENT>                        29
10 <ARRAY DECLARATION> ::= <ARRAY KIND> ARRAY <ARRAY LIST>            30 31
11 <DEFINE DECLARATION> ::= DEFINE <DEFINITION LIST>                  32
12 <DEFINE INVOCATION> ::= <DEFINED IDENTIFIER> <ACTUAL TEXT PART>    33 34
13 <DIRECT ARRAY DECLARATION> ::= DIRECT <DIRECT ARRAY KIND> ARRAY    35
                                 <ARRAY LIST>                        31
14 <EVENT DECLARATION> ::= <EVENT IDENTIFIER LIST>                    36
15 <EVENT ARRAY DECLARATION> ::= EVENT ARRAY <EVENT SEGMENT LIST>     37
16 <FIELD DECLARATION> ::= FIELD <FIELD PART LIST>                    38
17 <FILE DECLARATION> ::= <DIRECT SPECIFIER> FILE <FILE LIST>         39 40
18 <INTERRUPT DECLARATION> ::= INTERRUPT <INTERRUPT SEGMENT>          41
```

```
19 <LABEL DECLARATION> ::= LABEL <LABEL LIST>                                  42
20 <LAYOUT DECLARATION> ::= LAYOUT <LAYOUT PART LIST>                          43
21 <MONITOR DECLARATION> ::= MONITOR <PROCEDURE IDENTIFIER> (                  44
                               <MONITOR LIST> )                                45
22 <PICTURE DECLARATION> ::= <SAVE OR OWN> PICTURE <PICTURE PART LIST>         46  47
23 <PROCEDURE DECLARATION> ::= <SAVE PART> <PROCEDURE TYPE> PROCEDURE          48  49
                                 <PROCEDURE HEADING> <PROCEDURE BODY>          50  51
24 <QUEUE ARRAY DECLARATION> ::= QUEUE ARRAY <QUEUE ARRAY HEAD> [              52
                                   <INDEX BOUND> ] <QUEUE BODY>                53  54
25 <QUEUE DECLARATION> ::= QUEUE <QUEUE HEAD> <QUEUE BODY>                     55  54
26 <TYPE DECLARATION> ::= <TYPE> <TYPE LIST>                                   56  57
                        / OWN <TYPE> <TYPE LIST>                               56  57
27 <VALUE ARRAY DECLARATION> ::= <ARRAY SAVE PART> <VALUE TYPE> VALUE          58  59
                                   ARRAY <VALUE ARRAY PART LIST>               60
28 <CONDITIONAL STATEMENT> ::= <IF CLAUSE> <STATEMENT>                         61   9
                             / <IF CLAUSE> <UNCONDITIONAL STATEMENT>           61  29
                               ELSE <CONDITIONAL STATEMENT>                    28
                             / <CONDITIONAL ITERATION>                         62
                             / <LABEL> : <CONDITIONAL STATEMENT>               63  28
29 <UNCONDITIONAL STATEMENT> ::= <EMPTY>                                       64
                               / <IF CLAUSE> <UNCONDITIONAL STATEMENT>         61  29
                                 ELSE <UNCONDITIONAL STATEMENT>                29
                               / <LABEL> : <UNCONDITIONAL STATEMENT>           63  29
                               / <ASSIGNMENT STATEMENT>                        65
                               / <BASIC STATEMENT>                             66
                               / <BLOCK>                                        2
                               / <COMPOUND STATEMENT>                           4
                               / <DEFINE INVOCATION>                           12
                               / <EVENT STATEMENT>                             67
                               / <FORK STATEMENT>                              68
                               / <FUNCTION DESIGNATOR>                         69
                               / <I/O STATEMENT>                               70
                               / / <INTERRUPT STATEMENT>                       71
                               / <ON STATEMENT>                                72
                               / <PROCEDURE STATEMENT>                         73
                               / <REPLACE STATEMENT>                           74
                               / <SCAN STATEMENT>                              75
                               / <UNCONDITIONAL ITERATION>                     76
30 <ARRAY KIND> ::= <EMPTY>                                                    64
                  / <LOCAL OR OWN TYPE>                                        77
                  / SAVE <LOCAL OR OWN TYPE>                                   77
```

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

```
31 <ARRAY LIST> ::= <ARRAY SEGMENT>                                          78
                  / <ARRAY LIST> , <ARRAY SEGMENT>                           31  78
                  / <INITIALIZED ARRAY>                                      79
                  / <ARRAY LIST> , <INITIALIZED ARRAY>                       31  79
32 <DEFINITION LIST> ::= <DEFINITION>                                        80
                       / <DEFINITION LIST> , <DEFINITION>                    32  80
33 <DEFINED IDENTIFIER> ::= <IDENTIFIER>                                     81
34 <ACTUAL TEXT PART> ::= <EMPTY>                                            64
                        / ( <CLOSED TEXT LIST> )                             82
                        / [ <CLOSED TEXT LIST> ]                             82
35 <DIRECT ARRAY KIND> ::= <EMPTY>                                           64
                         / <TYPE>                                            56
36 <EVENT IDENTIFIER LIST> ::= <EVENT IDENTIFIER> <ADDRESS PART>             83  84
                             / <EVENT LIST> , <EVENT IDENTIFIER>             85  83
                               <ADDRESS PART>                                84
37 <EVENT SEGMENT LIST> ::= <EVENT SEGMENT>                                  86
                          / <EVENT SEGMENT LIST> , <EVENT SEGMENT>           37  86
38 <FIELD PART LIST> ::= <FIELD PART>                                        87
                       / <FIELD PART LIST> , <FIELD PART>                    38  87
39 <DIRECT SPECIFIER> ::= <EMPTY>                                            64
                        / DIRECT
40 <FILE LIST> ::= <FILE LIST PART>                                          88
                 / <FILE LIST> , <FILE LIST PART>                            40  88
41 <INTERRUPT SEGMENT> ::= <INTERRUPT IDENTIFIER> ; <STATEMENT>              89   9
42 <LABEL LIST> ::= <LABEL IDENTIFIER>                                       90
                  / <LABEL LIST> , <LABEL IDENTIFIER>                        42  90
43 <LAYOUT PART LIST> ::= <LAYOUT PART>                                      91
                        / <LAYOUT PART LIST> , <LAYOUT PART>                 43  91
44 <PROCEDURE IDENTIFIER> ::= <IDENTIFIER>                                   81
45 <MONITOR LIST> ::= <MONITORED ITEM>                                       92
                    / <MONITOR LIST> , <MONITORED ITEM>                      45  92
46 <SAVE OR OWN> ::= SAVE
                   / OWN
47 <PICTURE PART LIST> ::= <PICTURE PART>                                    93
                         / <PICTURE PART LIST> , <PICTURE PART>              47  93
48 <SAVE PART> ::= <EMPTY>                                                   64
                 / SAVE
                 / SAVE 1
49 <PROCEDURE TYPE> ::= <EMPTY>                                              64
                      / <TYPE>                                               56
50 <PROCEDURE HEADING> ::= <PROCEDURE IDENTIFIER> <ADDRESS PART>             44  84
```

```
                          <FORMAL PARAMETER PART> ;                                  94
51  <PROCEDURE BODY> ::= <STATEMENT>                                                  9
                       / FORWARD
                       / EXTERNAL
                       / NULL
52  <QUEUE ARRAY HEAD> ::= <QUEUE ARRAY IDENTIFIER> <INTERNAL NAME PART>          95  96
53  <INDEX BOUND> ::= <ARITHMETIC EXPRESSION>                                     97
54  <QUEUE BODY> ::= <ENTRY DESCRIPTION> ; <ALGORITHM PART>                       98  99
55  <QUEUE HEAD> ::= <QUEUE IDENTIFIER> <INTERNAL NAME PART>                     100  96
56  <TYPE> ::= BOOLEAN
             / DOUBLE
             / EVENT
             / INTEGER
             / NAME
             / POINTER
             / REAL
             / REFERENCE
             / WORD
57  <TYPE LIST> ::= <TYPE PART>                                                   101
                  / <TYPE LIST> , <TYPE PART>                                   57 101
58  <ARRAY SAVE PART> ::= <EMPTY>                                                 64
                        / SAVE
59  <VALUE TYPE> ::= REAL
                   / INTEGER
                   / DOUBLE
                   / BOOLEAN
60  <VALUE ARRAY PART LIST> ::= <VALUE ARRAY IDENTIFIER>                          102
                                <REPLACEMENT OPERATOR> ( <CONSTANT LIST>          103 104
                                )
                              / <VALUE ARRAY IDENTIFIER> (                      102
                                <CONSTANT LIST> )                               104
61  <IF CLAUSE> ::= IF <BOOLEAN EXPRESSION> THEN                                 105
62  <CONDITIONAL ITERATION> ::= <ITERATION CLAUSE> <CONDITIONAL STATEMENT>        106  28
63  <LABEL> ::= <LABEL IDENTIFIER>                                                90
64  <EMPTY> ::= ¢THE NULL STRING OF CHARACTERS¢
65  <ASSIGNMENT STATEMENT> ::= <ARITHMETIC ASSIGNMENT STATEMENT>                  107
                             / <ARRAY ASSIGNMENT STATEMENT>                     108
                             / <BOOLEAN ASSIGNMENT STATEMENT>                   109
                             / <QUEUE ASSIGNMENT>                               110
                             / <REFERENCE ASSIGNMENT>                           111
                             / <WORD ASSIGNMENT>                                112
```

