Preprocessed HAL/S

BNF-converter

February 25, 2023

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of HAL/S

Literals

NeqToken = ([“~”] | {“NOT”}[“ ”]∗)[“=”]
LeToken = ([“~”] | {“NOT”}[“ ”]∗)[“>”] | {“<=”}
GeToken = ([“~”] | {“NOT”}[“ ”]∗)[“<”] | {“>=”}
BitIdentifierToken = {“b_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
BitFunctionIdentifierToken = {“bf_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
CharFunctionIdentifierToken = {“cf_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
CharIdentifierToken = {“c_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
StructIdentifierToken = {“s_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
StructFunctionIdentifierToken = {“sf_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
LabelToken = {“l_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
EventToken = {“e_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
ArithFieldToken = {“a_”}⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
IdentifierToken = ⟨*letter*⟩((⟨*letter*⟩ | ⟨*digit*⟩ | {“_”}) ∗ (⟨*letter*⟩ | ⟨*digit*⟩))?
StringToken = [“’”](⟨*letter*⟩ | ⟨*digit*⟩ | {“””} | [“_+−]% $()[|*./&˜=<>#@,;:{}!?’”]) ∗ [“’”]
TextToken = [“"”](⟨*letter*⟩ | ⟨*digit*⟩ | [“_+−]% $()[|*./&˜=<>#@,;:{}!?”’]) ∗ [“"”]
LevelToken = [“123456789”] | [“12”]⟨*digit*⟩ | [“3”][“012”]
NumberToken = ⟨*digit*⟩+
CompoundToken = (⟨*digit*⟩ + ([“.”]⟨*digit*⟩∗)? | [“.”]⟨*digit*⟩+)([“EBH”][“−”]?⟨*digit*⟩+)∗

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in HAL/S are the following:

```
ABS           ABVAL       ACCESS
AFTER         ALIGNED     AND
ARCCOS        ARCCOSH     ARCSIN
ARCSINH       ARCTAN      ARCTAN2
ARCTANH       ARRAY       ASSIGN
AT            AUTOMATIC   BIN
BIT           BOOLEAN     BY
CALL          CANCEL      CASE
CAT           CEILING     CHAR
CHARACTER     CLOCKTIME   CLOSE
COLUMN        COMPOOL     CONSTANT
COS           COSH        DATE
DEC           DECLARE     DENSE
DEPENDENT     DET         DIV
DO            DOUBLE      ELSE
END           EQUATE      ERRGRP
ERRNUM        ERROR       EVENT
EVERY         EXCLUSIVE   EXIT
EXP           EXTERNAL    FALSE
FILE          FLOOR       FOR
FUNCTION      GO          HEX
IF            IGNORE      IN
INDEX         INITIAL     INTEGER
INVERSE       LATCHED     LENGTH
LINE          LJUST       LOCK
LOG           MATRIX      MAX
MIDVAL        MIN         MOD
NAME          NEXTIME     NONHAL
NOT           NULL        OCT
ODD           OFF         ON
OR            PAGE        PRIO
PRIORITY      PROCEDURE   PROD
PROGRAM       RANDOM      RANDOMG
READ          READALL     REENTRANT
REMAINDER     REMOTE      REPEAT
REPLACE       RESET       RETURN
RIGID         RJUST       ROUND
RUNTIME       SCALAR      SCHEDULE
SEND          SET         SHL
SHR           SIGN        SIGNAL
SIGNUM        SIN         SINGLE
SINH          SIZE        SKIP
SQRT          STATIC      STRUCTURE
SUBBIT        SUM         SYSTEM
TAB           TAN         TANH
TASK          TEMPORARY   TERMINATE
THEN          TO          TRACE
TRANSPOSE     TRIM        TRUE
TRUNCATE      UNIT        UNTIL
UPDATE        VECTOR      WAIT
WHILE         WRITE       XOR
initialized   typeof      typeofv
```

The symbols used in HAL/S are the following:

```
,   )   (
/   +   −
*   .   **T
**  #   [
]   {   }
$   @   ;
:   |   &
||  ~   <
=   >   %
```

Comments

There are no single-line comments in the grammar.
Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of HAL/S

Non-terminals are enclosed between ⟨ and ⟩. The symbols ::= (production), | (union) and ϵ (empty rule) belong to the BNF notation. All other symbols are terminals.

⟨*DECLARE-BODY*⟩ ::= ⟨*DECLARATION-LIST*⟩
| ⟨*ATTRIBUTES*⟩ `,` ⟨*DECLARATION-LIST*⟩

⟨*ATTRIBUTES*⟩ ::= ⟨*ARRAY-SPEC*⟩ ⟨*TYPE-AND-MINOR-ATTR*⟩
| ⟨*ARRAY-SPEC*⟩
| ⟨*TYPE-AND-MINOR-ATTR*⟩

⟨*DECLARATION*⟩ ::= ⟨*NAME-ID*⟩
| ⟨*NAME-ID*⟩ ⟨*ATTRIBUTES*⟩
| ⟨*LabelToken*⟩
| ⟨*LabelToken*⟩ ⟨*TYPE-AND-MINOR-ATTR*⟩
| ⟨*LabelToken*⟩ `PROCEDURE` ⟨*MINOR-ATTR-LIST*⟩
| ⟨*LabelToken*⟩ `PROCEDURE`
| ⟨*LabelToken*⟩ `FUNCTION` ⟨*TYPE-AND-MINOR-ATTR*⟩
| ⟨*LabelToken*⟩ `FUNCTION`
| ⟨*EventToken*⟩ `EVENT`
| ⟨*EventToken*⟩ `EVENT` ⟨*MINOR-ATTR-LIST*⟩
| ⟨*EventToken*⟩
| ⟨*EventToken*⟩ ⟨*MINOR-ATTR-LIST*⟩

