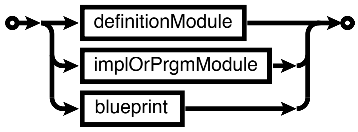
The Syntax Of Modula-2 — Revision 2010

Copyright © 2010-15 B.Kowarsch & R.Sutcliffe; Status: Aug 31, 2015

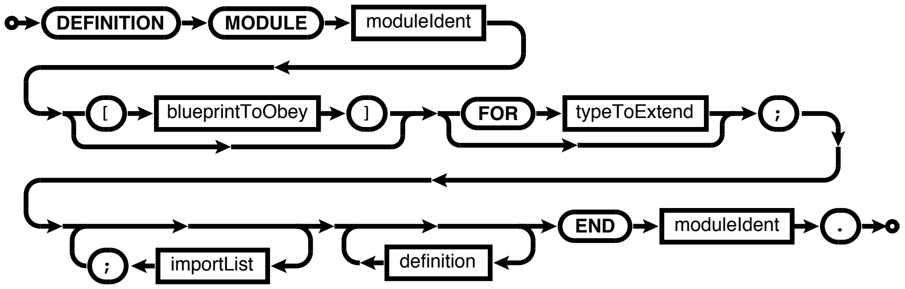
(1) Non-Terminals

#1 Compilation Unit

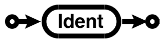


Definition Module Syntax

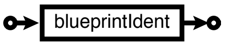
#2 Defintion Module



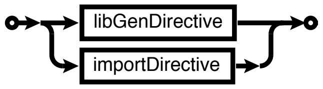
#2.1 Module Identifier, Blueprint Identifier, Type to Extend



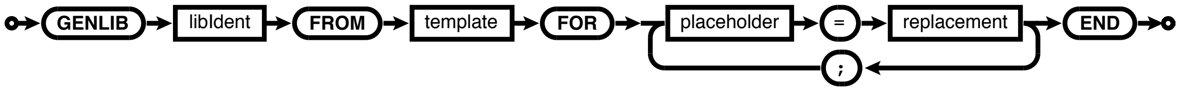
#2.2 Blueprint to Obey



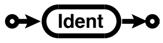
#3 Import List



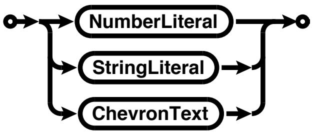
#4 Library Generation Directive



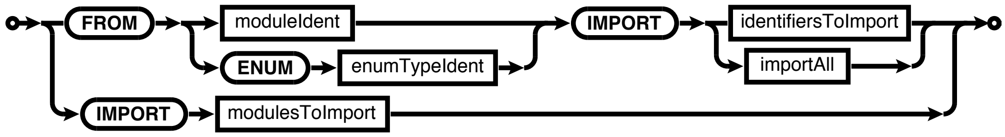
#4.1 Library Identifier, Template, Placeholder



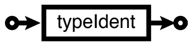
#4.2 Replacement



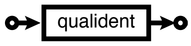
#5 Import Directive



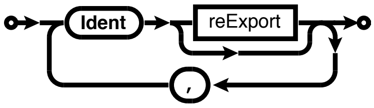
#5.1 Enumeration Type Identifier



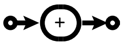
#5.2 Type Identifier



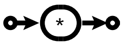
#5.3 Modules to Import, Identifiers to Import



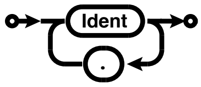
#5.4 Re-Export



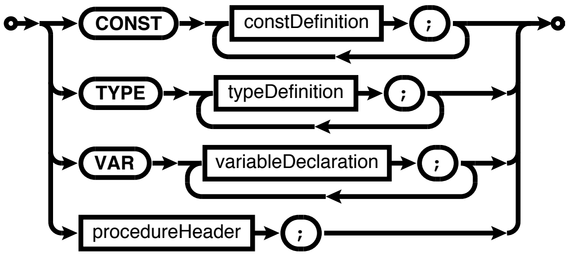
#5.5 Import All



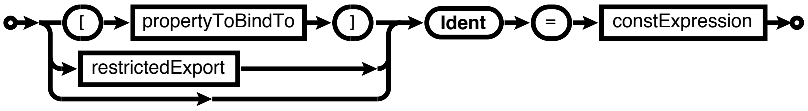
#6 Qualified Identifier



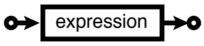
#7 Definition



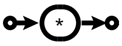
#8 Constant Definition



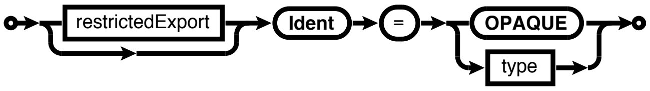
#8.1 Constant Expression



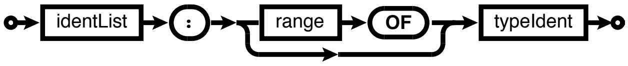
#8.2 Restricted Export



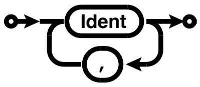
#9 Type Definition



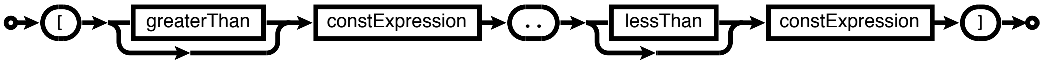
#10 Variable Declaration



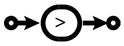
#11 Identifier List



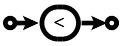
#12 Range



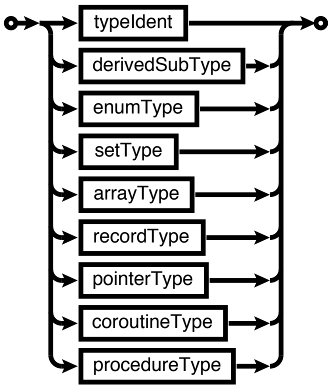
#12.1 Greater Than



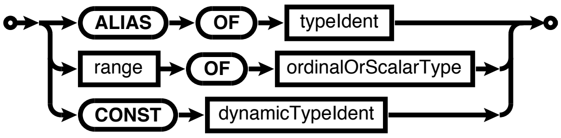
#12.2 Less Than



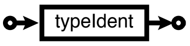
#13 Type



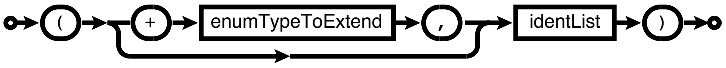
#13.1 Derived Sub-Type



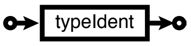
#13.2 Ordinal or Scalar Type, Dynamic Type Identifier