```
66 <BASIC STATEMENT> ::= <GO TO STATEMENT>                                    113
                       / <DUMMY STATEMENT>                                    114
                       / <CASE STATEMENT>                                     115
67 <EVENT STATEMENT> ::= <SET STATEMENT>                                      116
                       / <RESET STATEMENT>                                    117
                       / <CAUSE STATEMENT>                                    118
                       / <CAUSEANDRESET STATEMENT>                            119
                       / <WAIT STATEMENT>                                     120
                       / <WAITANDRESET STATEMENT>                             121
                       / <DSWAIT STATEMENT>                                   122
                       / <DSWAITANDRESET STATEMENT>                           123
                       / <PROCURE STATEMENT>                                  124
                       / <FIX STATEMENT>                                      125
                       / <LIBERATE STATEMENT>                                 126
                       / <FREE STATEMENT>                                     127
68 <FORK STATEMENT> ::= FORK <PROCEDURE STATEMENT> [ <FORK PARAMETERS> ]       73 128
69 <FUNCTION DESIGNATOR> ::= <FUNCTION IDENTIFIER>                            129
                               <ACTUAL PARAMETER PART>                        130
70 <I/O STATEMENT> ::= <CLOSE STATEMENT>                                      131
                     / <LOCK STATEMENT>                                       132
                     / <READ STATEMENT>                                       133
                     / <REWIND STATEMENT>                                     134
                     / <SEEK STATEMENT>                                       135
                     / <SPACE STATEMENT>                                      136
                     / <WRITE STATEMENT>                                      137
71 <INTERRUPT STATEMENT> ::= <ATTACH STATEMENT>                               138
                           / <DETACH STATEMENT>                               139
                           / <ENABLE STATEMENT>                               140
                           / <DISABLE STATEMENT>                              141
72 <ON STATEMENT> ::= ON <FAULT NAME>                                         142
                    / ON <FAULT NAME> , <STATEMENT>                           142   9
73 <PROCEDURE STATEMENT> ::= <PROCEDURE IDENTIFIER>                            44
                               <ACTUAL PARAMETER PART>                        130
74 <REPLACE STATEMENT> ::= REPLACE <DESTINATION> BY <SOURCE LIST>             143 144
75 <SCAN STATEMENT> ::= SCAN <SOURCE> <SCAN PART>                             145 146
76 <UNCONDITIONAL ITERATION> ::= <DO STATEMENT>                               147
                               / <ITERATION CLAUSE>                           106
                                 <UNCONDITIONAL STATEMENT>                     29
77 <LOCAL OR OWN TYPE> ::= <TYPE>                                              56
                         / OWN <TYPE>                                          56
78 <ARRAY SEGMENT> ::= <ARRAY IDENTIFIER> <ADDRESS PART> [ <BOUND LIST> ]     148   84 14
```

```
                    / <ARRAY IDENTIFIER> <ADDRESS PART> ,                148  84
                      <ARRAY SEGMENT>                                    78
79 <INITIALIZED ARRAY> ::= <ARRAY IDENTIFIER> <REPLACEMENT OPERATOR> (   148 103
                           <CONSTANT LIST> )                           104
80 <DEFINITION> ::= <DEFINED IDENTIFIER> <FORMAL SYMBOL PART> = <TEXT> #  33 150 151
81 <IDENTIFIER> ::= <LETTER>                                           152
                  / <IDENTIFIER> <LETTER>                               81 152
                  / <IDENTIFIER> <DIGIT>                                81 153
82 <CLOSED TEXT LIST> ::= <CLOSED TEXT>                                154
                        / <CLOSED TEXT LIST> , <CLOSED TEXT>            82 154
83 <EVENT IDENTIFIER> ::= <IDENTIFIER>                                  81
84 <ADDRESS PART> ::= <EMPTY>                                           64
                    / = <ADDRESS>                                      155
85 <EVENT LIST> ::= <EVENT DESIGNATOR>                                 156
                  / <EVENT LIST> , <EVENT DESIGNATOR>                   85 156
86 <EVENT SEGMENT> ::= <EVENT ARRAY LIST> [ <BOUND LIST> ]             157 149
87 <FIELD PART> ::= <FIELD IDENTIFIER> = <FIELD>                       158 159
88 <FILE LIST PART> ::= <FILE DESIGNATOR>                              160
                      / <FILE DESIGNATOR> ( <INITIAL ATTRIBUTE LIST> ) 160 161
89 <INTERRUPT IDENTIFIER> ::= <IDENTIFIER>                              81
90 <LABEL IDENTIFIER> ::= <IDENTIFIER>                                  81
91 <LAYOUT PART> ::= <LAYOUT IDENTIFIER> ( <LAYOUT ITEM LIST> )         162 163
92 <MONITORED ITEM> ::= <SIMPLE VARIABLE>                              164
                      / <ARRAY IDENTIFIER>                             148
93 <PICTURE PART> ::= <PICTURE IDENTIFIER> ( <PICTURE> )                165 166
94 <FORMAL PARAMETER PART> ::= <EMPTY>                                  64
                             / ( <FORMAL PARAMETER LIST> ) ;           167
                               <VALUE PART> <SPECIFICATION PART>       168 169
95 <QUEUE ARRAY IDENTIFIER> ::= <IDENTIFIER>                            81
96 <INTERNAL NAME PART> ::= <EMPTY>                                     64
                          / = <SECOND NAME> <ADDRESS PART>              170  84
97 <ARITHMETIC EXPRESSION> ::= <CONDITIONAL ARITHMETIC EXPRESSION>      171
                             / <ARITHMETIC ASSIGNMENT>                  172
                             / <SIMPLE ARITHMETIC EXPRESSION>           173
                             / <WORD EXPRESSION>                        174
98 <ENTRY DESCRIPTION> ::= ( <ENTRY ITEM LIST> ) ; <VALUE PART> ;       175 168
                           <SPECIFICATION PART>                         169
99 <ALGORITHM PART> ::= <EMPTY>                                         64
                      / USING <ALGORITHM LIST>                          176
100 <QUEUE IDENTIFIER> ::= <IDENTIFIER>                                 81
101 <TYPE PART> ::= <TYPE IDENTIFIER> <ADDRESS PART>                    177  84
```

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

```
                / <TYPE IDENTIFIER> <REPLACEMENT OPERATOR>                    177 103
                  <INITIAL VALUE>                                            178
102 <VALUE ARRAY IDENTIFIER> ::= <IDENTIFIER>                                 81
103 <REPLACEMENT OPERATOR> ::= =
                             / := =
                             / =: =
                             / := =:
104 <CONSTANT LIST> ::= <CONSTANT>                                            179
                      / <CONSTANT LIST> , <CONSTANT>                          104 179
105 <BOOLEAN EXPRESSION> ::= <CONDITIONAL BOOLEAN EXPRESSION>                 180
                           / <BOOLEAN ASSIGNMENT>                             181
                           / <SIMPLE BOOLEAN EXPRESSION>                      182
                           / <WORD EXPRESSION>                                174
106 <ITERATION CLAUSE> ::= <WHILE PART> DO                                    183
                         / <FOR CLAUSE> DO                                    184
                         / <THRU CLAUSE> DO                                   185
107 <ARITHMETIC ASSIGNMENT STATEMENT> ::= <ARITHMETIC ASSIGNMENT>             172
                                        / <ARITHMETIC FIELD ASSIGNMENT>       186
108 <ARRAY ASSIGNMENT STATEMENT> ::= <ARRAY ASSIGNMENT>                       187
                                   / <ARRAY FIELD ASSIGNMENT>                 188
109 <BOOLEAN ASSIGNMENT STATEMENT> ::= <BOOLEAN ASSIGNMENT>                   181
                                     / <BOOLEAN FIELD ASSIGNMENT>             189
110 <QUEUE ASSIGNMENT> ::= <QUEUE DESIGNATOR> <REPLACEMENT OPERATOR>          190 103
                           <REFERENCE EXPRESSION>                             191
111 <REFERENCE ASSIGNMENT> ::= <REFERENCE VARIABLE> <REPLACEMENT OPERATOR>    192 103
                               <REFERENCE EXPRESSION>                         191
112 <WORD ASSIGNMENT> ::= <WORD VARIABLE> <REPLACEMENT OPERATOR>              193 103
                          <EXPRESSION>                                        194
113 <GO TO STATEMENT> ::= GO <TO PART> <DESIGNATIONAL EXPRESSION>             195 196
                        / GO <TO PART> <CASE HEAD> (                          195 197
                          <DESIGNATIONAL EXPRESSION LIST> ) GO <TO PART>      198 195
                          <WORD VARIABLE>                                     193
114 <DUMMY STATEMENT> ::= <EMPTY>                                             64
                        / <DUMMY STATEMENT> <LABEL> :                         114  63
115 <CASE STATEMENT> ::= <CASE HEAD> <COMPOUND STATEMENT>                     197   4
116 <SET STATEMENT> ::= SET ( <EVENT DESIGNATOR> )                            156
117 <RESET STATEMENT> ::= RESET ( <EVENT DESIGNATOR> )                        156
118 <CAUSE STATEMENT> ::= CAUSE ( <EVENT DESIGNATOR> )                        156
119 <CAUSEANDRESET STATEMENT> ::= CAUSENRESET ( <EVENT DESIGNATOR> )           156
120 <WAIT STATEMENT> ::= <WAITONEVENT STATEMENT>                              199
                       / <WAITONTIME STATEMENT>                               200
```

```
121 <WAITANDRESET STATEMENT> ::= <WAITANDRESET-SIMPLE STATEMENT>                 201
                              / <WAITANDRESET-COMPLEX STATEMENT>                 202
122 <DSWAIT STATEMENT> ::= <DSWAIT-SIMPLE STATEMENT>                            203
                         / <DSWAIT-COMPLEX STATEMENT>                           204
123 <DSWAITANDRESET STATEMENT> ::= <DSWAITANDRESET-SIMPLE STATEMENT>             205
                                 / <DSWAITANDRESET-COMPLEX STATEMENT>           206
124 <PROCURE STATEMENT> ::= PROCURE ( <EVENT DESIGNATOR> )                      156
125 <FIX STATEMENT> ::= FIX ( <EVENT DESIGNATOR> )                              156
126 <LIBERATE STATEMENT> ::= LIBERATE ( <EVENT DESIGNATOR> )                    156
127 <FREE STATEMENT> ::= FREE ( <EVENT DESIGNATOR> )                            156
128 <FORK PARAMETERS> ::= <STACK SIZE> , <PRIORITY> <VISIBLE NAME INDEX>        207 208 209
129 <FUNCTION IDENTIFIER> ::= <IDENTIFIER>                                       81
130 <ACTUAL PARAMETER PART> ::= <EMPTY>                                          64
                              / ( <ACTUAL PARAMETER LIST> )                     210
131 <CLOSE STATEMENT> ::= CLOSE ( <FILE DESIGNATOR> )                           160
                        / CLOSE ( <FILE DESIGNATOR> , <CLOSE OPTION> )          160 211
132 <LOCK STATEMENT> ::= LOCK ( <FILE DESIGNATOR> )                             160
                       / LOCK ( <FILE DESIGNATOR> , * )                         160
133 <READ STATEMENT> ::= READ ( <FILE PART> , <WORD COUNT> , <I/O AREA> )       212 213 214
                          <I/O FINISH EVENT>                                    215
134 <REWIND STATEMENT> ::= REWIND ( <FILE DESIGNATOR> )                         160
135 <SEEK STATEMENT> ::= SEEK ( <FILE DESIGNATOR> [ <RECORD NUMBER> ] )         160 216
136 <SPACE STATEMENT> ::= SPACE ( <FILE DESIGNATOR> ,                           160
                           <ARITHMETIC EXPRESSION> )                            97
137 <WRITE STATEMENT> ::= WRITE ( <FILE PART> , <WORD COUNT> , <I/O AREA>       212 213 214
                           ) <I/O FINISH EVENT>                                 215
138 <ATTACH STATEMENT> ::= ATTACH <INTERRUPT IDENTIFIER> TO                      89
                            <EVENT DESIGNATOR>                                  156
139 <DETACH STATEMENT> ::= DETACH <INTERRUPT IDENTIFIER>                        89
140 <ENABLE STATEMENT> ::= ENABLE <INTERRUPT IDENTIFIER>                        89
141 <DISABLE STATEMENT> ::= DISABLE <INTERRUPT IDENTIFIER>                      89
142 <FAULT NAME> ::= ANYFAULT
                   / BOTTOMOFSTACK
                   / EXPONENTOVERFLOW
                   / EXPONENTUNDERFLOW
                   / INACTIVEQUEUE
                   / INTEGEROVERFLOW
                   / INVALIDADDRESS
                   / INVALIDINDEX
                   / INVALIDOP
                   / INVALIDPROGRAMWORD
```