⟨*ARRAY-SPEC*⟩ ::= ⟨*ARRAY-HEAD*⟩ ⟨*LITERAL-EXP-OR-STAR*⟩ `)`
| `FUNCTION`
| `PROCEDURE`
| `PROGRAM`
| `TASK`

⟨*TYPE-AND-MINOR-ATTR*⟩ ::= ⟨*TYPE-SPEC*⟩
| ⟨*TYPE-SPEC*⟩ ⟨*MINOR-ATTR-LIST*⟩
| ⟨*MINOR-ATTR-LIST*⟩

⟨*IDENTIFIER*⟩ ::= ⟨*IdentifierToken*⟩

⟨*SQ-DQ-NAME*⟩ ::= ⟨*DOUBLY-QUAL-NAME-HEAD*⟩ ⟨*LITERAL-EXP-OR-STAR*⟩ `)`
| ⟨*ARITH-CONV*⟩

⟨*DOUBLY-QUAL-NAME-HEAD*⟩ ::= `VECTOR (`
| `MATRIX (` ⟨*LITERAL-EXP-OR-STAR*⟩ `,`

⟨*ARITH-CONV*⟩ ::= `INTEGER`
| `SCALAR`
| `VECTOR`
| `MATRIX`

⟨*DECLARATION-LIST*⟩ ::= ⟨*DECLARATION*⟩
| ⟨*DCL-LIST-COMMA*⟩ ⟨*DECLARATION*⟩

⟨*NAME-ID*⟩ ::= ⟨*IDENTIFIER*⟩
| ⟨*IDENTIFIER*⟩ `NAME`
| ⟨*BIT-ID*⟩
| ⟨*CHAR-ID*⟩
| ⟨*BitFunctionIdentifierToken*⟩
| ⟨*CharFunctionIdentifierToken*⟩
| ⟨*StructIdentifierToken*⟩
| ⟨*StructFunctionIdentifierToken*⟩

⟨*ARITH-EXP*⟩ ::= ⟨*TERM*⟩
| ⟨*PLUS*⟩ ⟨*TERM*⟩
| ⟨*MINUS*⟩ ⟨*TERM*⟩
| ⟨*ARITH-EXP*⟩ ⟨*PLUS*⟩ ⟨*TERM*⟩
| ⟨*ARITH-EXP*⟩ ⟨*MINUS*⟩ ⟨*TERM*⟩

⟨*TERM*⟩ ::= ⟨*PRODUCT*⟩
| ⟨*PRODUCT*⟩ `/` ⟨*TERM*⟩

⟨*PLUS*⟩ ::= +

⟨*MINUS*⟩ ::= −

⟨*PRODUCT*⟩ ::= ⟨*FACTOR*⟩
| ⟨*FACTOR*⟩ `*` ⟨*PRODUCT*⟩
| ⟨*FACTOR*⟩ `.` ⟨*PRODUCT*⟩
| ⟨*FACTOR*⟩ ⟨*PRODUCT*⟩

⟨*FACTOR*⟩ ::= ⟨*PRIMARY*⟩
| ⟨*PRIMARY*⟩ ⟨*EXPONENTIATION*⟩ ⟨*FACTOR*⟩
| ⟨*PRIMARY*⟩ `**T`

⟨*EXPONENTIATION*⟩ ::= `**`

⟨*PRIMARY*⟩ ::= ⟨*ARITH-VAR*⟩
| ⟨*PRE-PRIMARY*⟩
| ⟨*MODIFIED-ARITH-FUNC*⟩
| ⟨*PRE-PRIMARY*⟩ ⟨*QUALIFIER*⟩

⟨*ARITH-VAR*⟩ ::= ⟨*ARITH-ID*⟩
| ⟨*ARITH-ID*⟩ ⟨*SUBSCRIPT*⟩
| `[` ⟨*ARITH-ID*⟩ `]`
| `[` ⟨*ARITH-ID*⟩ `]` ⟨*SUBSCRIPT*⟩
| `{` ⟨*QUAL-STRUCT*⟩ `}`
| `{` ⟨*QUAL-STRUCT*⟩ `}` ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*ARITH-ID*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*ARITH-ID*⟩ ⟨*SUBSCRIPT*⟩

⟨*PRE-PRIMARY*⟩ ::= `(` ⟨*ARITH-EXP*⟩ `)`
| ⟨*NUMBER*⟩
| ⟨*COMPOUND-NUMBER*⟩
| ⟨*ARITH-FUNC*⟩ `(` ⟨*CALL-LIST*⟩ `)`
| `typeof (` ⟨*CALL-LIST*⟩ `)`
| `typeofv (` ⟨*CALL-LIST*⟩ `)`
| ⟨*SHAPING-HEAD*⟩ `)`
| ⟨*SHAPING-HEAD*⟩ `, * )`
| ⟨*LabelToken*⟩ `(` ⟨*CALL-LIST*⟩ `)`

⟨*NUMBER*⟩ ::= ⟨*SIMPLE-NUMBER*⟩
| ⟨*LEVEL*⟩

⟨*LEVEL*⟩ ::= ⟨*LevelToken*⟩

⟨*COMPOUND-NUMBER*⟩ ::= ⟨*CompoundToken*⟩

⟨*SIMPLE-NUMBER*⟩ ::= ⟨*NumberToken*⟩

⟨*MODIFIED-ARITH-FUNC*⟩ ::= ⟨*NO-ARG-ARITH-FUNC*⟩
| ⟨*NO-ARG-ARITH-FUNC*⟩ ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*NO-ARG-ARITH-FUNC*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*NO-ARG-ARITH-FUNC*⟩ ⟨*SUBSCRIPT*⟩

⟨*SHAPING-HEAD*⟩ ::= `INTEGER (` ⟨*REPEATED-CONSTANT*⟩
| `SCALAR (` ⟨*REPEATED-CONSTANT*⟩
| `VECTOR (` ⟨*REPEATED-CONSTANT*⟩
| `MATRIX (` ⟨*REPEATED-CONSTANT*⟩
| `INTEGER` ⟨*SUBSCRIPT*⟩ `(` ⟨*REPEATED-CONSTANT*⟩
| `SCALAR` ⟨*SUBSCRIPT*⟩ `(` ⟨*REPEATED-CONSTANT*⟩
| `VECTOR` ⟨*SUBSCRIPT*⟩ `(` ⟨*REPEATED-CONSTANT*⟩
| `MATRIX` ⟨*SUBSCRIPT*⟩ `(` ⟨*REPEATED-CONSTANT*⟩
| ⟨*SHAPING-HEAD*⟩ `,` ⟨*REPEATED-CONSTANT*⟩

⟨*CALL-LIST*⟩ ::= ⟨*LIST-EXP*⟩
| ⟨*CALL-LIST*⟩ `,` ⟨*LIST-EXP*⟩

⟨*LIST-EXP*⟩ ::= ⟨*EXPRESSION*⟩
| ⟨*ARITH-EXP*⟩ `#` ⟨*EXPRESSION*⟩
| ⟨*QUAL-STRUCT*⟩

⟨*EXPRESSION*⟩ ::= ⟨*ARITH-EXP*⟩
| ⟨*BIT-EXP*⟩
| ⟨*CHAR-EXP*⟩
| ⟨*NAME-EXP*⟩
| ⟨*STRUCTURE-EXP*⟩

⟨*ARITH-ID*⟩ ::= ⟨*IDENTIFIER*⟩
| ⟨*ArithFieldToken*⟩

⟨*NO-ARG-ARITH-FUNC*⟩ ::= `CLOCKTIME`
| `DATE`
| `ERRGRP`
| `ERRNUM`
| `PRIO`
| `RANDOM`
| `RANDOMG`
| `RUNTIME`

⟨*ARITH-FUNC*⟩ ::= NEXTIME
| ABS
| CEILING
| DIV
| FLOOR
| MIDVAL
| MOD
| ODD
| REMAINDER
| ROUND
| SIGN
| SIGNUM
| TRUNCATE
| ARCCOS
| ARCCOSH
| ARCSIN
| ARCSINH
| ARCTAN2
| ARCTAN
| ARCTANH
| COS
| COSH
| EXP
| LOG
| SIN
| SINH
| SQRT
| TAN
| TANH
| SHL
| SHR
| ABVAL
| DET
| TRACE
| UNIT
| INDEX
| LENGTH
| INVERSE
| TRANSPOSE
| PROD
| SUM
| SIZE
| MAX
| MIN