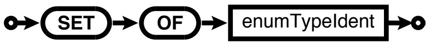
#14 Enumeration Type



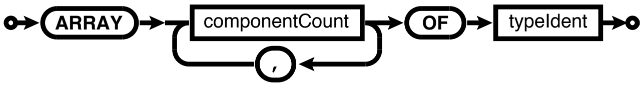
#14.1 Enumeration Type to Extend



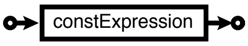
#15 Set Type



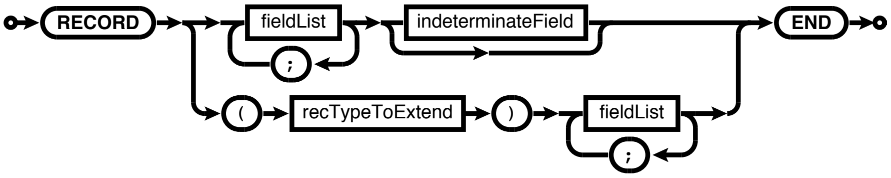
#16 Array Type



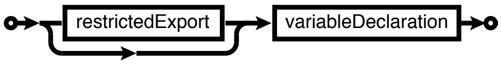
#16.1 Component Count



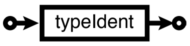
#17 Record Type



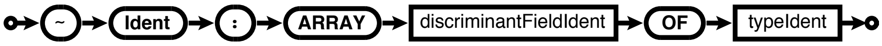
#17.1 Field List



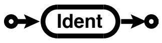
#17.2 Record Type to Extend



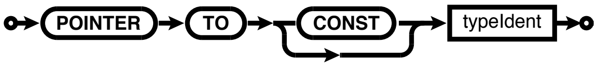
#17.3 Indeterminate Field



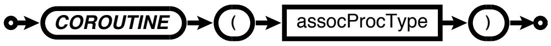
#17.4 Discriminant Field Identifier



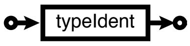
#18 Pointer Type



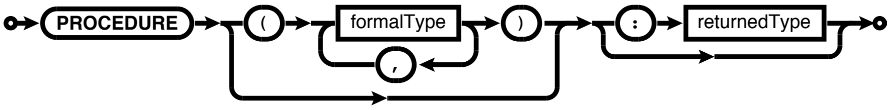
#19 Coroutine Type



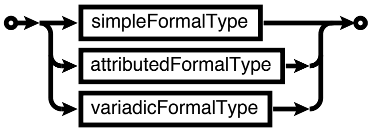
#19.1 Associated Procedure Type



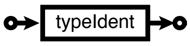
#20 Procedure Type



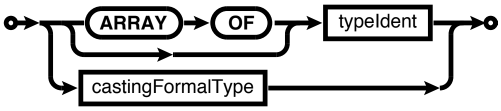
#20.1 Formal Type



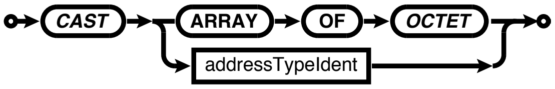
#20.2 Returned Type



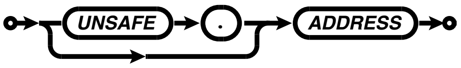
#21 Simple Formal Type



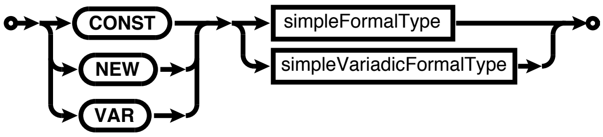
#21.1 Casting Formal Type



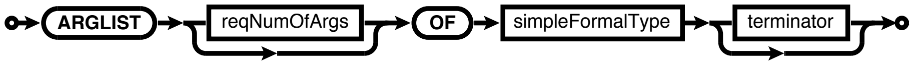
#21.2 Address Type Identifier



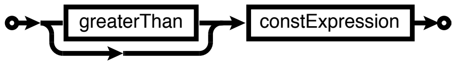
#22 Attributed Formal Type



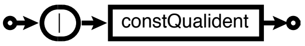
#23 Simple Variadic Formal Type



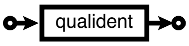
#23.1 Required Number of Arguments



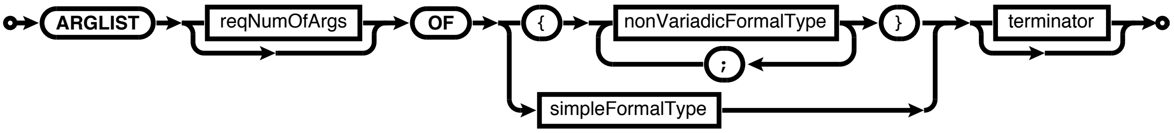
#23.2 Argument List Terminator



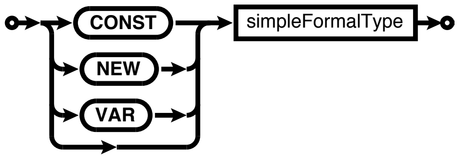
#23.3 Constant Qualified Identifier



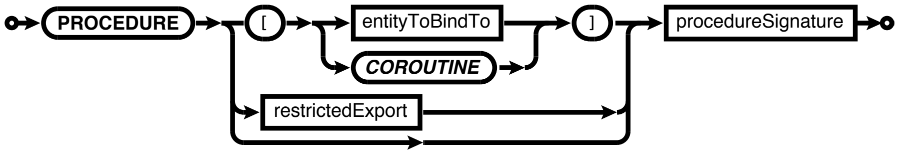
#24 Variadic Formal Type



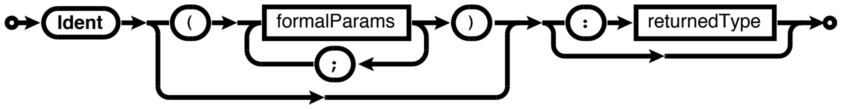
#25 Non-Variadic Formal Type



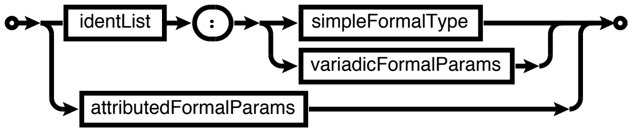
#26 Procedure Header



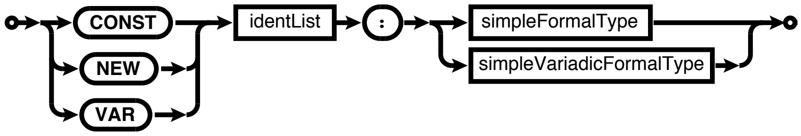
#27 Procedure Signature



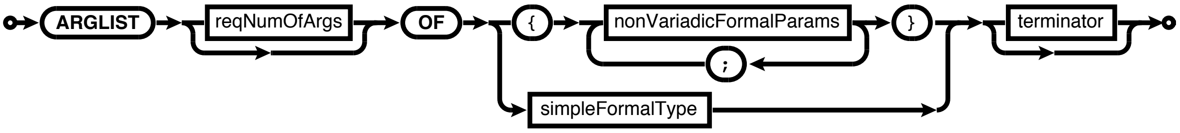
#28 Formal Parameters



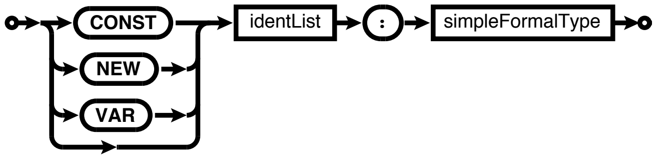