```
                    / LOOP
                    / MEMORYPARITY
                    / MEMORYPROTECT
                    / PROGRAMMEDOPERATOR
                    / SCANPARITY
                    / SEQUENCE
                    / STACKOVERFLOW
                    / STACKUNDERFLOW
                    / STRINGPROTECT
                    / ZERODIVIDE
143 <DESTINATION> ::= <UPDATE POINTER> <POINTER EXPRESSION>                 217 218
144 <SOURCE LIST> ::= <SOURCE PART>                                         219
                    / <SOURCE PART> , <SOURCE LIST>                         219 144
145 <SOURCE> ::= <POINTER SOURCE>                                           220
                / <ARITHMETIC SOURCE>                                       221
146 <SCAN PART> ::= <SCAN COUNT> <CONDITION>                                222 223
                  / <CONDITION>                                             223
147 <DO STATEMENT> ::= DO <STATEMENT> UNTIL <BOOLEAN EXPRESSION>             9 105
148 <ARRAY IDENTIFIER> ::= <IDENTIFIER>                                      81
                         / <ARRAY INTRINSIC>                                224
149 <BOUND LIST> ::= <BOUND>                                                225
                   / <BOUND LIST> , <BOUND>                                 149 225
150 <FORMAL SYMBOL PART> ::= <EMPTY>                                        64
                           / ( <FORMAL SYMBOL LIST> )                       226
151 <TEXT> ::=
                S ANY SEQUENCE OF VALID <CHARACTER>S NOT INCLUDING
                A FREE # Z
152 <LETTER> ::= A
                / B
                / C
                / D
                / E
                / F
                / G
                / H
                / I
                / J
                / K
                / L
                / M
                / N
```

```
                / O
                / P
                / Q
                / R
                / S
                / T
                / U
                / V
                / W
                / X
                / Y
                / Z
153 <DIGIT> ::= 0
                / 1
                / 2
                / 3
                / 4
                / 5
                / 6
                / 7
                / 8
                / 9
154 <CLOSED TEXT> ::=
                      ≤ AN ACTUAL TEXT NOT CONTAINING
                      UNMATCHED BRACKETING SYMBOLS OR UNBRACKETED COMMAS
                      ≥
155 <ADDRESS> ::= <IDENTIFIER>                                          81
                / <ADDRESS COUPLE>                                     227
                / <IDENTIFIER> <ADDING OPERATOR> <UNSIGNED INTEGER>     81 228 229
156 <EVENT DESIGNATOR> ::= <EVENT IDENTIFIER>                           83
                         / <EVENT ITEM>                                230
                         / <EVENT ARRAY IDENTIFIER> [ <SUBSCRIPT LIST> ] 231 232
                         / <EVENT ARRAY ITEM> [ <SUBSCRIPT LIST> ]      233 232
157 <EVENT ARRAY LIST> ::= <EVENT ARRAY IDENTIFIER> <ADDRESS PART>      231  84
                         / <EVENT ARRAY LIST> , <EVENT ARRAY IDENTIFIER> 157 231
                           <ADDRESS PART>                               84
158 <FIELD IDENTIFIER> ::= <IDENTIFIER>                                 81
159 <FIELD> ::= <ARITHMETIC EXPRESSION> : <ARITHMETIC EXPRESSION>       97  97
160 <FILE DESIGNATOR> ::= <FILE IDENTIFIER>                            234
                        / <DIRECT FILE IDENTIFIER>                     235
161 <INITIAL ATTRIBUTE LIST> ::= <INITIAL ATTRIBUTE>                   236
```

```
                              / <INITIAL ATTRIBUTE LIST> ,                    161
                                <INITIAL ATTRIBUTE>                           236
162 <LAYOUT IDENTIFIER> ::= <IDENTIFIER>                                       81
163 <LAYOUT ITEM LIST> ::= <LAYOUT ITEM>                                      237
                         / <LAYOUT ITEM LIST> , <LAYOUT ITEM>                 163 237
164 <SIMPLE VARIABLE> ::= <IDENTIFIER>                                         81
165 <PICTURE IDENTIFIER> ::= <IDENTIFIER>                                      81
166 <PICTURE> ::= <PICTURE SYMBOL>                                            238
                / <PICTURE> <PICTURE SYMBOL>                                  166 238
167 <FORMAL PARAMETER LIST> ::= <FORMAL PARAMETER>                             239
                              / <FORMAL PARAMETER LIST>                       167
                                <PARAMETER DELIMITER> <FORMAL PARAMETER>      240 239
168 <VALUE PART> ::= <EMPTY>                                                   64
                   / VALUE <IDENTIFIER LIST> ;                                241
169 <SPECIFICATION PART> ::= <SPECIFICATION>                                   242
                           / <SPECIFICATION PART> ; <SPECIFICATION>            169 242
170 <SECOND NAME> ::= <IDENTIFIER>                                             81
171 <CONDITIONAL ARITHMETIC EXPRESSION> ::= <IF CLAUSE>                         61
                                           <ARITHMETIC EXPRESSION> ELSE        97
                                           <ARITHMETIC EXPRESSION>             97
172 <ARITHMETIC ASSIGNMENT> ::= <ARITHMETIC VARIABLE>                         243
                                <REPLACEMENT OPERATOR>                        103
                                <ARITHMETIC EXPRESSION>                        97
173 <SIMPLE ARITHMETIC EXPRESSION> ::= <SIMPLE ARITHMETIC EXPRESSION>         173
                                         <ADDING OPERATOR> <TERM>             228 244
                                       / <UNARY OPERATOR> <TERM>              245 244
                                       / <TERM>                               244
174 <WORD EXPRESSION> ::= <WORD ASSIGNMENT>                                   112
                        / <WORD VARIABLE>                                     193
                        / <WORD ITEM>                                         246
                        / <IF CLAUSE> <WORD EXPRESSION> ELSE                   61 174
                          <WORD EXPRESSION>                                   174
                        / WORD ( <MOST EXPRESSIONS> )                         247
                        / <FUNCTION DESIGNATOR>                                69
                        / <CASE HEAD> ( <WORD EXPRESSION LIST> )              197 248
                        / ( <WORD EXPRESSION> )                               174
175 <ENTRY ITEM LIST> ::= <ITEM LIST> <INVISIBLE ITEM LIST>                    249 250
176 <ALGORITHM LIST> ::= <ALGORITHM>                                          251
                       / <ALGORITHM> ; <ALGORITHM LIST>                       251 176
177 <TYPE IDENTIFIER> ::= <IDENTIFIER>                                         81
178 <INITIAL VALUE> ::= <EXPRESSION>                                          194
```

```
179 <CONSTANT> ::= <NUMBER>                                                  252
                 / <LOGICAL VALUE>                                           253
                 / <STRING>                                                  254
                 / <CONSTANT EXPRESSION>                                     255
                 / <UNSIGNED INTEGER> ( <CONSTANT LIST> )                    229 104
180 <CONDITIONAL BOOLEAN EXPRESSION> ::= <IF CLAUSE> <BOOLEAN EXPRESSION>     61 105
                                         ELSE <BOOLEAN EXPRESSION>           105
181 <BOOLEAN ASSIGNMENT> ::= <BOOLEAN VARIABLE> <REPLACEMENT OPERATOR>        256 103
                             <BOOLEAN EXPRESSION>                            105
182 <SIMPLE BOOLEAN EXPRESSION> ::= <IMPLICATION>                            257
                                  / <SIMPLE BOOLEAN EXPRESSION> EQV          182
                                    <IMPLICATION>                            257
183 <WHILE PART> ::= WHILE <BOOLEAN EXPRESSION>                              105
184 <FOR CLAUSE> ::= FOR <CONTROLLED VARIABLE> <REPLACEMENT OPERATOR>         258 103
                     <FOR PART>                                              259
185 <THRU CLAUSE> ::= THRU <ARITHMETIC EXPRESSION>                           97
186 <ARITHMETIC FIELD ASSIGNMENT> ::= <ARITHMETIC VARIABLE> .                 243
                                      <FIELD IDENTIFIER>                     158
                                      <REPLACEMENT OPERATOR>                 103
                                      <ARITHMETIC EXPRESSION>                97
187 <ARRAY ASSIGNMENT> ::= <ARRAY DESIGNATOR> <REPLACEMENT OPERATOR>          260 103
                           <ARRAY EXPRESSION>                                261
188 <ARRAY FIELD ASSIGNMENT> ::= <ARRAY DESIGNATOR> . <FIELD IDENTIFIER>     260 158
                                 <REPLACEMENT OPERATOR>                      103
                                 <ARITHMETIC EXPRESSION>                     97
189 <BOOLEAN FIELD ASSIGNMENT> ::= <BOOLEAN VARIABLE> . <FIELD IDENTIFIER>   256 158
                                   <REPLACEMENT OPERATOR>                    103
                                   <BOOLEAN EXPRESSION>                      105
190 <QUEUE DESIGNATOR> ::= <QUEUE IDENTIFIER>                                100
                         / <QUEUE ARRAY IDENTIFIER> [                        95
                           <ARITHMETIC EXPRESSION> ]                         97
191 <REFERENCE EXPRESSION> ::= NULL
                             / <REFERENCE ASSIGNMENT>                        111
                             / <IF CLAUSE> <REFERENCE EXPRESSION> ELSE       61 191
                               <REFERENCE EXPRESSION>                        191
                             / <REFERENCE VARIABLE>                          192
                             / <ENTRY EXPRESSION>                            262
                             / <QUEUE DESIGNATOR>                            190
                             / REFERENCE ( <VARIABLE> )                      263
                             / <FUNCTION DESIGNATOR>                         69
                             / <WORD EXPRESSION>                             174
```