⟨*SUBSCRIPT*⟩ ::= ⟨*SUB-HEAD*⟩)
| ⟨*QUALIFIER*⟩
| $ ⟨*NUMBER*⟩
| $ ⟨*ARITH-VAR*⟩

⟨*QUALIFIER*⟩ ::= $ (@ ⟨*PREC-SPEC*⟩)
| $ (⟨*SCALE-HEAD*⟩ ⟨*ARITH-EXP*⟩)
| $ (@ ⟨*PREC-SPEC*⟩ , ⟨*SCALE-HEAD*⟩ ⟨*ARITH-EXP*⟩)
| $ (@ ⟨*RADIX*⟩)

⟨*SCALE-HEAD*⟩ ::= `@`
| `@ @`

⟨*PREC-SPEC*⟩ ::= `SINGLE`
| `DOUBLE`

⟨*SUB-START*⟩ ::= `$ (`
| `$ ( @` ⟨*PREC-SPEC*⟩ `,`
| ⟨*SUB-HEAD*⟩ `;`
| ⟨*SUB-HEAD*⟩ `:`
| ⟨*SUB-HEAD*⟩ `,`

⟨*SUB-HEAD*⟩ ::= ⟨*SUB-START*⟩
| ⟨*SUB-START*⟩ ⟨*SUB*⟩

⟨*SUB*⟩ ::= ⟨*SUB-EXP*⟩
| `*`
| ⟨*SUB-RUN-HEAD*⟩ ⟨*SUB-EXP*⟩
| ⟨*ARITH-EXP*⟩ `AT` ⟨*SUB-EXP*⟩

⟨*SUB-RUN-HEAD*⟩ ::= ⟨*SUB-EXP*⟩ `TO`

⟨*SUB-EXP*⟩ ::= ⟨*ARITH-EXP*⟩
| ⟨*POUND-EXPRESSION*⟩

⟨*POUND-EXPRESSION*⟩ ::= `#`
| ⟨*POUND-EXPRESSION*⟩ ⟨*PLUS*⟩ ⟨*TERM*⟩
| ⟨*POUND-EXPRESSION*⟩ ⟨*MINUS*⟩ ⟨*TERM*⟩

⟨*BIT-EXP*⟩ ::= ⟨*BIT-FACTOR*⟩
| ⟨*BIT-EXP*⟩ ⟨*OR*⟩ ⟨*BIT-FACTOR*⟩

⟨*BIT-FACTOR*⟩ ::= ⟨*BIT-CAT*⟩
| ⟨*BIT-FACTOR*⟩ ⟨*AND*⟩ ⟨*BIT-CAT*⟩

⟨*BIT-CAT*⟩ ::= ⟨*BIT-PRIM*⟩
| ⟨*BIT-CAT*⟩ ⟨*CAT*⟩ ⟨*BIT-PRIM*⟩
| ⟨*NOT*⟩ ⟨*BIT-PRIM*⟩
| ⟨*BIT-CAT*⟩ ⟨*CAT*⟩ ⟨*NOT*⟩ ⟨*BIT-PRIM*⟩

⟨*OR*⟩ ::= ⟨*CHAR-VERTICAL-BAR*⟩
| `OR`

⟨*CHAR-VERTICAL-BAR*⟩ ::= |

⟨*AND*⟩ ::= `&`
| `AND`

⟨*BIT-PRIM*⟩ ::= ⟨*BIT-VAR*⟩
| ⟨*LABEL-VAR*⟩
| ⟨*EVENT-VAR*⟩
| ⟨*BIT-CONST*⟩
| `(` ⟨*BIT-EXP*⟩ `)`
| ⟨*SUBBIT-HEAD*⟩ ⟨*EXPRESSION*⟩ `)`
| ⟨*BIT-FUNC-HEAD*⟩ `(` ⟨*CALL-LIST*⟩ `)`
| `initialized (` ⟨*CALL-LIST*⟩ `)`
| `[` ⟨*BIT-VAR*⟩ `]`
| `[` ⟨*BIT-VAR*⟩ `]` ⟨*SUBSCRIPT*⟩
| `{` ⟨*BIT-VAR*⟩ `}`
| `{` ⟨*BIT-VAR*⟩ `}` ⟨*SUBSCRIPT*⟩

⟨*CAT*⟩ ::= ||
| `CAT`

⟨*NOT*⟩ ::= `NOT`
| `~`

⟨*BIT-VAR*⟩ ::= ⟨*BIT-ID*⟩
| ⟨*BIT-ID*⟩ ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*BIT-ID*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*BIT-ID*⟩ ⟨*SUBSCRIPT*⟩

⟨*LABEL-VAR*⟩ ::= ⟨*LABEL*⟩
| ⟨*LABEL*⟩ ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*LABEL*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*LABEL*⟩ ⟨*SUBSCRIPT*⟩

⟨*EVENT-VAR*⟩ ::= ⟨*EVENT*⟩
| ⟨*EVENT*⟩ ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*EVENT*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*EVENT*⟩ ⟨*SUBSCRIPT*⟩

⟨*BIT-CONST-HEAD*⟩ ::= ⟨*RADIX*⟩
| ⟨*RADIX*⟩ `(` ⟨*NUMBER*⟩ `)`

⟨*BIT-CONST*⟩ ::= ⟨*BIT-CONST-HEAD*⟩ ⟨*CHAR-STRING*⟩
| `TRUE`
| `FALSE`
| `ON`
| `OFF`

⟨*RADIX*⟩ ::= `HEX`
| `OCT`
| `BIN`
| `DEC`

⟨*CHAR-STRING*⟩ ::= ⟨*StringToken*⟩

⟨*SUBBIT-HEAD*⟩ ::= ⟨*SUBBIT-KEY*⟩ `(`
| ⟨*SUBBIT-KEY*⟩ ⟨*SUBSCRIPT*⟩ `(`

⟨*SUBBIT-KEY*⟩ ::= `SUBBIT`

⟨*BIT-FUNC-HEAD*⟩ ::= ⟨*BIT-FUNC*⟩
| `BIT`
| `BIT` ⟨*SUB-OR-QUALIFIER*⟩

⟨*BIT-ID*⟩ ::= ⟨*BitIdentifierToken*⟩

⟨*LABEL*⟩ ::= ⟨*LabelToken*⟩
| ⟨*BitFunctionIdentifierToken*⟩
| ⟨*CharFunctionIdentifierToken*⟩
| ⟨*StructFunctionIdentifierToken*⟩