#29 Attributed Formal Parameters



#30 Variadic Formal Parameters

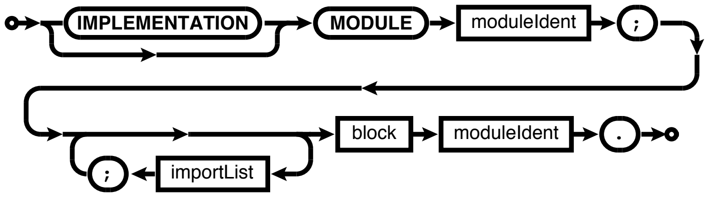


#31 Non-Variadic Formal Parameters

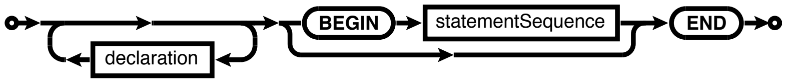


Implementation and Program Module Syntax

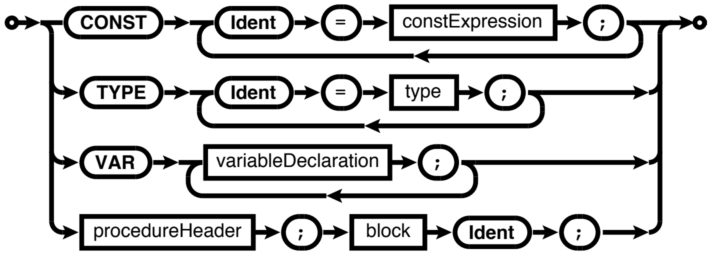
#32 Implementation or Program Module



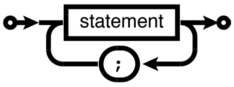
#33 Block



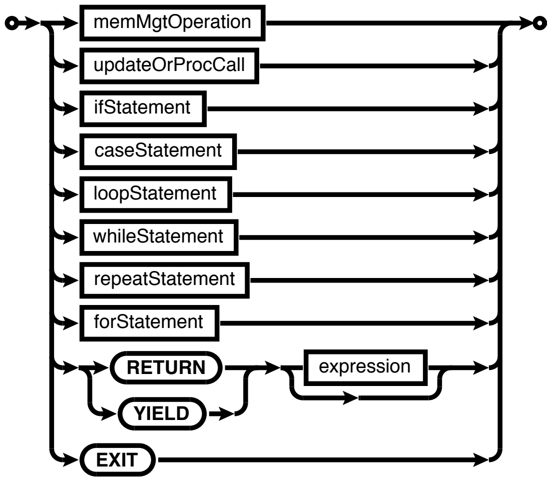
#34 Declaration



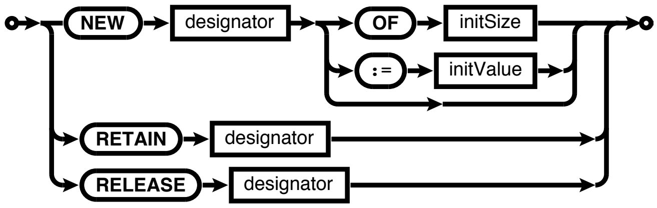
#35 Statement Sequence



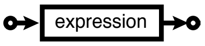
#36 Statement



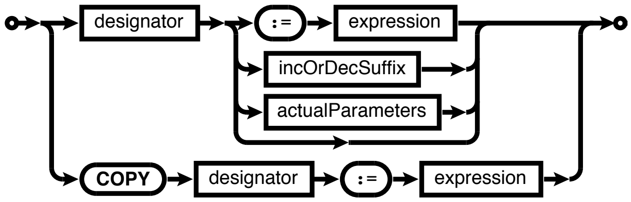
#37 Memory Management Operation



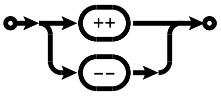
#37.1 Initialisation Size, Initialisation Value



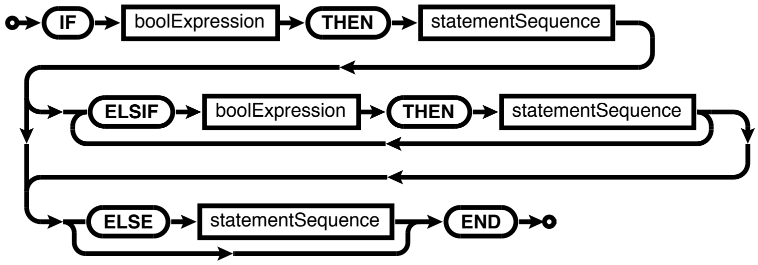
#38 Update or Procedure Call



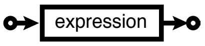
#38.1 Increment or Decrement Suffix



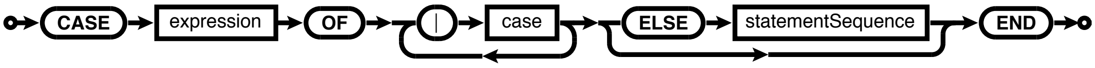
#39 IF Statement



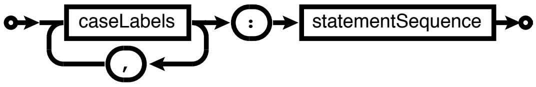
#39.1 Boolean Expression



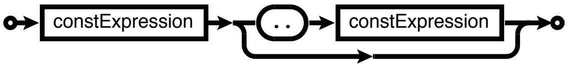
#40 CASE Statement



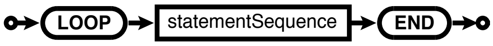
#40.1 Case



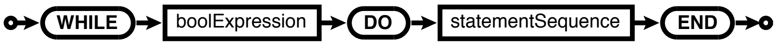
#40.2 Case Labels



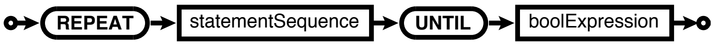
#41 LOOP Statement



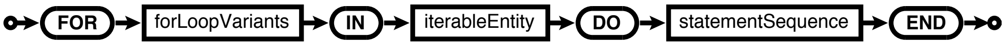
#42 WHILE Statement



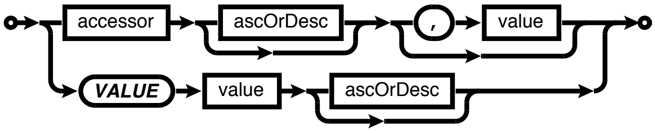
#43 REPEAT Statement



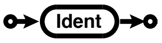
#44 FOR Statement



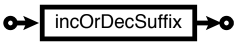
#44.1 FOR Loop Variants



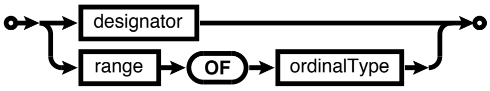
#44.2 Accessor, Value



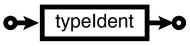
#44.3 Ascender or Descender



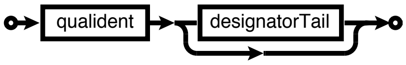
#44.4 Iterable Entity



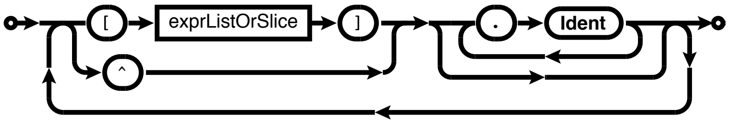
#44.5 Ordinal Type



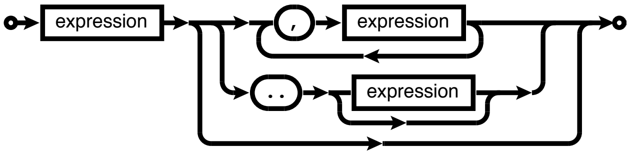
#45 Designator



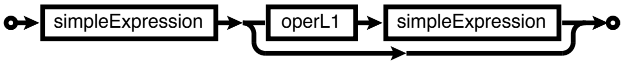
#45.1 Designator Tail



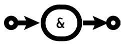
#45.2 Expression List or Slice



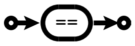
#46 Expression (Evaluation Level 1)