```
                        / ( <REFERENCE EXPRESSION> )                          191
                        / <REFERENCE ITEM>                                    264
                        / <CASE HEAD> ( <REFERENCE EXPRESSION LIST>           197 265
                          )
192 <REFERENCE VARIABLE> ::= <VARIABLE>                                       263
193 <WORD VARIABLE> ::= <VARIABLE>                                            263
194 <EXPRESSION> ::= <GENERAL COMPONENT>                                      266
                   / <ARITHMETIC EXPRESSION>                                   97
                   / <ARRAY EXPRESSION>                                       261
                   / <BOOLEAN EXPRESSION>                                     105
                   / <CASE EXPRESSION>                                        267
                   / <CONDITIONAL EXPRESSION>                                 268
                   / <DESIGNATIONAL EXPRESSION>                               196
                   / <POINTER EXPRESSION>                                     218
                   / <REFERENCE EXPRESSION>                                   191
                   / <WORD EXPRESSION>                                        174
195 <TO PART> ::= <EMPTY>                                                      64
                / TO
196 <DESIGNATIONAL EXPRESSION> ::= <LABEL DESIGNATOR>                         269
                                 / <CASE HEAD> (                              197
                                   <DESIGNATIONAL EXPRESSION LIST> )          198
                                 /
                                   <CONDITIONAL DESIGNATIONAL EXPRESSION>     270
197 <CASE HEAD> ::= CASE <ARITHMETIC EXPRESSION> OF                            97
198 <DESIGNATIONAL EXPRESSION LIST> ::= <DESIGNATIONAL EXPRESSION>             196
                                      / <DESIGNATIONAL EXPRESSION LIST>        198
                                        , <DESIGNATIONAL EXPRESSION>           196
199 <WAITONEVENT STATEMENT> ::= <WAITONEVENT-SIMPLE STATEMENT>                 271
                              / <WAITONEVENT-COMPLEX STATEMENT>                272
200 <WAITONTIME STATEMENT> ::= WAIT ( ( <TIME> ) )                            273
201 <WAITANDRESET-SIMPLE STATEMENT> ::= WAITANDRESET ( <EVENT DESIGNATOR>      156
                                        )
202 <WAITANDRESET-COMPLEX STATEMENT> ::= WAITANDRESET (
                                          <WAIT PARAMETER LIST> )             274
203 <DSWAIT-SIMPLE STATEMENT> ::= DSWAIT ( <EVENT DESIGNATOR> )                156
204 <DSWAIT-COMPLEX STATEMENT> ::= DSWAIT ( <WAIT PARAMETER LIST> )            274
205 <DSWAITANDRESET-SIMPLE STATEMENT> ::= DSWAITANDRESET (
                                          <EVENT DESIGNATOR> )                156
206 <DSWAITANDRESET-COMPLEX STATEMENT> ::= DSWAITANDRESET (
                                          <WAIT PARAMETER LIST> )              274
207 <STACK SIZE> ::= <ARITHMETIC EXPRESSION>                                    97
```

```
208 <PRIORITY> ::= <ARITHMETIC EXPRESSION>                                    97
209 <VISIBLE NAME INDEX> ::= <EMPTY>                                          64
                           / , <ARITHMETIC EXPRESSION>                       97
210 <ACTUAL PARAMETER LIST> ::= <ACTUAL PARAMETER>                            275
                              / <ACTUAL PARAMETER LIST>                      210
                                <PARAMETER DELIMITER> <ACTUAL PARAMETER>     240 275
211 <CLOSE OPTION> ::= *
                     / PURGE
                     / REEL
212 <FILE PART> ::= <FILE DESIGNATOR>                                        160
                  / <FILE DESIGNATOR>                                        160
                    <RECORD NUMBER OR CARRIAGE CONTROL>                      276
213 <WORD COUNT> ::= <ARITHMETIC EXPRESSION>                                  97
214 <I/O AREA> ::= <ARRAY ROW>                                               277
                 / <POINTER EXPRESSION>                                      218
                 / <SUBSCRIPTED VARIABLE>                                    278
215 <I/O FINISH EVENT> ::= <EMPTY>                                            64
                         / [ <EVENT DESIGNATOR> ]                            156
216 <RECORD NUMBER> ::= <ARITHMETIC EXPRESSION>                               97
217 <UPDATE POINTER> ::= <EMPTY>                                              64
                       / <POINTER IDENTIFIER> :                              279
218 <POINTER EXPRESSION> ::= <CONDITIONAL POINTER EXPRESSION>                 280
                           / <SIMPLE POINTER EXPRESSION>                     281
219 <SOURCE PART> ::= <SOURCE> <TRANSFER PART>                                145 282
                    / <STRING> <TRANSFER PART>                                254 282
                    / <ARITHMETIC SOURCE> <ARITHMETIC TRANSFER PART>          221 283
                    / <STRING>                                                254
220 <POINTER SOURCE> ::= <UPDATE POINTER> <POINTER EXPRESSION>                217 218
221 <ARITHMETIC SOURCE> ::= <UPDATE VARIABLE> <ARITHMETIC EXPRESSION>         284  97
222 <SCAN COUNT> ::= FOR <UPDATE COUNT> <ARITHMETIC EXPRESSION>               285  97
223 <CONDITION> ::= WHILE <RELATIONAL OPERATOR> <ARITHMETIC EXPRESSION>       286  97
                  / UNTIL <RELATIONAL OPERATOR> <ARITHMETIC EXPRESSION>       286  97
                  / UNTIL IN <TABLE>                                          287
                  / WHILE IN <TABLE>                                          287
224 <ARRAY INTRINSIC> ::= M
                        / MEMORY
                        / REGISTERS
                        / STACK
                        / STACKVECTOR
                        / WORDSTACK
225 <BOUND> ::= <WORD COUNT>                                                  213
```

```
                / *
226 <FORMAL SYMBOL LIST> ::= <FORMAL SYMBOL>                                    288
                           / <FORMAL SYMBOL LIST> , <FORMAL SYMBOL>             226 288
227 <ADDRESS COUPLE> ::= ( <LEVEL> <DISPLACEMENT> )                             289 290
228 <ADDING OPERATOR> ::= +
                        / -
229 <UNSIGNED INTEGER> ::= <DIGIT>                                              153
                         / <UNSIGNED INTEGER> <DIGIT>                           229 153
230 <EVENT ITEM> ::= <ITEM>                                                     291
231 <EVENT ARRAY IDENTIFIER> ::= <IDENTIFIER>                                    81
232 <SUBSCRIPT LIST> ::= <SUBSCRIPT>                                            292
                       / <SUBSCRIPT LIST> , <SUBSCRIPT>                         232 292
233 <EVENT ARRAY ITEM> ::= <ITEM>                                               291
234 <FILE IDENTIFIER> ::= <IDENTIFIER>                                          81
235 <DIRECT FILE IDENTIFIER> ::= <IDENTIFIER>                                   81
236 <INITIAL ATTRIBUTE> ::= <FILE ATTRIBUTE NAME> = <FILE ATTRIBUTE VALUE>      293 294
237 <LAYOUT ITEM> ::= <LAYOUT FIELD> <FIELD VALUE PART>                          295 296
238 <PICTURE SYMBOL> ::= "<HEXADECIMAL STRING>"
                       / "<BCL STRING>"
                       / "<ASCII STRING>"
                       / "<EBCDIC STRING>"
                       / <PICTURE CHARACTER> <REPEAT PART>                      297 298
                       / <CONTROL CHARACTER>                                    299
                       / <INTRODUCTION>                                         300
                       / <SKIP CHARACTER> <REPEAT PART>                         301 298
                       / <SINGLE PICTURE CHARACTER>                             302
239 <FORMAL PARAMETER> ::= <IDENTIFIER>                                         81
240 <PARAMETER DELIMITER> ::= ,
                            / )
                              "≤ANY SEQUENCE OF LETTERS AND SINGLE SPACES≥"
                              (
241 <IDENTIFIER LIST> ::= <IDENTIFIER>                                          81
                        / <IDENTIFIER LIST> , <IDENTIFIER>                      241  81
242 <SPECIFICATION> ::= <SPECIFIER> <IDENTIFIER LIST>                           303 241
                      / <ARRAY SPECIFICATION>                                   304
243 <ARITHMETIC VARIABLE> ::= <VARIABLE>                                        263
244 <TERM> ::= <FACTOR>                                                         305
             / <TERM> <MULTIPLYING OPERATOR> <FACTOR>                           244 306 305
245 <UNARY OPERATOR> ::= <ADDING OPERATOR>                                      228
246 <WORD ITEM> ::= <ITEM>                                                      291
247 <MOST EXPRESSIONS> ::= <ARITHMETIC EXPRESSION>                               97
```