⟨*BIT-FUNC*⟩ ::= `XOR`
| ⟨*BitFunctionIdentifierToken*⟩

⟨*EVENT*⟩ ::= ⟨*EventToken*⟩

⟨*SUB-OR-QUALIFIER*⟩ ::= ⟨*SUBSCRIPT*⟩
| ⟨*BIT-QUALIFIER*⟩

⟨*BIT-QUALIFIER*⟩ ::= < `$ ( @` ⟨*RADIX*⟩ `)`

⟨*CHAR-EXP*⟩ ::= ⟨*CHAR-PRIM*⟩
| ⟨*CHAR-EXP*⟩ ⟨*CAT*⟩ ⟨*CHAR-PRIM*⟩
| ⟨*CHAR-EXP*⟩ ⟨*CAT*⟩ ⟨*ARITH-EXP*⟩
| ⟨*ARITH-EXP*⟩ ⟨*CAT*⟩ ⟨*ARITH-EXP*⟩
| ⟨*ARITH-EXP*⟩ ⟨*CAT*⟩ ⟨*CHAR-PRIM*⟩

⟨*CHAR-PRIM*⟩ ::= ⟨*CHAR-VAR*⟩
| ⟨*CHAR-CONST*⟩
| ⟨*CHAR-FUNC-HEAD*⟩ `(` ⟨*CALL-LIST*⟩ `)`
| `(` ⟨*CHAR-EXP*⟩ `)`

⟨*CHAR-FUNC-HEAD*⟩ ::= ⟨*CHAR-FUNC*⟩
| `CHARACTER` ⟨*SUB-OR-QUALIFIER*⟩

⟨*CHAR-VAR*⟩ ::= ⟨*CHAR-ID*⟩
| ⟨*CHAR-ID*⟩ ⟨*SUBSCRIPT*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*CHAR-ID*⟩
| ⟨*QUAL-STRUCT*⟩ `.` ⟨*CHAR-ID*⟩ ⟨*SUBSCRIPT*⟩

⟨*CHAR-CONST*⟩ ::= ⟨*CHAR-STRING*⟩
| `CHAR (` ⟨*NUMBER*⟩ `)` ⟨*CHAR-STRING*⟩

⟨*CHAR-FUNC*⟩ ::= `LJUST`
| `RJUST`
| `TRIM`
| ⟨*CharFunctionIdentifierToken*⟩
| `CHARACTER`

⟨*CHAR-ID*⟩ ::= ⟨*CharIdentifierToken*⟩

⟨*NAME-EXP*⟩ ::= ⟨*NAME-KEY*⟩ `(` ⟨*NAME-VAR*⟩ `)`
| `NULL`
| ⟨*NAME-KEY*⟩ `( NULL )`

⟨*NAME-KEY*⟩ ::= `NAME`

⟨*NAME-VAR*⟩ ::= ⟨*VARIABLE*⟩
| ⟨*MODIFIED-ARITH-FUNC*⟩
| ⟨*LABEL-VAR*⟩

⟨*VARIABLE*⟩ ::= ⟨*ARITH-VAR*⟩
| ⟨*BIT-VAR*⟩
| ⟨*SUBBIT-HEAD*⟩ ⟨*VARIABLE*⟩)
| ⟨*CHAR-VAR*⟩
| ⟨*NAME-KEY*⟩ (⟨*NAME-VAR*⟩)
| ⟨*EVENT-VAR*⟩
| ⟨*STRUCTURE-VAR*⟩

⟨*STRUCTURE-EXP*⟩ ::= ⟨*STRUCTURE-VAR*⟩
| ⟨*STRUCT-FUNC-HEAD*⟩ (⟨*CALL-LIST*⟩)
| ⟨*STRUC-INLINE-DEF*⟩ ⟨*CLOSING*⟩ ;
| ⟨*STRUC-INLINE-DEF*⟩ ⟨*BLOCK-BODY*⟩ ⟨*CLOSING*⟩ ;

⟨*STRUCT-FUNC-HEAD*⟩ ::= ⟨*STRUCT-FUNC*⟩

⟨*STRUCTURE-VAR*⟩ ::= ⟨*QUAL-STRUCT*⟩ ⟨*SUBSCRIPT*⟩

⟨*STRUCT-FUNC*⟩ ::= ⟨*StructFunctionIdentifierToken*⟩

⟨*QUAL-STRUCT*⟩ ::= ⟨*STRUCTURE-ID*⟩
| ⟨*QUAL-STRUCT*⟩ . ⟨*STRUCTURE-ID*⟩

⟨*STRUCTURE-ID*⟩ ::= ⟨*StructIdentifierToken*⟩

⟨*ASSIGNMENT*⟩ ::= ⟨*VARIABLE*⟩ ⟨*EQUALS*⟩ ⟨*EXPRESSION*⟩
| ⟨*VARIABLE*⟩ , ⟨*ASSIGNMENT*⟩
| ⟨*QUAL-STRUCT*⟩ ⟨*EQUALS*⟩ ⟨*EXPRESSION*⟩
| ⟨*QUAL-STRUCT*⟩ , ⟨*ASSIGNMENT*⟩