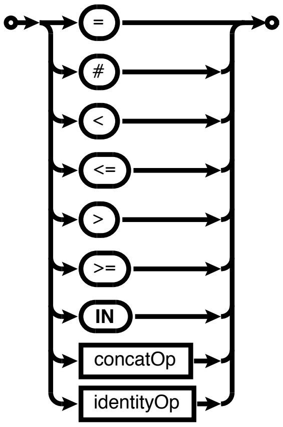
#46.2 Concatenation Operator



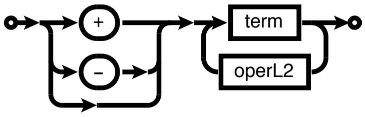
#46.3 Identity Operator



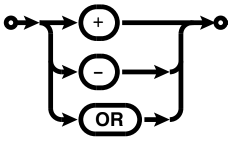
#46.1 Level-1 Operator



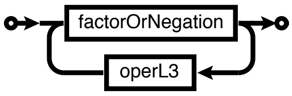
#47 Simple Expression (Evaluation Level 2)



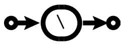
#47.1 Level-2 Operator



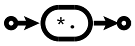
#48 Term (Evaluation Level 3)



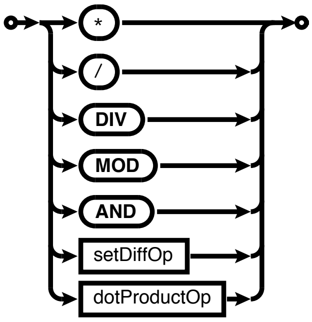
#48.2 Set Difference Operator



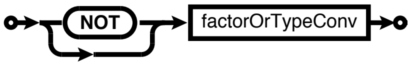
#48.3 Dot Product Operator



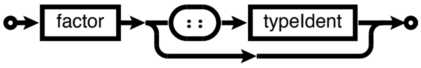
#48.1 Level-3 Operator



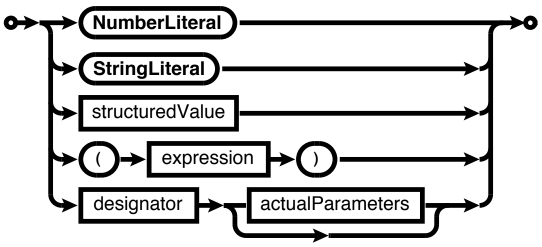
#49 Factor or Negation (Evaluation Level 4)



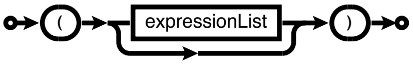
#50 Factor or Type Conversion (Evaluation Level 5)



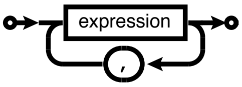
#51 Factor (Evaluation Level 6)



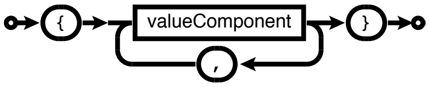
#52 Actual Parameters



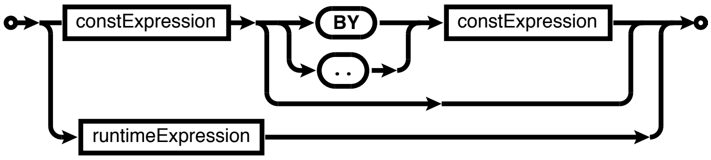
#53 Expression List



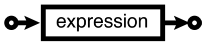
#54 Structured Value



#54.1 Value Component

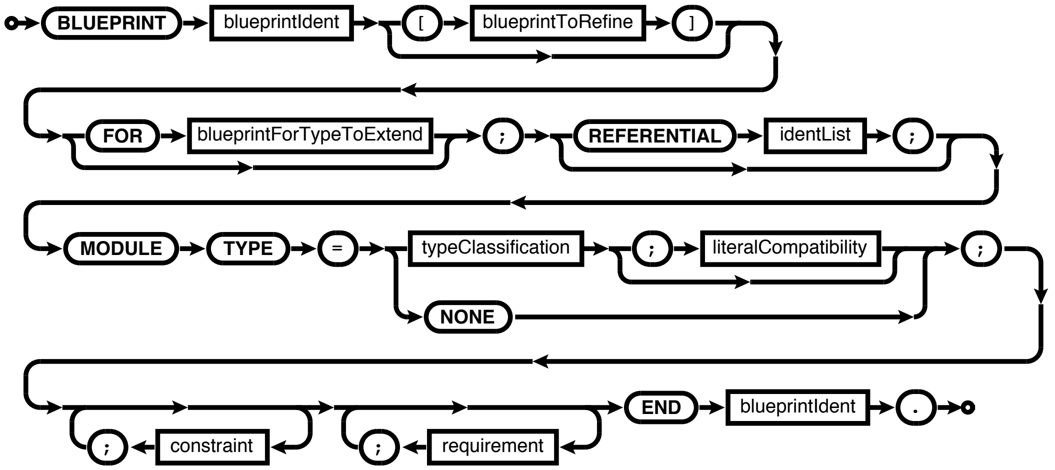


#54.2 Runtime Expression

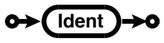


Blueprint Syntax

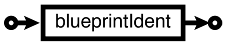
#55 Blueprint



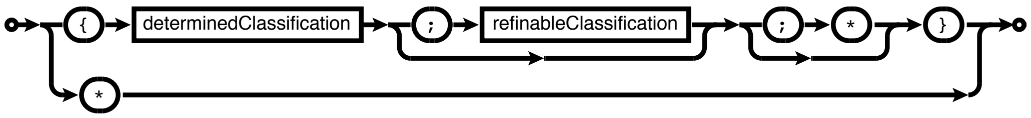
#55.1 Blueprint Identifier



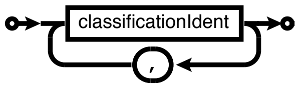
#55.2 Blueprint To Refine, Blueprint For Type To Extend