```
                        / <BOOLEAN EXPRESSION>                                  105
                        / <REFERENCE EXPRESSION>                                191
                        / <ARRAY EXPRESSION>                                    261
248 <WORD EXPRESSION LIST> ::= <WORD EXPRESSION>                                174
                             / <WORD EXPRESSION LIST> ,                         248
                               <WORD EXPRESSION>                                174
249 <ITEM LIST> ::= <ITEM IDENTIFIER>                                           307
                  / <ITEM LIST> <PARAMETER DELIMITER> <ITEM IDENTIFIER>         249 240 807
250 <INVISIBLE ITEM LIST> ::= <ITEM LIST>                                       249
251 <ALGORITHM> ::= <BOOLEAN ALGORITHM IDENTIFIER> IF <BOOLEAN EXPRESSION>      308 105
                  / <REFERENCE ALGORITHM IDENTIFIER> IS                         309
                    <REFERENCE EXPRESSION>                                      191
                  / TO <ALGORITHM IDENTIFIER> , <STATEMENT>                     310   9
                  / <LOCK SPECIFICATION>                                        311
                  / <INTEGER ALGORITHM IDENTIFIER> , <STATEMENT>                312   9
252 <NUMBER> ::= <SIGN> <UNSIGNED NUMBER>                                       313 314
253 <LOGICAL VALUE> ::= TRUE
                      / FALSE
254 <STRING> ::= <SIMPLE STRING>                                                315
               / <STRING> <SIMPLE STRING>                                       254 315
255 <CONSTANT EXPRESSION> ::=
                            SAN EXPRESSION WHICH CAN BE ENTIRELY
                            EVALUATED BY THE ESPOL COMPILER AT COMPILE
                            TIME ≥
256 <BOOLEAN VARIABLE> ::= <VARIABLE>                                           263
257 <IMPLICATION> ::= <BOOLEAN TERM>                                            316
                    / <IMPLICATION> IMP <BOOLEAN TERM>                          257 316
258 <CONTROLLED VARIABLE> ::= <SIMPLE VARIABLE>                                 164
                            / <SUBSCRIPTED VARIABLE>                            278
259 <FOR PART> ::= <INITIAL PART> <STEP PART> <FINAL PART>                      317 318 319
260 <ARRAY DESIGNATOR> ::= <ARRAY IDENTIFIER> <SUBARRAY DESIGNATOR>             148 320
                         / <ARRAY VARIABLE>                                     321
261 <ARRAY EXPRESSION> ::= <ARRAY ASSIGNMENT>                                   187
                         / <ARRAY PRIMARY>                                      322
                         / <IF CLAUSE> <ARRAY EXPRESSION> ELSE                   61 261
                           <ARRAY EXPRESSION>                                   261
262 <ENTRY EXPRESSION> ::= <QUEUE NAME> ( <ACTUAL ITEM LIST> )                  323 324
263 <VARIABLE> ::= <SIMPLE VARIABLE>                                            164
                 / <SUBSCRIPTED VARIABLE>                                       278
264 <REFERENCE ITEM> ::= <ITEM>                                                 291
265 <REFERENCE EXPRESSION LIST> ::= <REFERENCE EXPRESSION>                      191
```

```
                                / <REFERENCE EXPRESSION LIST> ,          265
                                  <REFERENCE EXPRESSION>                 191
266 <GENERAL COMPONENT> ::= <BASIC COMPONENT>                            325
                          / <VARIABLE>                                   263
                          / <ITEM DESIGNATOR>                            326
                          / <VALUE DESIGNATOR>                           327
                          / <EVENT DESIGNATOR>                           156
                          / <INTRINSIC>                                  328
267 <CASE EXPRESSION> ::= <CASE HEAD> ( <EXPRESSION LIST> )              197 329
268 <CONDITIONAL EXPRESSION> ::= <CONDITIONAL ARITHMETIC EXPRESSION>      171
                               / <CONDITIONAL BOOLEAN EXPRESSION>        180
                               / <CONDITIONAL DESIGNATIONAL EXPRESSION>  270
                               / <CONDITIONAL POINTER EXPRESSION>        280
269 <LABEL DESIGNATOR> ::= <LABEL IDENTIFIER>                             90
270 <CONDITIONAL DESIGNATIONAL EXPRESSION> ::= <IF CLAUSE>                61
                                               <DESIGNATIONAL EXPRESSION> 196
                                               ELSE
                                               <DESIGNATIONAL EXPRESSION> 196
271 <WAITUNEVENT-SIMPLE STATEMENT> ::= WAIT ( <EVENT DESIGNATOR> )       156
272 <WAITUNEVENT-COMPLEX STATEMENT> ::= WAIT ( <WAIT PARAMETER LIST> )    274
273 <TIME> ::=
            $THE MINIMUM AMOUNT OF TIME IN SECONDS
            ( FRACTIONAL SECONDS ALLOWED ) THAT A TASK IS TO BE
            SUSPENDED $
274 <WAIT PARAMETER LIST> ::= <EVENT LIST>                                85
                            / ( <TIME> ) , <EVENT LIST>                  273 85
275 <ACTUAL PARAMETER> ::= <EXPRESSION>                                  194
                         / <PROCEDURE IDENTIFIER>                        44
                         / <EVENT DESIGNATOR>                           156
276 <RECORD NUMBER OR CARRIAGE CONTROL> ::= [ <ARITHMETIC EXPRESSION> ]   97
                                          / [ LINE
                                            <ARITHMETIC EXPRESSION> ]    97
                                          / [ SPACE
                                            <ARITHMETIC EXPRESSION> ]    97
                                          / [ SKIP
                                            <ARITHMETIC EXPRESSION> ]    97
                                          / [ STACKER
                                            <ARITHMETIC EXPRESSION> ]    97
                                          / [ NO ]
277 <ARRAY ROW> ::= <ARRAY IDENTIFIER> [ <ROW DESIGNATOR> ]              148 330
278 <SUBSCRIPTED VARIABLE> ::= <ARRAY IDENTIFIER> [ <SUBSCRIPT LIST> ]   148 232
```

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

```
279 <POINTER IDENTIFIER> ::= <IDENTIFIER>                                    81
280 <CONDITIONAL POINTER EXPRESSION> ::= <IF CLAUSE> <POINTER EXPRESSION>    61 218
                                        ELSE <POINTER EXPRESSION>           218
281 <SIMPLE POINTER EXPRESSION> ::= <POINTER PRIMARY> <SKIP>                 331 332
                                  / <POINTER ASSIGNMENT>                    333
                                  / <WORD ARRAY ROW>                        334
                                  / <SUBSCRIPTED WORD VARIABLE>             335
282 <TRANSFER PART> ::= <SCAN COUNT> <CONDITION>                            222 223
                      / <CONDITION> <FINAL COUNT> <UNITS>                   223 336 337
                      / WITH <PICTURE DESIGNATOR>                           338
                      / <FINAL COUNT> WITH <TRANSLATE TABLE>                336 339
283 <ARITHMETIC TRANSFER PART> ::= <DIGIT COUNT>                            340
                                 / <CORRECT COUNT>                          341
                                 / <CORRECT COUNT> <CONDITION>              341 223
284 <UPDATE VARIABLE> ::= <EMPTY>                                           64
                        / <SIMPLE VARIABLE> !                               164
285 <UPDATE COUNT> ::= <EMPTY>                                              64
                     / <SIMPLE VARIABLE> !                                  164
286 <RELATIONAL OPERATOR> ::= >
                            / ≤
                            / =
                            / ≥
                            / <
                            / ≠
                            / IS
287 <TABLE> ::= <ARRAY ROW>                                                 277
              / <SUBSCRIPTED VARIABLE>                                      278
288 <FORMAL SYMBOL> ::= <IDENTIFIER>                                        81
289 <LEVEL> ::= <UNSIGNED INTEGER>                                          229
              / " <UNSIGNED INTEGER>                                        229
290 <DISPLACEMENT> ::= <EMPTY>                                              64
                     / , <UNSIGNED INTEGER>                                 229
291 <ITEM> ::= <ITEM IDENTIFIER> @ <REFERENCE EXPRESSION>                   307 191
292 <SUBSCRIPT> ::= <ARITHMETIC EXPRESSION>                                 97
293 <FILE ATTRIBUTE NAME> ::= <ARITHMETIC-VALUED FILE ATTRIBUTE NAME>        342
                            / <BOOLEAN-VALUED FILE ATTRIBUTE NAME>          343
                            / <MNEMONIC-VALUED FILE ATTRIBUTE NAME>         344
                            / <POINTER-VALUED FILE ATTRIBUTE NAME>          345
294 <FILE ATTRIBUTE VALUE> ::= <CONSTANT>                                   179
                             / <FILE ATTRIBUTE MNEMONIC VALUE>              346
295 <LAYOUT FIELD> ::= <FIELD PART>                                         87
```

```
                    / <FIELD>                                                    159
                    / <FIELD IDENTIFIER>                                         158
296 <FIELD VALUE PART> ::= <EMPTY>                                               64
                         / <REPLACEMENT OPERATOR> <UNSIGNED INTEGER>             103 229
297 <PICTURE CHARACTER> ::= A
                          / D
                          / E
                          / F
                          / I
                          / Q
                          / R
                          / X
                          / Z
                          / 9
298 <REPEAT PART> ::= <EMPTY>                                                    64
                    / ( <UNSIGNED INTEGER> )                                     229
                    / ( * )
299 <CONTROL CHARACTER> ::= 4
                          / 6
                          / 7
                          / 8
                          / 9
300 <INTRODUCTION> ::= <INTRODUCTION CODE> <NEW CHARACTER>                       347 348
301 <SKIP CHARACTER> ::= >
                       / <
302 <SINGLE PICTURE CHARACTER> ::= J
                                 / S
303 <SPECIFIER> ::= <TYPE>                                                       56
                  / <PROCEDURE TYPE> PROCEDURE                                   49
                  / QUEUE
                  / EVENT
                  / PICTURE
304 <ARRAY SPECIFICATION> ::= <DIRECT SPECIFIER> <ARRAY TYPE> ARRAY              39 349
                              <ARRAY SPECIFIER LIST>                             350
305 <FACTOR> ::= <PRIMARY>                                                       351
               / <PRIMARY> * <INTEGER>                                           351 352
306 <MULTIPLYING OPERATOR> ::= *
                             / <SLASH>                                           353
                             / DIV
                             / MOD
                             / MUX
```

APPENDIX C. SYNTAX OF THE ESPOL LANGUAGE