⟨*EQUALS*⟩ ::= =

⟨*STATEMENT*⟩ ::= ⟨*BASIC-STATEMENT*⟩
| ⟨*OTHER-STATEMENT*⟩
| ⟨*INLINE-DEFINITION*⟩

⟨*BASIC-STATEMENT*⟩ ::= ⟨*ASSIGNMENT*⟩ ;
| ⟨*LABEL-DEFINITION*⟩ ⟨*BASIC-STATEMENT*⟩
| `EXIT` ;
| `EXIT` ⟨*LABEL*⟩ ;
| `REPEAT` ;
| `REPEAT` ⟨*LABEL*⟩ ;
| `GO TO` ⟨*LABEL*⟩ ;
| ;
| ⟨*CALL-KEY*⟩ ;
| ⟨*CALL-KEY*⟩ `(` ⟨*CALL-LIST*⟩ `)` ;
| ⟨*CALL-KEY*⟩ ⟨*ASSIGN*⟩ `(` ⟨*CALL-ASSIGN-LIST*⟩ `)` ;
| ⟨*CALL-KEY*⟩ `(` ⟨*CALL-LIST*⟩ `)` ⟨*ASSIGN*⟩ `(` ⟨*CALL-ASSIGN-LIST*⟩ `)` ;
| `RETURN` ;
| `RETURN` ⟨*EXPRESSION*⟩ ;
| ⟨*DO-GROUP-HEAD*⟩ ⟨*ENDING*⟩ ;
| ⟨*READ-KEY*⟩ ;
| ⟨*READ-PHRASE*⟩ ;
| ⟨*WRITE-KEY*⟩ ;
| ⟨*WRITE-PHRASE*⟩ ;
| ⟨*FILE-EXP*⟩ ⟨*EQUALS*⟩ ⟨*EXPRESSION*⟩ ;
| ⟨*VARIABLE*⟩ ⟨*EQUALS*⟩ ⟨*FILE-EXP*⟩ ;
| ⟨*FILE-EXP*⟩ ⟨*EQUALS*⟩ ⟨*QUAL-STRUCT*⟩ ;
| ⟨*WAIT-KEY*⟩ `FOR DEPENDENT` ;
| ⟨*WAIT-KEY*⟩ ⟨*ARITH-EXP*⟩ ;
| ⟨*WAIT-KEY*⟩ `UNTIL` ⟨*ARITH-EXP*⟩ ;
| ⟨*WAIT-KEY*⟩ `FOR` ⟨*BIT-EXP*⟩ ;
| ⟨*TERMINATOR*⟩ ;
| ⟨*TERMINATOR*⟩ ⟨*TERMINATE-LIST*⟩ ;
| `UPDATE PRIORITY TO` ⟨*ARITH-EXP*⟩ ;
| `UPDATE PRIORITY` ⟨*LABEL-VAR*⟩ `TO` ⟨*ARITH-EXP*⟩ ;
| ⟨*SCHEDULE-PHRASE*⟩ ;
| ⟨*SCHEDULE-PHRASE*⟩ ⟨*SCHEDULE-CONTROL*⟩ ;
| ⟨*SIGNAL-CLAUSE*⟩ ;
| `SEND ERROR` ⟨*SUBSCRIPT*⟩ ;
| `SEND ERROR` ;
| ⟨*ON-CLAUSE*⟩ ;
| ⟨*ON-CLAUSE*⟩ `AND` ⟨*SIGNAL-CLAUSE*⟩ ;
| `OFF ERROR` ⟨*SUBSCRIPT*⟩ ;
| `OFF ERROR` ;
| ⟨*PERCENT-MACRO-NAME*⟩ ;
| ⟨*PERCENT-MACRO-HEAD*⟩ ⟨*PERCENT-MACRO-ARG*⟩ `)` ;

⟨*OTHER-STATEMENT*⟩ ::= ⟨*IF-STATEMENT*⟩
| ⟨*ON-PHRASE*⟩ ⟨*STATEMENT*⟩
| ⟨*LABEL-DEFINITION*⟩ ⟨*OTHER-STATEMENT*⟩

⟨*IF-STATEMENT*⟩ ::= ⟨*IF-CLAUSE*⟩ ⟨*STATEMENT*⟩
| ⟨*TRUE-PART*⟩ ⟨*STATEMENT*⟩

⟨*IF-CLAUSE*⟩ ::= ⟨*IF*⟩ ⟨*RELATIONAL-EXP*⟩ ⟨*THEN*⟩
| ⟨*IF*⟩ ⟨*BIT-EXP*⟩ ⟨*THEN*⟩

⟨*TRUE-PART*⟩ ::= ⟨*IF-CLAUSE*⟩ ⟨*BASIC-STATEMENT*⟩ `ELSE`

⟨*IF*⟩ ::= `IF`

⟨*THEN*⟩ ::= `THEN`

⟨*RELATIONAL-EXP*⟩ ::= ⟨*RELATIONAL-FACTOR*⟩
| ⟨*RELATIONAL-EXP*⟩ ⟨*OR*⟩ ⟨*RELATIONAL-FACTOR*⟩

⟨*RELATIONAL-FACTOR*⟩ ::= ⟨*REL-PRIM*⟩
| ⟨*RELATIONAL-FACTOR*⟩ ⟨*AND*⟩ ⟨*REL-PRIM*⟩

⟨*REL-PRIM*⟩ ::= `(` ⟨*RELATIONAL-EXP*⟩ `)`
| ⟨*NOT*⟩ `(` ⟨*RELATIONAL-EXP*⟩ `)`
| ⟨*COMPARISON*⟩

⟨*COMPARISON*⟩ ::= ⟨*ARITH-EXP*⟩ ⟨*EQUALS*⟩ ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ ⟨*EQUALS*⟩ ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ ⟨*EQUALS*⟩ ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ ⟨*EQUALS*⟩ ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ ⟨*EQUALS*⟩ ⟨*NAME-EXP*⟩
| ⟨*ARITH-EXP*⟩ ⟨*NeqToken*⟩ ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ ⟨*NeqToken*⟩ ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ ⟨*NeqToken*⟩ ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ ⟨*NeqToken*⟩ ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ ⟨*NeqToken*⟩ ⟨*NAME-EXP*⟩
| ⟨*ARITH-EXP*⟩ < ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ < ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ < ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ < ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ < ⟨*NAME-EXP*⟩
| ⟨*ARITH-EXP*⟩ > ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ > ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ > ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ > ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ > ⟨*NAME-EXP*⟩
| ⟨*ARITH-EXP*⟩ ⟨*LeToken*⟩ ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ ⟨*LeToken*⟩ ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ ⟨*LeToken*⟩ ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ ⟨*LeToken*⟩ ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ ⟨*LeToken*⟩ ⟨*NAME-EXP*⟩
| ⟨*ARITH-EXP*⟩ ⟨*GeToken*⟩ ⟨*ARITH-EXP*⟩
| ⟨*CHAR-EXP*⟩ ⟨*GeToken*⟩ ⟨*CHAR-EXP*⟩
| ⟨*BIT-CAT*⟩ ⟨*GeToken*⟩ ⟨*BIT-CAT*⟩
| ⟨*STRUCTURE-EXP*⟩ ⟨*GeToken*⟩ ⟨*STRUCTURE-EXP*⟩
| ⟨*NAME-EXP*⟩ ⟨*GeToken*⟩ ⟨*NAME-EXP*⟩

⟨*ANY-STATEMENT*⟩ ::= ⟨*STATEMENT*⟩
| ⟨*BLOCK-DEFINITION*⟩

⟨*ON-PHRASE*⟩ ::= `ON ERROR` ⟨*SUBSCRIPT*⟩
| `ON ERROR`

⟨*ON-CLAUSE*⟩ ::= `ON ERROR` ⟨*SUBSCRIPT*⟩ `SYSTEM`
| `ON ERROR` ⟨*SUBSCRIPT*⟩ `IGNORE`
| `ON ERROR SYSTEM`
| `ON ERROR IGNORE`