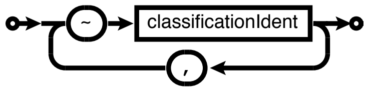
#56 Type Classification



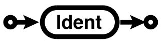
#56.1 Determined Classification



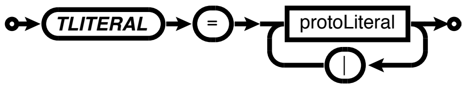
#56.2 Refinable Classification



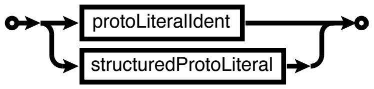
#56.3 Classification Identifier



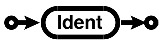
#57 Literal Compatibility



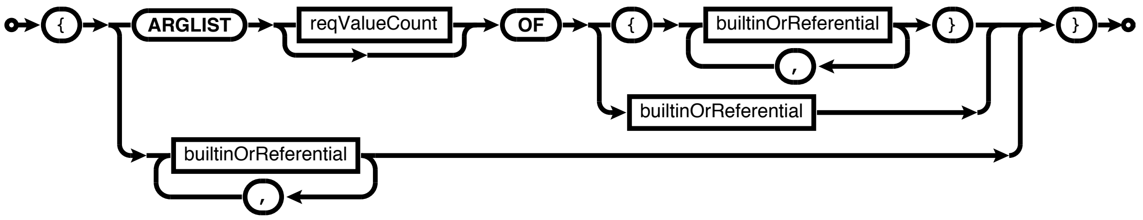
#57.1 Proto Literal



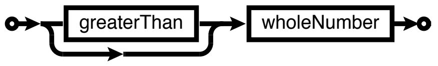
#57.2 Proto Literal Identifier



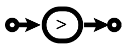
#58 Structured Proto Literal



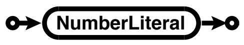
#58.1 Required Value Count



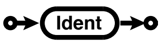
#58.2 Greater Than



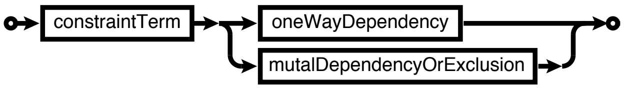
#58.3 Whole Number



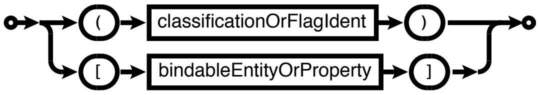
#58.4 Built-in Type Or Referential



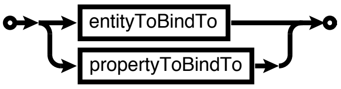
#59 Constraint



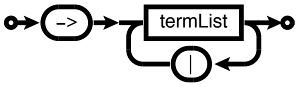
#59.1 Constraint Term



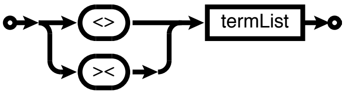
#59.2 Bindable Entity Or Property



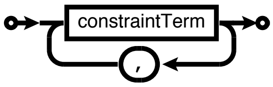
#59.3 One-Way Dependency



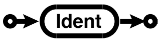
#59.4 Mutual Dependency Or Exclusion



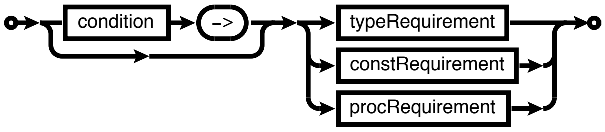
#59.5 Term List



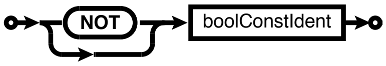
#59.6 Classification Or Flag Identifier



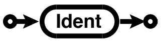
#60 Requirement



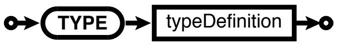
#60.1 Condition



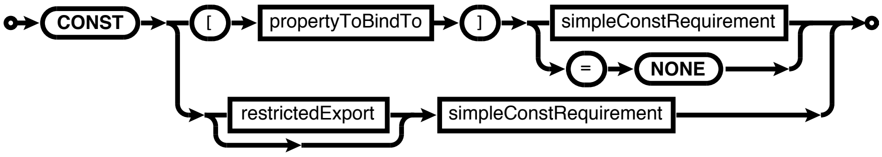
#60.2 Boolean Constant Identifier



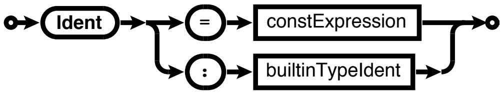
#60.3 Type Requirement



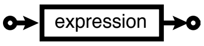
#61 Constant Requirement



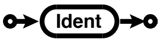
#61.1 Simple Constant Requirement



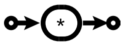
#61.2 Constant Expression



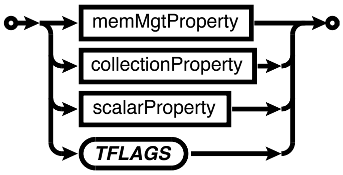
#61.3 Built-in Type Identifier



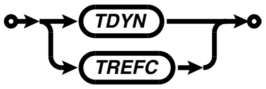
#61.4 Restricted Export



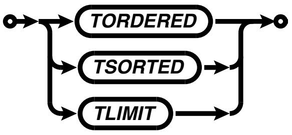
#62 Property To Bind To



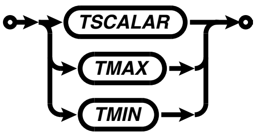
#62.1 Memory Management Property



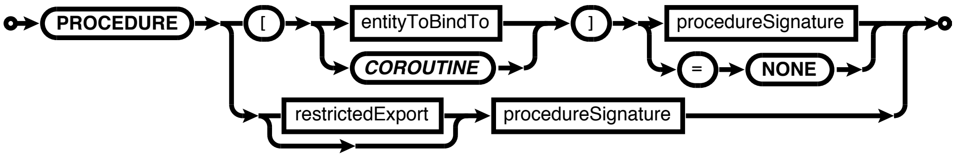
#62.2 Collection Property



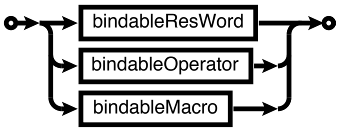
#62.3 Scalar Property



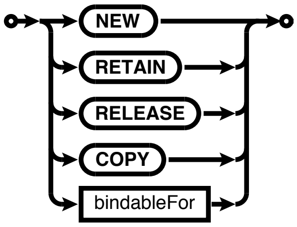