```
307 <ITEM IDENTIFIER> ::= <IDENTIFIER>                                    81
                        / <ITEM>                                        291
308 <BOOLEAN ALGORITHM IDENTIFIER> ::= EMPTY
                                     / FULL
309 <REFERENCE ALGORITHM IDENTIFIER> ::= ALLOCATE
                                       / NEXT
                                       / LAST
                                       / FIRST
                                       / PRIOR
310 <ALGORITHM IDENTIFIER> ::= INSERT
                             / REMOVE
                             / DELINK
                             / <IDENTIFIER>                              81
311 <LOCK SPECIFICATION> ::= LOCKED
                           / LOCKED <QUEUE NAME>                        323
312 <INTEGER ALGORITHM IDENTIFIER> ::= POPULATION
313 <SIGN> ::= <EMPTY>                                                   64
             / +
             / -
314 <UNSIGNED NUMBER> ::= <DECIMAL NUMBER> <EXPONENT PART>          354 355
                        / <DECIMAL NUMBER>                           354
                        / <OCTAL NUMBER>                             356
315 <SIMPLE STRING> ::= <NUMERIC STRING>                             357
                      / <ALPHA STRING>                               358
316 <BOOLEAN TERM> ::= <BOOLEAN FACTOR>                              359
                     / <BOOLEAN TERM> OR <BOOLEAN FACTOR>            316 359
317 <INITIAL PART> ::= <ARITHMETIC EXPRESSION>                        97
318 <STEP PART> ::= STEP <ARITHMETIC EXPRESSION>                      97
                  / BY <ARITHMETIC EXPRESSION>                        97
319 <FINAL PART> ::= UNTIL <ARITHMETIC EXPRESSION>                    97
                   / <WHILE PART>                                    183
320 <SUBARRAY DESIGNATOR> ::= <EMPTY>                                 64
                            / [ <SUBSCRIPT PART> <SUBARRAY PART> ]   360 361
321 <ARRAY VARIABLE> ::= <SIMPLE VARIABLE>                           164
                       / <ARRAY ITEM>                                362
322 <ARRAY PRIMARY> ::= <ARRAY DESIGNATOR>                           260
                      / ( <ARRAY EXPRESSION> )                       261
                      / <ARRAY PRIMARY> & <LAYOUT>                   322 363
                      / <WORD EXPRESSION>                            174
323 <QUEUE NAME> ::= <QUEUE IDENTIFIER>                              100
                   / <QUEUE ARRAY IDENTIFIER>                         95
```

```
324 <ACTUAL ITEM LIST> ::= <ACTUAL PARAMETER LIST>                          210
325 <BASIC COMPONENT> ::= <BASIC SYMBOL>                                   364
                        / <IDENTIFIER>                                     81
                        / <NUMBER>                                        252
                        / <STRING> , 2 , IDENTIFIERS                      254
326 <ITEM DESIGNATOR> ::= <ARITHMETIC ITEM>                                365
                        / <WORD ITEM>                                     246
327 <VALUE DESIGNATOR> ::= <VALUE ARRAY IDENTIFIER> [ <SUBSCRIPT LIST> ]   102 232
328 <INTRINSIC> ::= <ARRAY INTRINSIC>                                      224
                  / <FUNCTION INTRINSIC>                                   366
                  / <PROCEDURE INTRINSIC>                                  367
                  / <WORD INTRINSIC>                                       368
329 <EXPRESSION LIST> ::= <ARITHMETIC EXPRESSION LIST>                     369
                        / <BOOLEAN EXPRESSION LIST>                        370
                        / <DESIGNATIONAL EXPRESSION LIST>                  198
                        / <POINTER EXPRESSION LIST>                        371
330 <ROW DESIGNATOR> ::= *
                       / <ROW> , *                                         372
331 <POINTER PRIMARY> ::= <POINTER IDENTIFIER>                             279
                        / ( <POINTER EXPRESSION> )                         218
                        / <CASE HEAD> ( <POINTER EXPRESSION LIST> )        197 371
                        / <POINTER DESIGNATOR>                             373
332 <SKIP> ::= <EMPTY>                                                      64
             / <ADDING OPERATOR> <PRIMARY>                                 228 351
333 <POINTER ASSIGNMENT> ::= <POINTER VARIABLE> <REPLACEMENT OPERATOR>     374 103
                             <POINTER EXPRESSION>                          218
334 <WORD ARRAY ROW> ::= <ARRAY ROW>                                       277
335 <SUBSCRIPTED WORD VARIABLE> ::= <SUBSCRIPTED VARIABLE>                 278
336 <FINAL COUNT> ::= FOR <ARITHMETIC EXPRESSION>                           97
337 <UNITS> ::= <EMPTY>                                                     64
              / WORDS
              / OVERWRITE
338 <PICTURE DESIGNATOR> ::= <PICTURE IDENTIFIER> <REPEAT PARAMETERS>       165 375
339 <TRANSLATE TABLE> ::= <TABLE>                                          287
340 <DIGIT COUNT> ::= FOR <ARITHMETIC EXPRESSION> DIGITS                     97
341 <CORRECT COUNT> ::= <SCAN COUNT> CORRECTLY                             222
342 <ARITHMETIC-VALUED FILE ATTRIBUTE NAME> ::= AREACLASS
                                              / AREAS
                                              / AREASIZE
                                              / ASSIGNTIME
                                              / ATTVALUE
```

```
                / ATTYPE
                / BLOCK
                / BLOCKSIZE
                / BUFFERS
                / CENSUS
                / COPIES
                / CYCLE
                / DATE
                / DIRECTION
                / DISPOSITION
                / ENABLEINPUT
                / ERRORTYPE
                / FAMILYSIZE
                / FILETYPE
                / LASTRECORD
                / LASTSTATION
                / LINENUM
                / MAXGENNO
                / MAXRECSIZE
                / MINRECSIZE
                / NEXTRECORD
                / PAGE
                / PAGESIZE
                / POPULATION
                / RECEPTIONS
                / RECORD
                / RECORDSZ
                / REEL
                / ROWADDRESS
                / ROWINUSE
                / SAVEFACTOR
                / SECURITYTYPE
                / SECURITYUSE
                / SCREEN
                / SERIALNO
                / SIZEMODE
                / SIZEOFFSET
                / SIZE2
                / SPEED
                / STATE
                / TAPEREELRECORD
```

```
                                        / TRANSMISSIONO
                                        / TRANSMISSIONS
                                        / UNITNO
                                        / UNITS
                                        / UNITSLEFT
                                        / USEDATE
                                        / VERSION
                                        / WIDTH
343 <BOOLEAN-VALUED FILE ATTRIBUTE NAME> ::= ATTERR
                                        / BEGINAT1
                                        / BREAK
                                        / CODEFILE
                                        / COPIES
                                        / CYLINDERMODE
                                        / DUPLICATED
                                        / EOF
                                        / INTERCHANGE
                                        / NULINPUT
                                        / OPEN
                                        / OPTIONAL
                                        / PRESENT
                                        / READCHECK
                                        / RESIDENT
                                        / SINGLEPACK
                                        / UPDATE
344 <MNEMONIC-VALUED FILE ATTRIBUTE NAME> ::= DENSITY
                                        / EXTMODE
                                        / FILEKIND
                                        / INTMODE
                                        / KIND
                                        / LABELTYPE
                                        / MYUSE
                                        / OTHERUSE
                                        / PARITY
                                        / PROTECTION
                                        / SECURITYTYPE
                                        / SECURITYUSE
                                        / SIZEMODE
                                        / SPEED
                                        / UNITS
345 <POINTER-VALUED FILE ATTRIBUTE NAME> ::= FAMILY
```

```
                                        / FORMESSAGE
                                        / INTNAME
                                        / TITLE
346 <FILE ATTRIBUTE MNEMONIC VALUE> ::= <DENSITY MNEMONIC>                  376
                                      / <ERRORTYPE MNEMONIC>                377
                                      / <EXTMODE MNEMONIC>                  378
                                      / <FILEKIND MNEMONIC>                 379
                                      / <INTMODE MNEMONIC>                  380
                                      / <KIND MNEMONIC>                     381
                                      / <LABELTYPE MNEMONIC>                382
                                      / <MYUSE MNEMONIC>                    383
                                      / <OTHERUSE MNEMONIC>                 384
                                      / <PARITY MNEMONIC>                   385
                                      / <PROTECTION MNEMONIC>               386
                                      / <SECURITYTYPE MNEMONIC>             387
                                      / <SECURITYUSE MNEMONIC>              388
                                      / <SIZEMODE MNEMONIC>                 389
                                      / <SPEED MNEMONIC>                    390
                                      / <STATE MNEMONIC>                    391
                                      / <UNITS MNEMONIC>                    392
347 <INTRODUCTION CODE> ::= B
                          / C
                          / M
                          / N
                          / P
                          / U
348 <NEW CHARACTER> ::= <STRING CHARACTER>                                  393
349 <ARRAY TYPE> ::= <EMPTY>                                                 64
                   / <TYPE>                                                  56
                   / EVENT
350 <ARRAY SPECIFIER LIST> ::= <ARRAY SPECIFIER>                            394
                             / <ARRAY SPECIFIER LIST> ,                     350
                               <ARRAY SPECIFIER>                            394
351 <PRIMARY> ::= <UNSIGNED NUMBER>                                         314
                / <STRING>                                                  254
                / <FIELD DESIGNATOR>                                        395
                / <OPERAND>                                                 396
                / <PRIMARY> & <ARITHMETIC EXPRESSION> <CONCATENATION>       351  97 397
                / <PRIMARY> & <LAYOUT>                                      351 363
352 <INTEGER> ::= <SIGN> <UNSIGNED INTEGER>                                 313 229
353 <SLASH> ::= /
```