⟨*LABEL-DEFINITION*⟩ ::= ⟨*LABEL*⟩ `:`

⟨*CALL-KEY*⟩ ::= `CALL` ⟨*LABEL-VAR*⟩

⟨*ASSIGN*⟩ ::= `ASSIGN`

⟨*CALL-ASSIGN-LIST*⟩ ::= ⟨*VARIABLE*⟩
| ⟨*CALL-ASSIGN-LIST*⟩ `,` ⟨*VARIABLE*⟩
| ⟨*QUAL-STRUCT*⟩
| ⟨*CALL-ASSIGN-LIST*⟩ `,` ⟨*QUAL-STRUCT*⟩

⟨*DO-GROUP-HEAD*⟩ ::= `DO ;`
| `DO` ⟨*FOR-LIST*⟩ `;`
| `DO` ⟨*FOR-LIST*⟩ ⟨*WHILE-CLAUSE*⟩ `;`
| `DO` ⟨*WHILE-CLAUSE*⟩ `;`
| `DO CASE` ⟨*ARITH-EXP*⟩ `;`
| ⟨*CASE-ELSE*⟩ ⟨*STATEMENT*⟩
| ⟨*DO-GROUP-HEAD*⟩ ⟨*ANY-STATEMENT*⟩
| ⟨*DO-GROUP-HEAD*⟩ ⟨*TEMPORARY-STMT*⟩

⟨*ENDING*⟩ ::= `END`
| `END` ⟨*LABEL*⟩
| ⟨*LABEL-DEFINITION*⟩ ⟨*ENDING*⟩

⟨*READ-KEY*⟩ ::= `READ (` ⟨*NUMBER*⟩ `)`
| `READALL (` ⟨*NUMBER*⟩ `)`

⟨*WRITE-KEY*⟩ ::= `WRITE (` ⟨*NUMBER*⟩ `)`

⟨*READ-PHRASE*⟩ ::= ⟨*READ-KEY*⟩ ⟨*READ-ARG*⟩
| ⟨*READ-PHRASE*⟩ `,` ⟨*READ-ARG*⟩

⟨*WRITE-PHRASE*⟩ ::= ⟨*WRITE-KEY*⟩ ⟨*WRITE-ARG*⟩
| ⟨*WRITE-PHRASE*⟩ `,` ⟨*WRITE-ARG*⟩

⟨*READ-ARG*⟩ ::= ⟨*VARIABLE*⟩
| ⟨*IO-CONTROL*⟩

⟨*WRITE-ARG*⟩ ::= ⟨*EXPRESSION*⟩
| ⟨*IO-CONTROL*⟩
| ⟨*StructIdentifierToken*⟩

⟨*FILE-EXP*⟩ ::= ⟨*FILE-HEAD*⟩ `,` ⟨*ARITH-EXP*⟩ `)`

⟨*FILE-HEAD*⟩ ::= `FILE (` ⟨*NUMBER*⟩

⟨*IO-CONTROL*⟩ ::= `SKIP (` ⟨*ARITH-EXP*⟩ `)`
| `TAB (` ⟨*ARITH-EXP*⟩ `)`
| `COLUMN (` ⟨*ARITH-EXP*⟩ `)`
| `LINE (` ⟨*ARITH-EXP*⟩ `)`
| `PAGE (` ⟨*ARITH-EXP*⟩ `)`

⟨*WAIT-KEY*⟩ ::= `WAIT`

⟨*TERMINATOR*⟩ ::= `TERMINATE`
| `CANCEL`

⟨*TERMINATE-LIST*⟩ ::= ⟨*LABEL-VAR*⟩
| ⟨*TERMINATE-LIST*⟩ `,` ⟨*LABEL-VAR*⟩

⟨*SCHEDULE-HEAD*⟩ ::= SCHEDULE ⟨*LABEL-VAR*⟩
| ⟨*SCHEDULE-HEAD*⟩ AT ⟨*ARITH-EXP*⟩
| ⟨*SCHEDULE-HEAD*⟩ IN ⟨*ARITH-EXP*⟩
| ⟨*SCHEDULE-HEAD*⟩ ON ⟨*BIT-EXP*⟩

⟨*SCHEDULE-PHRASE*⟩ ::= ⟨*SCHEDULE-HEAD*⟩
| ⟨*SCHEDULE-HEAD*⟩ PRIORITY (⟨*ARITH-EXP*⟩)
| ⟨*SCHEDULE-PHRASE*⟩ DEPENDENT

⟨*SCHEDULE-CONTROL*⟩ ::= ⟨*STOPPING*⟩
| ⟨*TIMING*⟩
| ⟨*TIMING*⟩ ⟨*STOPPING*⟩

⟨*TIMING*⟩ ::= ⟨*REPEAT*⟩ EVERY ⟨*ARITH-EXP*⟩
| ⟨*REPEAT*⟩ AFTER ⟨*ARITH-EXP*⟩
| ⟨*REPEAT*⟩

⟨*REPEAT*⟩ ::= , REPEAT

⟨*STOPPING*⟩ ::= ⟨*WHILE-KEY*⟩ ⟨*ARITH-EXP*⟩
| ⟨*WHILE-KEY*⟩ ⟨*BIT-EXP*⟩

⟨*SIGNAL-CLAUSE*⟩ ::= SET ⟨*EVENT-VAR*⟩
| RESET ⟨*EVENT-VAR*⟩
| SIGNAL ⟨*EVENT-VAR*⟩

⟨*PERCENT-MACRO-NAME*⟩ ::= % ⟨*IDENTIFIER*⟩

⟨*PERCENT-MACRO-HEAD*⟩ ::= ⟨*PERCENT-MACRO-NAME*⟩ (
| ⟨*PERCENT-MACRO-HEAD*⟩ ⟨*PERCENT-MACRO-ARG*⟩ ,

⟨*PERCENT-MACRO-ARG*⟩ ::= ⟨*NAME-VAR*⟩
| ⟨*CONSTANT*⟩

⟨*CASE-ELSE*⟩ ::= DO CASE ⟨*ARITH-EXP*⟩ ; ELSE

⟨*WHILE-KEY*⟩ ::= WHILE
| UNTIL

⟨*WHILE-CLAUSE*⟩ ::= ⟨*WHILE-KEY*⟩ ⟨*BIT-EXP*⟩
| ⟨*WHILE-KEY*⟩ ⟨*RELATIONAL-EXP*⟩

⟨*FOR-LIST*⟩ ::= ⟨*FOR-KEY*⟩ ⟨*ARITH-EXP*⟩ ⟨*ITERATION-CONTROL*⟩
| ⟨*FOR-KEY*⟩ ⟨*ITERATION-BODY*⟩

⟨*ITERATION-BODY*⟩ ::= ⟨*ARITH-EXP*⟩
| ⟨*ITERATION-BODY*⟩ , ⟨*ARITH-EXP*⟩

⟨*ITERATION-CONTROL*⟩ ::= TO ⟨*ARITH-EXP*⟩
| TO ⟨*ARITH-EXP*⟩ BY ⟨*ARITH-EXP*⟩

⟨*FOR-KEY*⟩ ::= FOR ⟨*ARITH-VAR*⟩ ⟨*EQUALS*⟩
| FOR TEMPORARY ⟨*IDENTIFIER*⟩ =

⟨*TEMPORARY-STMT*⟩ ::= TEMPORARY ⟨*DECLARE-BODY*⟩ ;