#63 Procedure Requirement



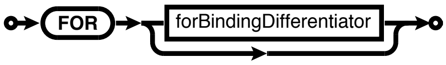
#64 Entity To Bind To



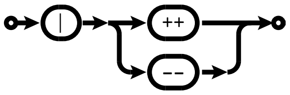
#64.1 Bindable Reserved Word



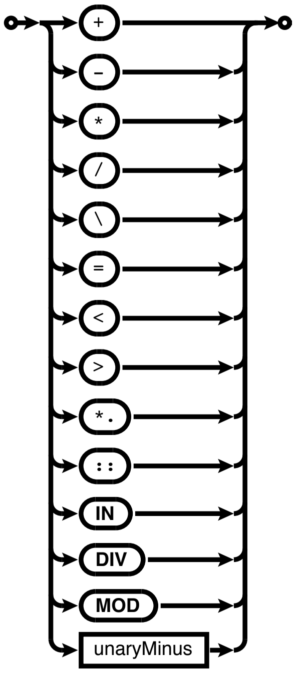
#64.2 Bindable FOR



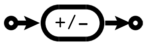
#64.3 FOR Binding Differentiator



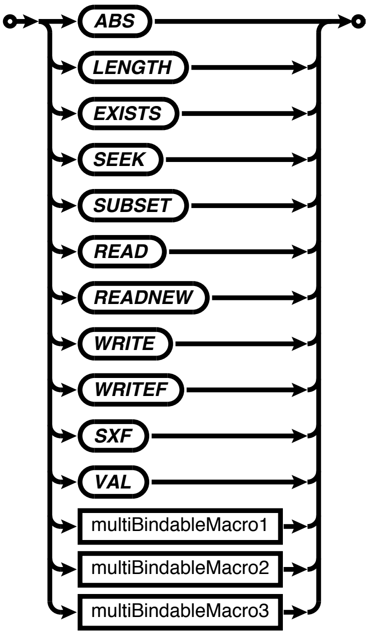
#64.4 Bindable Operator



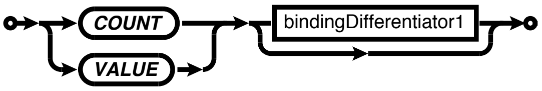
#64.5 Unary Minus



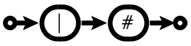
#64.6 Bindable Macro



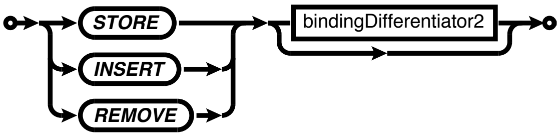
#64.7 Multi-Bindable Macro 1



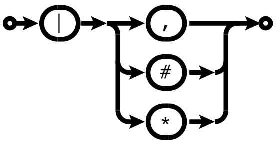
#64.8 Binding Differentiator 1



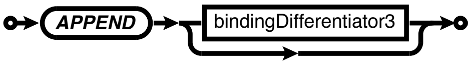
#64.9 Multi-Bindable Macro 2



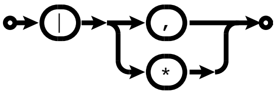
#64.10 Binding Differentiator 2



#64.11 Multi-Bindable Macro 3



#64.12 Binding Differentiator 3



(2) Terminals

#1 Reserved Words

ALIAS
AND
ARGLIST
ARRAY
BEGIN
BLUEPRINT
BY
CASE
CONST
COPY
DEFINITION
DIV
DO
ELSE
ELSIF
END
ENUM
EXIT
FOR
FROM
GENLIB
IF
IMPLEMENTATION
IMPORT
IN
LOOP
MOD
MODULE
NEW
NONE
NOT
OF
OPAQUE
OR
POINTER
PROCEDURE
RECORD
REFERENTIAL
RELEASE
REPEAT
RETAIN
RETURN
SET
THEN
TO
TYPE
UNTIL
VAR
WHILE
YIELD

#2 Dual-Use Identifiers (Schrödinger's Tokens)

ABS
ADDRESS
APPEND
CAST
COUNT
COROUTINE
EXISTS
INSERT
LENGTH
OCTET
READ
READNEW
REMOVE
SEEK
STORE
SUBSET
SXF
TDYN
TFLAGS
TLIMIT
TLITERAL
TMAX
TMIN
TORDERED
TREFC
TSCALAR
TSORTED
UNSAFE
VAL
VALUE
WRITE
WRITEF
ASM[*]
REG[*]

#3 Special Symbol Tokens

```
.      ~      +      =      ==     (      )
,      ..     -      #      &      [      ]
:      :=     *      >      ->     {      }
;      ++     *.     >=     <>
|      --     /      <      ><
^      ::     \      <=     +/-
```

#3.1 Quoted Text Delimiters

```
'      "      <<     >>
```

#3.2 Comment Delimiters

```
!      (*     *)
```

#3.3 Pragma Affix and Delimiters

```
?      <*     *>
```

#3.4 Template Language Symbols

```
##     <#     #>     @@     //     /*     */
```

#3.5 Reserved Symbols

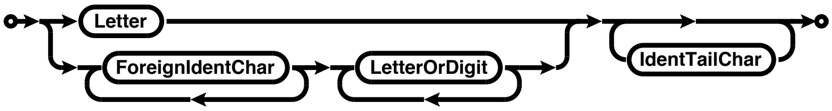
` for use as a token by Objective Modula-2

@ for use as lead character in identifiers and reserved words by language supersets

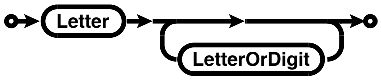
% for use as a character in identifiers and reserved words by implementations targeting OpenVMS