```
354 <DECIMAL NUMBER> ::= <UNSIGNED INTEGER> <DECIMAL FRACTION>                229 398
                       / <UNSIGNED INTEGER>                                  229
                       / <DECIMAL FRACTION>                                  398
                       / <UNSIGNED INTEGER> ,                                229
355 <EXPONENT PART> ::= @ <INTEGER>                                          352
                      / @ @ <INTEGER>                                        352
356 <OCTAL NUMBER> ::= @ <OCTAL CONSTANT>                                    399
357 <NUMERIC STRING> ::= <BINARY CODE> <QUOTE> <BINARY STRING> <QUOTE>       400 464 401 464
                       / <QUATERNARY CODE> <QUOTE> <QUATERNARY STRING>       402 464 403
                         <QUOTE>                                             464
                       / <OCTAL CODE> <QUOTE> <OCTAL STRING> <QUOTE>         404 464 405 464
                       / <HEXADECIMAL CODE> <QUOTE> <HEXADECIMAL STRING>     406 464 407
                         <QUOTE>                                             464
358 <ALPHA STRING> ::= <BCL CODE> <QUOTE> <BCL STRING> <QUOTE>               408 464 409 464
                     / <ASCII CODE> <QUOTE> <ASCII STRING> <QUOTE>           410 464 411 464
                     / <EBCDIC CODE> <QUOTE> <EBCDIC STRING> <QUOTE>         412 464 413 464
359 <BOOLEAN FACTOR> ::= <BOOLEAN SECONDARY>                                 414
                       / <BOOLEAN FACTOR> AND <BOOLEAN SECONDARY>            359 414
360 <SUBSCRIPT PART> ::= <EMPTY>                                              64
                       / <SUBSCRIPT LIST> ,                                  232
361 <SUBARRAY PART> ::= *
                      / <SUBARRAY PART> , *                                  361
362 <ARRAY ITEM> ::= <ITEM>                                                  291
363 <LAYOUT> ::= <LAYOUT IDENTIFIER> ( <FIELD VALUE LIST> )                  162 415
364 <BASIC SYMBOL> ::= <EMPTY>                                                64
                     / <CHARACTER>                                           416
                     / <DELIMITER>                                           417
                     / <LOGICAL VALUE>                                       253
365 <ARITHMETIC ITEM> ::= <ITEM>                                             291
366 <FUNCTION INTRINSIC> ::= <ARITHMETIC INTRINSIC>                          418
                           / <BOOLEAN INTRINSIC>                             419
367 <PROCEDURE INTRINSIC> ::= ALLOW
                            / BUZZ
                            / DISALLOW
                            / EXIT
                            / HEYOU
                            / IIO
                            / MOVESTACK
                            / PAUSE
                            / RETURN
                            / SCANOUT
```

```
                        / STOP
                        / STOREITEM
                        / TIMER
                        / TOUCH
                        / ZAP
368 <WORD INTRINSIC> ::= TOPOFSTACK
369 <ARITHMETIC EXPRESSION LIST> ::= <ARITHMETIC EXPRESSION>                    97
                                   / <ARITHMETIC EXPRESSION LIST> ,            369
                                     <ARITHMETIC EXPRESSION>                   97
370 <BOOLEAN EXPRESSION LIST> ::= <BOOLEAN EXPRESSION>                         105
                                / <BOOLEAN EXPRESSION LIST> ,                  370
                                  <BOOLEAN EXPRESSION>                         105
371 <POINTER EXPRESSION LIST> ::= <POINTER EXPRESSION>                         218
                                / <POINTER EXPRESSION LIST> ,                  371
                                  <POINTER EXPRESSION>                         218
372 <ROW> ::= <ARITHMETIC EXPRESSION>                                          97
            / <ROW> , <ARITHMETIC EXPRESSION>                                  372  97
373 <POINTER DESIGNATOR> ::= POINTER ( <POINTER PARAMETERS> )                  420
374 <POINTER VARIABLE> ::= <VARIABLE>                                          263
375 <REPEAT PARAMETERS> ::= <EMPTY>                                            64
                          / ( <UNSIGNED INTEGER LIST> )                        421
376 <DENSITY MNEMONIC> ::= HIGH
                         / MEDIUM
                         / LOW
                         / SUPER
377 <ERRORTYPE MNEMONIC> ::= NOERROR
                           / SUNOTREADY
                           / READPARITYERROR
                           / READCHECKFAILURE
378 <EXTMODE MNEMONIC> ::= SINGLE
                         / BCL
                         / EBCDIC
379 <FILEKIND MNEMONIC> ::= SYSTEMDIRECTORY
                           / VERSIONDIRECTORY
                           / DIRECTORY
                           / CONTROLDECK
                           / BACKUPDISK
                           / RECONSTRUCTIONFILE
                           / SYSTEMDIRFILE
                           / COMPILERCODE
                           / LIBRARYCODE
```

```
                        / INTRINSICFILE
                        / MCPCODEFILE
                        / ALGOLCODE
                        / COBOLCODE
                        / FORTRANCODE
                        / XALGOLCODE
                        / PLICODE
                        / JOVIALCODE
                        / ESPOLCODE
                        / DCALGOLCODE
                        / BASICCODE
                        / XFORTRANCODE
                        / BOUNDCODE
                        / CODEFILE
                        / ALGOLSYMBOL
                        / COBOLSYMBOL
                        / FORTRANSYMBOL
                        / XALGOLSYMBOL
                        / PLISYMBOL
                        / JOVIALSYMBOL
                        / ESPOLSYMBOL
                        / DCALGOLSYMBOL
                        / BASICSYMBOL
                        / XFORTRANSYMBOL
                        / DATA
380 <INTMODE MNEMONIC> ::= SINGLE
                         / BCL
                         / EBCDIC
                         / ASCII
381 <KIND MNEMONIC> ::= DISK
                      / DISKPACK
                      / DISPLAY
                      / PAPER
                      / PAPERPUNCH
                      / PAPERREADER
                      / PETAPE
                      / PRINTER
                      / PUNCH
                      / READER
                      / REMOTE
                      / TAPE
```

```
                        / TAPE7
                        / TAPE9
382 <LABELTYPE MNEMONIC> ::= STANDARD
                           / OMITTED
                           / OMITTEDEOF
383 <MYUSE MNEMONIC> ::= CLOSED
                       / IN
                       / OUT
                       / IO
384 <OTHERUSE MNEMONIC> ::= SECURED
                          / IN
                          / OUT
                          / IO
385 <PARITY MNEMONIC> ::= STANDARD
                        / NONSTANDARD
386 <PROTECTION MNEMONIC> ::= TEMPORARY
                            / SAVE
                            / PROTECTED
387 <SECURITYTYPE MNEMONIC> ::= PRIVATE
                              / CLASSA
                              / CLASSB
                              / CLASSC
388 <SECURITYUSE MNEMONIC> ::= SECURED
                             / IN
                             / OUT
                             / IO
389 <SIZEMODE MNEMONIC> ::= SINGLE
                          / HEX
                          / BCL
                          / EBCDIC
390 <SPEED MNEMONIC> ::= FAST
                       / MEDIUMFAST
                       / MEDIUMSLOW
                       / SLOW
391 <STATE MNEMONIC> ::= NORMAL
                       / ATEND
                       / PARITYERROR
                       / DATA ERROR
                       / LOCKEDOUT
                       / NOINPUT
                       / BREAKHERE
```

```
                        / TIMEOUT
                        / NEWUSER
392 <UNITS MNEMONIC> ::= WORDS
                        / CHARACTERS
393 <STRING CHARACTER> ::= <LETTER>                                    152
                          / <DIGIT>                                    153
                          / <SPECIAL CHARACTER>                        422
                          / <SINGLE SPACE>                               7
394 <ARRAY SPECIFIER> ::= <ARRAY IDENTIFIER LIST> [ <BOUND SPECIFIER> ] 423 424
395 <FIELD DESIGNATOR> ::= <OPERAND> . <FIELD IDENTIFIER>              396 158
                          / <ARRAY PRIMARY> . <FIELD IDENTIFIER>       322 158
396 <OPERAND> ::= <ARITHMETIC VARIABLE> ( <ARITHMETIC EXPRESSION> )    243  97
                / <ARITHMETIC FUNCTION DESIGNATOR>                     425
                / <CASE HEAD> ( <ARITHMETIC EXPRESSION LIST> )         197 369
397 <CONCATENATION> ::= [ <LEFT BIT-TO> : <NUMBER OF BITS> ]           426 427
                      / [ <LEFT BIT-TO> : <LEFT BIT-FROM> :            426 428
                        <NUMBER OF BITS> ]                             427
398 <DECIMAL FRACTION> ::= . <UNSIGNED INTEGER>                        229
399 <OCTAL CONSTANT> ::= <OCTAL DIGIT>                                 429
                        / <OCTAL CONSTANT> <OCTAL DIGIT>               399 429
400 <BINARY CODE> ::= 1
                    / 10
                    / 12
                    / 13
                    / 14
                    / 16
                    / 17
                    / 18
                    / 120
                    / 130
                    / 140
                    / 160
                    / 170
                    / 180
401 <BINARY STRING> ::= <BINARY CHARACTER>                             430
                        / <BINARY STRING> <BINARY CHARACTER>           401 430
402 <QUATERNARY CODE> ::= 2
                        / 20
                        / 24
                        / 26
                        / 28
```

```
                        / 240
                        / 260
                        / 280
403 <QUATERNARY STRING> ::= <QUATERNARY CHARACTER>                          431
                          / <QUATERNARY STRING> <QUATERNARY CHARACTER>      403 431
404 <OCTAL CODE> ::= 3
                   / 30
                   / 36
                   / 360
405 <OCTAL STRING> ::= <OCTAL CHARACTER>                                    432
                     / <OCTAL STRING> <OCTAL CHARACTER>                     405 432
406 <HEXADECIMAL CODE> ::= 4
                         / 40
407 <HEXADECIMAL STRING> ::= <HEXADECIMAL CHARACTER>                        433
                           / <HEXADECIMAL STRING>                           407
                             <HEXADECIMAL CHARACTER>                        433
408 <BCL CODE> ::= <EMPTY>                                                   64
                 / 6
                 / 60
409 <BCL STRING> ::= <QUOTE>                                                464
                   / <BCL CHARACTER>                                        434
                   / <BCL STRING> <BCL CHARACTER>                           409 434
410 <ASCII CODE> ::= 7
                   / 70
411 <ASCII STRING> ::= <BCL STRING>                                         409
412 <EBCDIC CODE> ::= 8
                    / 80
413 <EBCDIC STRING> ::= <BCL STRING>                                        409
414 <BOOLEAN SECONDARY> ::= <BOOLEAN PRIMARY>                               435
                          / NOT <BOOLEAN PRIMARY>                           435
415 <FIELD VALUE LIST> ::= <FIELD VALUE>                                    436
                         / <FIELD VALUE LIST> , <FIELD VALUE>               415 436
416 <CHARACTER> ::= <STRING CHARACTER>                                      393
                  / <STRING BRACKET CHARACTER>                              437
                  / <INVALID CHARACTER>                                     438
417 <DELIMITER> ::= <BRACKET>                                               439
                  / <DECLARATOR>                                            440
                  / <OPERATOR>                                              441
                  / <SEPARATOR>                                             442
                  / <SPECIFICATOR>                                          443
418 <ARITHMETIC INTRINSIC> ::= ABS
```