⟨*CONSTANT*⟩ ::= ⟨*NUMBER*⟩
| ⟨*COMPOUND-NUMBER*⟩
| ⟨*BIT-CONST*⟩
| ⟨*CHAR-CONST*⟩

⟨*ARRAY-HEAD*⟩ ::= `ARRAY (`
| ⟨*ARRAY-HEAD*⟩ ⟨*LITERAL-EXP-OR-STAR*⟩ `,`

⟨*MINOR-ATTR-LIST*⟩ ::= ⟨*MINOR-ATTRIBUTE*⟩
| ⟨*MINOR-ATTR-LIST*⟩ ⟨*MINOR-ATTRIBUTE*⟩

⟨*MINOR-ATTRIBUTE*⟩ ::= `STATIC`
| `AUTOMATIC`
| `DENSE`
| `ALIGNED`
| `ACCESS`
| `LOCK (` ⟨*LITERAL-EXP-OR-STAR*⟩ `)`
| `REMOTE`
| `RIGID`
| ⟨*INIT-OR-CONST-HEAD*⟩ ⟨*REPEATED-CONSTANT*⟩ `)`
| ⟨*INIT-OR-CONST-HEAD*⟩ `* )`
| `LATCHED`
| `NONHAL (` ⟨*LEVEL*⟩ `)`

⟨*INIT-OR-CONST-HEAD*⟩ ::= `INITIAL (`
| `CONSTANT (`
| ⟨*INIT-OR-CONST-HEAD*⟩ ⟨*REPEATED-CONSTANT*⟩ `,`

⟨*REPEATED-CONSTANT*⟩ ::= ⟨*EXPRESSION*⟩
| ⟨*REPEAT-HEAD*⟩ ⟨*VARIABLE*⟩
| ⟨*REPEAT-HEAD*⟩ ⟨*CONSTANT*⟩
| ⟨*NESTED-REPEAT-HEAD*⟩ ⟨*REPEATED-CONSTANT*⟩ `)`
| ⟨*REPEAT-HEAD*⟩

⟨*REPEAT-HEAD*⟩ ::= ⟨*ARITH-EXP*⟩ `#`

⟨*NESTED-REPEAT-HEAD*⟩ ::= ⟨*REPEAT-HEAD*⟩ `(`
| ⟨*NESTED-REPEAT-HEAD*⟩ ⟨*REPEATED-CONSTANT*⟩ `,`

⟨*DCL-LIST-COMMA*⟩ ::= ⟨*DECLARATION-LIST*⟩ `,`

⟨*LITERAL-EXP-OR-STAR*⟩ ::= ⟨*ARITH-EXP*⟩
| `*`

⟨*TYPE-SPEC*⟩ ::= ⟨*STRUCT-SPEC*⟩
| ⟨*BIT-SPEC*⟩
| ⟨*CHAR-SPEC*⟩
| ⟨*ARITH-SPEC*⟩
| `EVENT`

⟨*BIT-SPEC*⟩ ::= `BOOLEAN`
| `BIT (` ⟨*LITERAL-EXP-OR-STAR*⟩ `)`

⟨*CHAR-SPEC*⟩ ::= `CHARACTER (` ⟨*LITERAL-EXP-OR-STAR*⟩ `)`

⟨*STRUCT-SPEC*⟩ ::= ⟨*STRUCT-TEMPLATE*⟩ ⟨*STRUCT-SPEC-BODY*⟩

⟨*STRUCT-SPEC-BODY*⟩ ::= − `STRUCTURE`
| ⟨*STRUCT-SPEC-HEAD*⟩ ⟨*LITERAL-EXP-OR-STAR*⟩ `)`

⟨*STRUCT-TEMPLATE*⟩ ::= ⟨*STRUCTURE-ID*⟩

⟨*STRUCT-SPEC-HEAD*⟩ ::= − `STRUCTURE (`

⟨*ARITH-SPEC*⟩ ::= ⟨*PREC-SPEC*⟩
| ⟨*SQ-DQ-NAME*⟩
| ⟨*SQ-DQ-NAME*⟩ ⟨*PREC-SPEC*⟩

⟨*COMPILATION*⟩ ::= ⟨*ANY-STATEMENT*⟩
| ⟨*COMPILATION*⟩ ⟨*ANY-STATEMENT*⟩
| ⟨*DECLARE-STATEMENT*⟩
| ⟨*COMPILATION*⟩ ⟨*DECLARE-STATEMENT*⟩
| ⟨*STRUCTURE-STMT*⟩
| ⟨*COMPILATION*⟩ ⟨*STRUCTURE-STMT*⟩
| ⟨*REPLACE-STMT*⟩ `;`
| ⟨*COMPILATION*⟩ ⟨*REPLACE-STMT*⟩ `;`
| ⟨*INIT-OR-CONST-HEAD*⟩ ⟨*EXPRESSION*⟩ `)`

⟨*BLOCK-DEFINITION*⟩ ::= ⟨*BLOCK-STMT*⟩ ⟨*CLOSING*⟩ `;`
| ⟨*BLOCK-STMT*⟩ ⟨*BLOCK-BODY*⟩ ⟨*CLOSING*⟩ `;`

⟨*BLOCK-STMT*⟩ ::= ⟨*BLOCK-STMT-TOP*⟩ `;`

⟨*BLOCK-STMT-TOP*⟩ ::= ⟨*BLOCK-STMT-TOP*⟩ `ACCESS`
| ⟨*BLOCK-STMT-TOP*⟩ `RIGID`
| ⟨*BLOCK-STMT-HEAD*⟩
| ⟨*BLOCK-STMT-HEAD*⟩ `EXCLUSIVE`
| ⟨*BLOCK-STMT-HEAD*⟩ `REENTRANT`