[*] *optional language facilities*

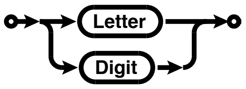
#4 Identifier



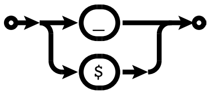
#4.1 Standard Library Identifier



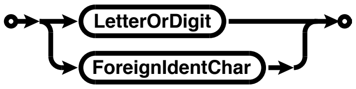
#4.2 Letter Or Digit



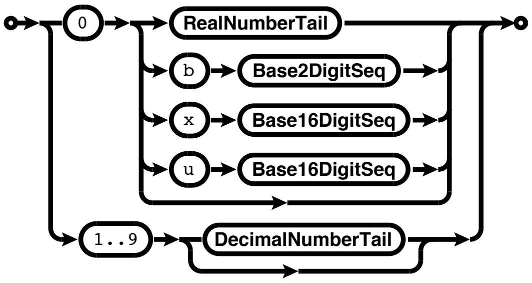
#4.3 Foreign Identifier Character



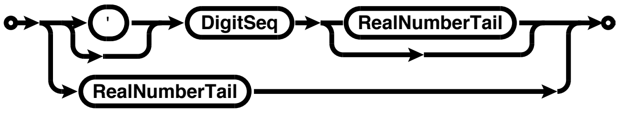
#4.4 Identifier Tail Character



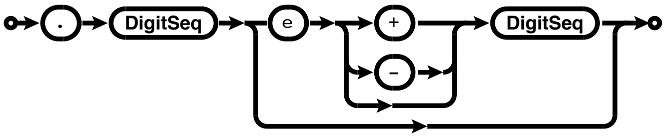
#5 Number Literal



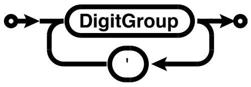
#5.1 Decimal Number Tail



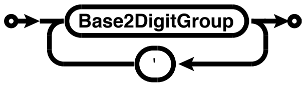
#5.2 Real Number Tail



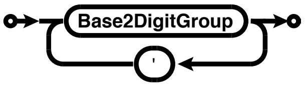
#5.3 Digit Sequence



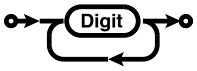
#5.4 Base-16 Digit Sequence



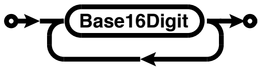
#5.5 Base-2 Digit Sequence



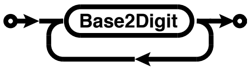
#5.3b Digit Group



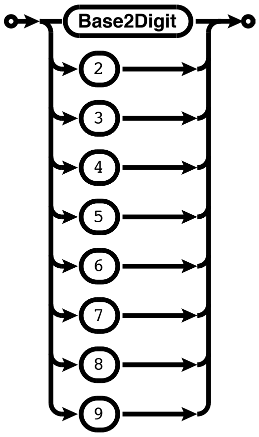
#5.4b Base-16 Digit Group



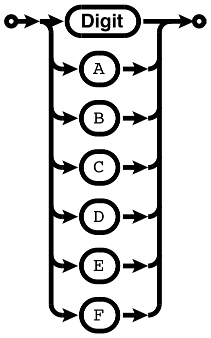
#5.5b Base-2 Digit Group



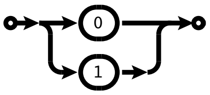
#5.6 Digit



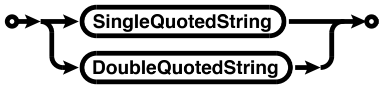
#5.7 Base-16 Digit



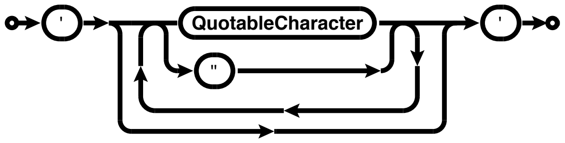
#5.8 Base-2 Digit



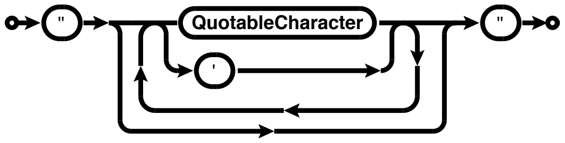
#6 String Literal



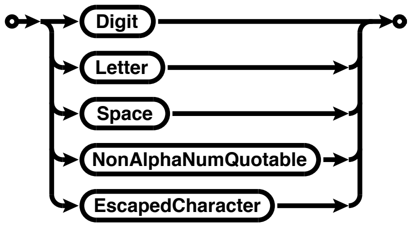
#6.1 Single Quoted String



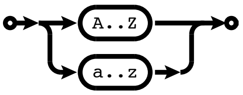
#6.2 Double Quoted String



#6.3 Quotable Character



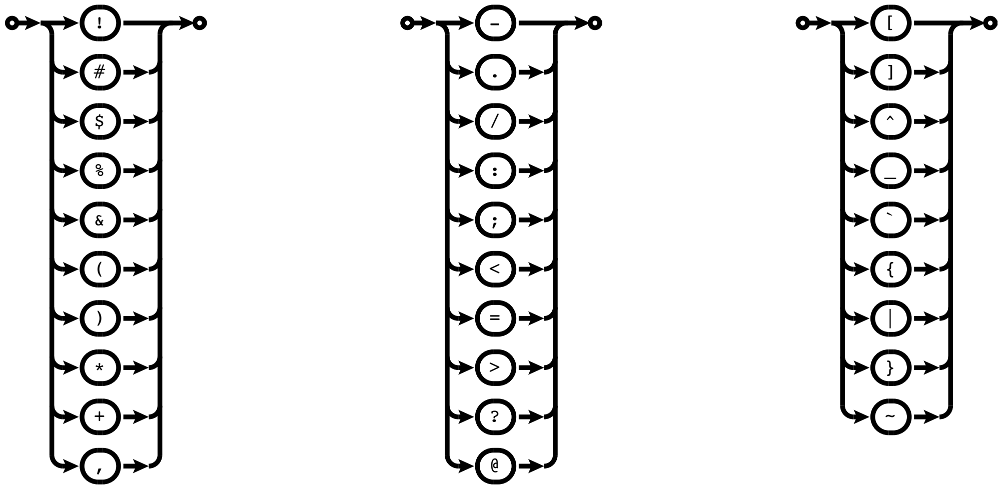
#6.4 Letter



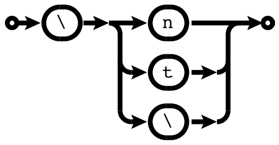
#6.5 Space

```
CONST Space = CHR(32);
```

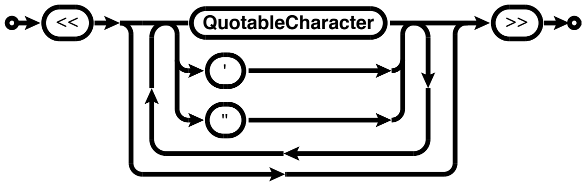
#6.6 Non-Alphanumeric Quotable Character



#6.7 Escaped Character

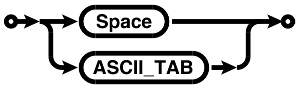


#7 Chevron Delimited Text



(3) Ignore Symbols

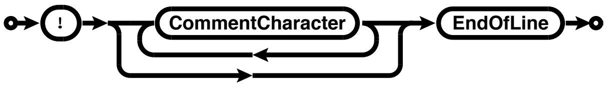
#1 Whitespace



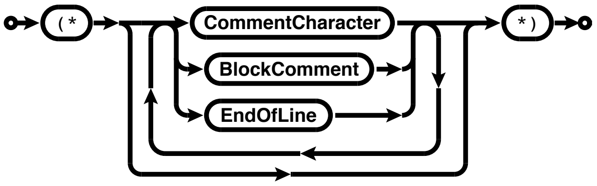
#1.1 ASCII Tabulator

```
CONST ASCII_TAB = CHR(8);
```

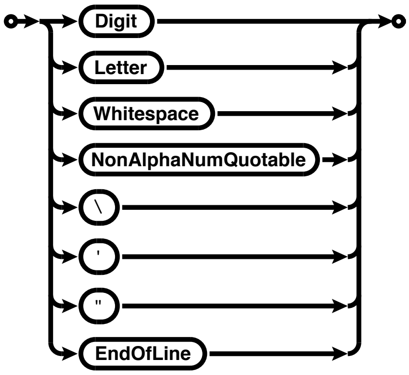
#2 Line Comment
(At the First Column of a Line)



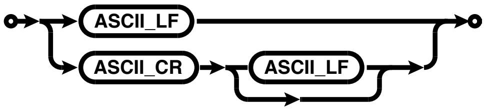
#3 Block Comment
(At Most Ten Nesting Levels)



#3.1 Comment Character



#4 End Of Line Marker



#4.1 ASCII Line Feed

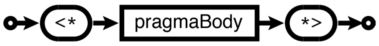
```
CONST ASCII_LF = CHR(10);
```

#4.2 ASCII Carriage Return

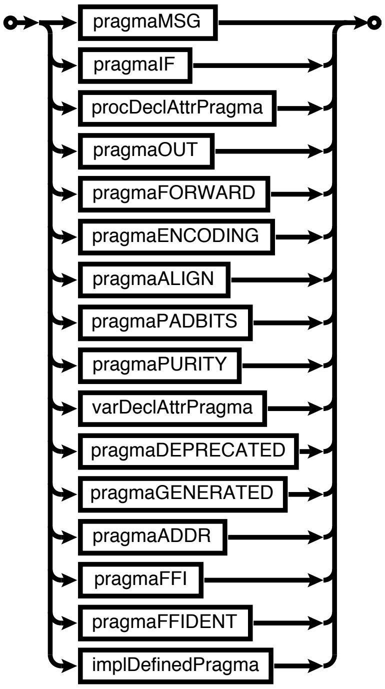
```
CONST ASCII_CR = CHR(13);
```

(5) Pragmas

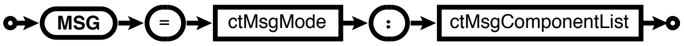
#1 Pragma



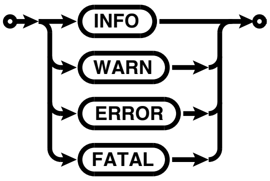
#1.1 Pragma Body



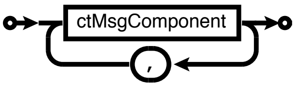
#2 Body Of Compile Time Message Pragma



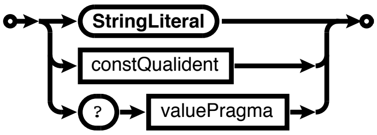
#2.1 Compile Time Message Mode



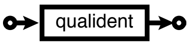
#2.2 Compile Time Message Component List