```
                    / BOOLEAN
                    / DABS
                    / DECIMAL
                    / DINTEGER
                    / DMAX
                    / DMIN
                    / DNABS
                    / DOUBLE
                    / DSCALELEFT
                    / DUPLICATE
                    / ENTIER
                    / EVAL
                    / EXCHANGE
                    / FIRSTONE
                    / FIRSTWORD
                    / INTEGER
                    / LISTLOOKUP
                    / MASKSEARCH
                    / MAX
                    / MIN
                    / MYSELF
                    / NABS
                    / NAME
                    / NORMALIZE
                    / OCRX
                    / ONES
                    / REAL
                    / SCALELEFT
                    / SCALERIGHT
                    / SCALERIGHTF
                    / SCALERIGHTT
                    / SCANIN
                    / SECONDWORD
                    / SIZE
                    / STUFF
                    / XSIGN
419 <BOOLEAN INTRINSIC> ::= AVAILABLE
                          / BUSY
                          / HAPPENED
                          / OVERFLOW
                          / TOGGLE
```

```
                          / UNLOCK
420 <POINTER PARAMETERS> ::= <ARRAY PART>                                  444
                           / <ARRAY PART> , <CHARACTER SIZE>               444 445
421 <UNSIGNED INTEGER LIST> ::= <ARITHMETIC EXPRESSION>                    97
                              / <ARITHMETIC EXPRESSION> ,                  97
                                <UNSIGNED INTEGER LIST>                    421
422 <SPECIAL CHARACTER> ::= .
                          / ,
                          / [
                          / ]
                          / (
                          / )
                          / +
                          / -
                          / <SLASH>                                        353
                          / >
                          / <
                          / ≤
                          / ≥
                          / =
                          / ≠
                          / ←
423 <ARRAY IDENTIFIER LIST> ::= <ARRAY IDENTIFIER>                         148
                              / <ARRAY IDENTIFIER LIST> ,                  423
                                <ARRAY IDENTIFIER>                         148
424 <BOUND SPECIFIER> ::= *
                        / <BOUND SPECIFIER> , *                            424
425 <ARITHMETIC FUNCTION DESIGNATOR> ::= <FUNCTION DESIGNATOR>             69
                                       / <ARITHMETIC INTRINSIC>            418
426 <LEFT BIT-TO> ::= <ARITHMETIC EXPRESSION>                              97
427 <NUMBER OF BITS> ::= <ARITHMETIC EXPRESSION>                           97
428 <LEFT BIT-FROM> ::= <ARITHMETIC EXPRESSION>                            97
429 <OCTAL DIGIT> ::= 0
                    / 1
                    / 2
                    / 3
                    / 4
                    / 5
                    / 6
                    / 7
430 <BINARY CHARACTER> ::= 0
```

```
                          / 1
431 <QUATERNARY CHARACTER> ::= 0
                              / 1
                              / 2
                              / 3
432 <OCTAL CHARACTER> ::= 0
                         / 1
                         / 2
                         / 3
                         / 4
                         / 5
                         / 6
                         / 7
433 <HEXADECIMAL CHARACTER> ::= 0
                               / 1
                               / 2
                               / 3
                               / 4
                               / 5
                               / 6
                               / 7
                               / 8
                               / 9
                               / A
                               / B
                               / C
                               / D
                               / E
                               / F
434 <BCL CHARACTER> ::= <STRING CHARACTER>                                  393
435 <BOOLEAN PRIMARY> ::= <LOGICAL VALUE>                                   253
                         / <RELATION>                                       446
                         / <BOOLEAN ITEM>                                   447
                         / <BOOLEAN FIELD DESIGNATOR>                       448
                         / <BOOLEAN OPERAND>                                449
                         / <BOOLEAN PRIMARY> & <BOOLEAN EXPRESSION>         435 105
                           <CONCATENATION>                                  397
                         / <BOOLEAN PRIMARY> & <BOOLEAN LAYOUT>             435 450
                         / <TABLE MEMBERSHIP>                               451
                         / <VALUE DESIGNATOR>                               327
436 <FIELD VALUE> ::= <EMPTY>                                               64
```

```
                    / <ARITHMETIC EXPRESSION>                                97
                    / *
437 <STRING BRACKET CHARACTER> ::= <QUOTE>                                   464
438 <INVALID CHARACTER> ::= @
439 <BRACKET> ::= (
                / )
                / [
                / ]
440 <DECLARATOR> ::= ARRAY
                   / BOOLEAN
                   / DEFINE
                   / DOUBLE
                   / EVENT
                   / FIELD
                   / FORWARD
                   / INTEGER
                   / LABEL
                   / LAYOUT
                   / MONITOR
                   / ON
                   / OWN
                   / PICTURE
                   / POINTER
                   / PROCEDURE
                   / QUEUE
                   / REAL
                   / REFERENCE
                   / SAVE
                   / USING
                   / WORD
441 <OPERATOR> ::= <ARITHMETIC OPERATOR>                                     452
                 / <CONCATENATE OPERATOR>                                    453
                 / <LOGICAL OPERATOR>                                        454
                 / <RELATIONAL OPERATOR>                                     286
                 / <REPLACEMENT OPERATOR>                                    103
                 / <SEQUENTIAL OPERATOR>                                     455
442 <SEPARATOR> ::= ,
                  / .
                  / @
                  / :
                  / ;
```

```
                    / <SPACE>                                                    3
                    / BY
                    / COMMENT
                    / IN
                    / NULL
                    / OF
                    / STEP
                    / UNTIL
                    / WHILE
                    / WITH
443 <SPECIFICATOR> ::= VALUE
444 <ARRAY PART> ::= <ARRAY ROW>                                                 277
                   / <SUBSCRIPTED VARIABLE>                                      278
                   / <ARRAY IDENTIFIER>                                          148
445 <CHARACTER SIZE> ::= 4
                       / 6
                       / 8
                       / *
446 <RELATION> ::= <ARITHMETIC RELATION>                                         456
                 / <REFERENCE RELATION>                                          457
                 / <STRING RELATION>                                             458
                 / <POINTER RELATION>                                            459
447 <BOOLEAN ITEM> ::= <ITEM>                                                    291
448 <BOOLEAN FIELD DESIGNATOR> ::= <BOOLEAN OPERAND> . <FIELD IDENTIFIER>        449 158
449 <BOOLEAN OPERAND> ::= ( <BOOLEAN EXPRESSION> )                               105
                        / <BOOLEAN VARIABLE>                                     256
                        / <BOOLEAN FUNCTION DESIGNATOR>                          460
                        / <CASE HEAD> ( <BOOLEAN EXPRESSION LIST> )              197 370
450 <BOOLEAN LAYOUT> ::= <LAYOUT>                                                363
451 <TABLE MEMBERSHIP> ::= <ARITHMETIC EXPRESSION> IN <TABLE POINTER>             97 461
                         / <POINTER EXPRESSION> IN <TABLE POINTER>               218 461
452 <ARITHMETIC OPERATOR> ::= +
                            / -
                            / *
                            / <SLASH>                                            353
                            / DIV
                            / MOD
                            / MUX
453 <CONCATENATE OPERATOR> ::= &
454 <LOGICAL OPERATOR> ::= AND
                         / EQV
```

```
                        / IMP
                        / NOT
                        / OR
455 <SEQUENTIAL OPERATOR> ::= CASE
                           / DO
                           / ELSE
                           / FOR
                           / GO
                           / IF
                           / THEN
                           / TO
456 <ARITHMETIC RELATION> ::= <ARITHMETIC EXPRESSION>                    97
                              <ARITHMETIC RELATIONAL>                   462
                              <ARITHMETIC EXPRESSION>                    97
457 <REFERENCE RELATION> ::= <REFERENCE EXPRESSION> <REFERENCE RELATIONAL>  191 463
                             <REFERENCE EXPRESSION>                      191
458 <STRING RELATION> ::= <POINTER EXPRESSION> <RELATIONAL OPERATOR>      218 286
                          <POINTER EXPRESSION> FOR                       218
                          <ARITHMETIC EXPRESSION>                         97
459 <POINTER RELATION> ::= <POINTER EXPRESSION> <REFERENCE RELATIONAL>    218 463
                           <POINTER EXPRESSION>                          218
460 <BOOLEAN FUNCTION DESIGNATOR> ::= <FUNCTION DESIGNATOR>               69
                                    / <BOOLEAN INTRINSIC>                419
461 <TABLE POINTER> ::= ALPHA
                      / ALPHA6
                      / ALPHA8
                      / <SUBSCRIPTED VARIABLE>                          278
462 <ARITHMETIC RELATIONAL> ::= <RELATIONAL OPERATOR>                    286
                              / IS
463 <REFERENCE RELATIONAL> ::= =
                             / ≠
464 QUOTE ::= "
```

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

THE METALINGUISTIC VARIABLES UTILIZED THROUGHOUT THE ESPOL LANGUAGE ARE OUTLINED IN THIS APPENDIX. METALINGUISTIC SYMBOLS ARE COMBINED TO FORM A METALINGUISTIC FORMULA, WHICH CONSISTS OF METALINGUISTIC VARIABLES. EACH METALINGUISTIC VARIABLE IS ASSOCIATED WITH A CORRESPONDING "LINE NUMBER COLUMN" AND "PAGE NUMBER COLUMN". THE LINE NUMBER REFERS TO THE CORRESPONDING LINE NUMBER IN APPENDIX C, WHICH CONTAINS THE COMPLETE SYNTAX FOR THE METALINGUISTIC VARIABLES THE PAGE NUMBER TO THE LEFT OF THE SEMICOLON REFERS TO THE TEXTUAL PAGE WHICH FIRST USES/DESCRIBES THE ITEM IN THE ESPOL SYNTAX. THE PAGE NUMBERS TO THE RIGHT OF THE SEMICOLON SHOW THOSE PAGES WHERE THE SYNTACTICAL ITEM IS USED EITHER IN ANOTHER METALINGUISTIC FORMULA, OR USED WITHIN SEMANTIC EXPLANATIONS.

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS

APPENDIX D. INDEX OF METALINGUISTIC VARIABLES AND RESERVED WORDS