⟨*BLOCK-STMT-HEAD*⟩ ::= ⟨*LABEL-EXTERNAL*⟩ `PROGRAM`
| ⟨*LABEL-EXTERNAL*⟩ `COMPOOL`
| ⟨*LABEL-DEFINITION*⟩ `TASK`
| ⟨*LABEL-DEFINITION*⟩ `UPDATE`
| `UPDATE`
| ⟨*FUNCTION-NAME*⟩
| ⟨*FUNCTION-NAME*⟩ ⟨*FUNC-STMT-BODY*⟩
| ⟨*PROCEDURE-NAME*⟩
| ⟨*PROCEDURE-NAME*⟩ ⟨*PROC-STMT-BODY*⟩

⟨*LABEL-EXTERNAL*⟩ ::= ⟨*LABEL-DEFINITION*⟩
| ⟨*LABEL-DEFINITION*⟩ `EXTERNAL`

⟨*CLOSING*⟩ ::= `CLOSE`
| `CLOSE` ⟨*LABEL*⟩
| ⟨*LABEL-DEFINITION*⟩ ⟨*CLOSING*⟩

⟨*BLOCK-BODY*⟩ ::= ⟨*DECLARE-GROUP*⟩
| ⟨*ANY-STATEMENT*⟩
| ⟨*BLOCK-BODY*⟩ ⟨*ANY-STATEMENT*⟩

⟨*FUNCTION-NAME*⟩ ::= ⟨*LABEL-EXTERNAL*⟩ `FUNCTION`

⟨*PROCEDURE-NAME*⟩ ::= ⟨*LABEL-EXTERNAL*⟩ `PROCEDURE`

⟨*FUNC-STMT-BODY*⟩ ::= ⟨*PARAMETER-LIST*⟩
| ⟨*TYPE-SPEC*⟩
| ⟨*PARAMETER-LIST*⟩ ⟨*TYPE-SPEC*⟩

⟨*PROC-STMT-BODY*⟩ ::= ⟨*PARAMETER-LIST*⟩
| ⟨*ASSIGN-LIST*⟩
| ⟨*PARAMETER-LIST*⟩ ⟨*ASSIGN-LIST*⟩

⟨*DECLARE-GROUP*⟩ ::= ⟨*DECLARE-ELEMENT*⟩
| ⟨*DECLARE-GROUP*⟩ ⟨*DECLARE-ELEMENT*⟩

⟨*DECLARE-ELEMENT*⟩ ::= ⟨*DECLARE-STATEMENT*⟩
| ⟨*REPLACE-STMT*⟩ ;
| ⟨*STRUCTURE-STMT*⟩
| `EQUATE EXTERNAL` ⟨*IDENTIFIER*⟩ `TO` ⟨*VARIABLE*⟩ ;

⟨*PARAMETER*⟩ ::= ⟨*IdentifierToken*⟩
| ⟨*BitIdentifierToken*⟩
| ⟨*CharIdentifierToken*⟩
| ⟨*StructIdentifierToken*⟩
| ⟨*EventToken*⟩
| ⟨*LabelToken*⟩

⟨*PARAMETER-LIST*⟩ ::= ⟨*PARAMETER-HEAD*⟩ ⟨*PARAMETER*⟩)

⟨*PARAMETER-HEAD*⟩ ::= (
| ⟨*PARAMETER-HEAD*⟩ ⟨*PARAMETER*⟩ ,

⟨*DECLARE-STATEMENT*⟩ ::= `DECLARE` ⟨*DECLARE-BODY*⟩ ;

⟨*ASSIGN-LIST*⟩ ::= ⟨*ASSIGN*⟩ ⟨*PARAMETER-LIST*⟩

⟨*TEXT*⟩ ::= ⟨*TextToken*⟩

⟨*REPLACE-STMT*⟩ ::= `REPLACE` ⟨*REPLACE-HEAD*⟩ `BY` ⟨*TEXT*⟩

⟨*REPLACE-HEAD*⟩ ::= ⟨*IDENTIFIER*⟩
| ⟨*IDENTIFIER*⟩ (⟨*ARG-LIST*⟩)

⟨*ARG-LIST*⟩ ::= ⟨*IDENTIFIER*⟩
| ⟨*ARG-LIST*⟩ , ⟨*IDENTIFIER*⟩

⟨*STRUCTURE-STMT*⟩ ::= `STRUCTURE` ⟨*STRUCT-STMT-HEAD*⟩ ⟨*STRUCT-STMT-TAIL*⟩

⟨*STRUCT-STMT-HEAD*⟩ ::= ⟨*STRUCTURE-ID*⟩ : ⟨*LEVEL*⟩
| ⟨*STRUCTURE-ID*⟩ ⟨*MINOR-ATTR-LIST*⟩ : ⟨*LEVEL*⟩
| ⟨*STRUCT-STMT-HEAD*⟩ ⟨*DECLARATION*⟩ , ⟨*LEVEL*⟩

⟨*STRUCT-STMT-TAIL*⟩ ::= ⟨*DECLARATION*⟩ ;

⟨*INLINE-DEFINITION*⟩ ::= ⟨*ARITH-INLINE*⟩
| ⟨*BIT-INLINE*⟩
| ⟨*CHAR-INLINE*⟩
| ⟨*STRUCTURE-EXP*⟩

⟨*ARITH-INLINE*⟩ ::= ⟨*ARITH-INLINE-DEF*⟩ ⟨*CLOSING*⟩ ;
| ⟨*ARITH-INLINE-DEF*⟩ ⟨*BLOCK-BODY*⟩ ⟨*CLOSING*⟩ ;

⟨*ARITH-INLINE-DEF*⟩ ::= `FUNCTION` ⟨*ARITH-SPEC*⟩ ;
| `FUNCTION` ;

⟨*BIT-INLINE*⟩ ::= ⟨*BIT-INLINE-DEF*⟩ ⟨*CLOSING*⟩ ;
| ⟨*BIT-INLINE-DEF*⟩ ⟨*BLOCK-BODY*⟩ ⟨*CLOSING*⟩ ;

⟨*BIT-INLINE-DEF*⟩ ::= `FUNCTION` ⟨*BIT-SPEC*⟩ ;

⟨*CHAR-INLINE*⟩ ::= ⟨*CHAR-INLINE-DEF*⟩ ⟨*CLOSING*⟩ ;
| ⟨*CHAR-INLINE-DEF*⟩ ⟨*BLOCK-BODY*⟩ ⟨*CLOSING*⟩ ;

⟨*CHAR-INLINE-DEF*⟩ ::= `FUNCTION` ⟨*CHAR-SPEC*⟩ ;

⟨*STRUC-INLINE-DEF*⟩ ::= `FUNCTION` ⟨*STRUCT-SPEC*⟩ ;