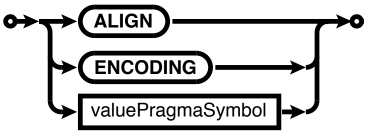
#2.3 Compile Time Message Component



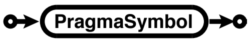
#2.4 Constant Qualified Identifier



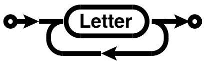
#2.5 Value Pragma



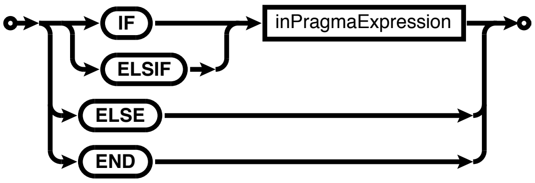
#2.6 Value Pragma Symbol



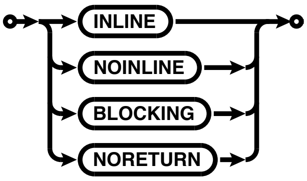
#2.7 Pragma Symbol



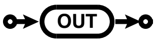
#3 Body Of Conditional Compilation Pragma



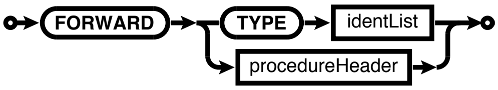
#4 Body Of Procedure Declaration Attribute Pragma



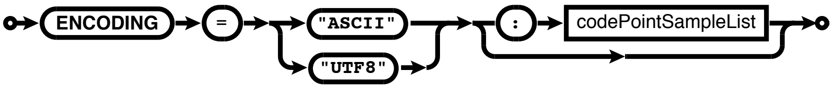
#5 Body Of Promise-To-Write Pragma



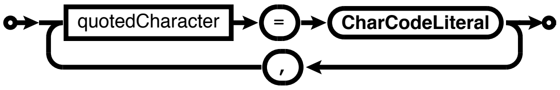
#6 Body Of Forward Declaration Pragma



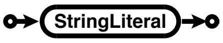
#7 Body Of Character Encoding Pragma



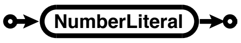
#7.1 Code Point Sample List



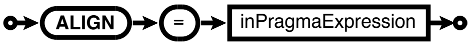
#7.2 Quoted Character



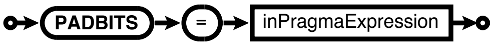
#7.3 Character Code Literal



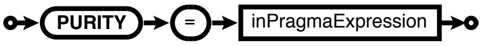
#8 Body Of Memory Alignment Pragma



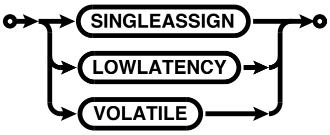
#9 Body Of Bit Padding Pragma



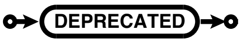
#10 Body Of Purity Attribute Pragma



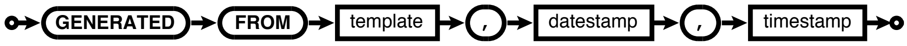
#11 Body Of Variable Declaration Attribute Pragma



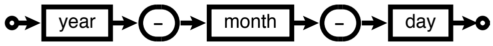
#12 Body Of Deprecation Pragma



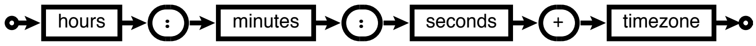
#13 Body Of Library Generation Timestamp Pragma



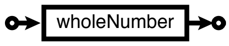
#13.1 Date Stamp



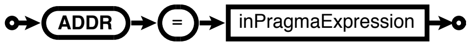
#13.2 Time Stamp



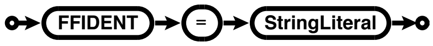
#13.3 Year, Month, Day, Hours, Minutes, Seconds, Timezone



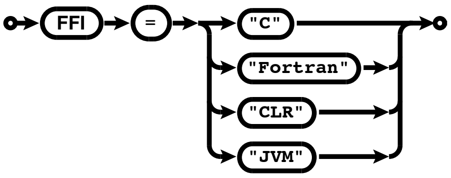
#14 Body Of Memory Mapping Pragma



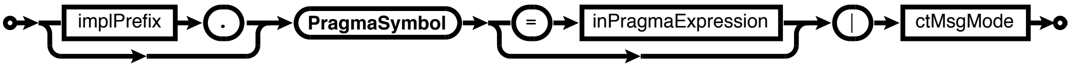
#16 Body Of Foreign Function Identifier Mapping Pragma



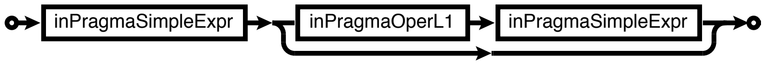
#15 Body Of Foreign Function Interface Pragma



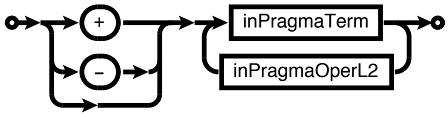
#17 Body Of Implementation Defined Pragma



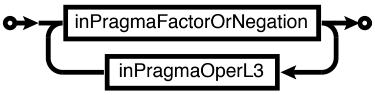
#18 In-Pragma Expression (Evaluation Level 1)



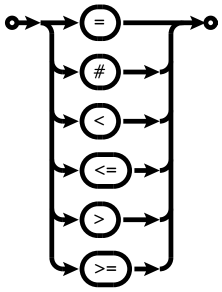
#19 In-Pragma Simple Expression (Evaluation Level 2)



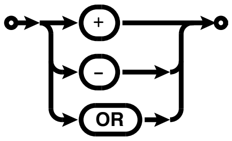
#20 In-Pragma Term (Evaluation Level 3)



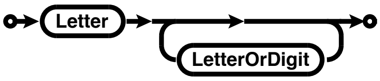
#18.1 In-Pragma Level-1 Operator



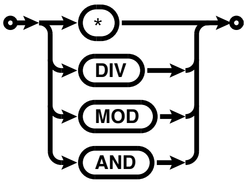
#19.1 In-Pragma Level-2 Operator



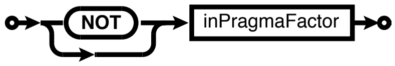
#17.1 Implementation Prefix



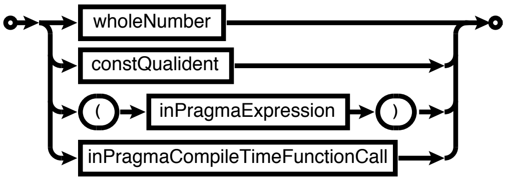
#20.1 In-Pragma Level-3 Operator



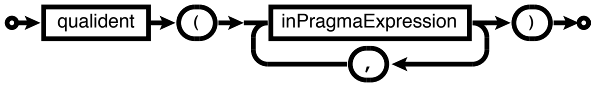
#21 In-Pragma Factor Or Negation (Evaluation Level 4)



#22 In-Pragma Factor (Evaluation Level 5)



#23 In-Pragma Compile-Time Function Call


qualident
(
inPragmaExpression
,
)