# Appendix A

## HARDWARE REPRESENTATION OF THE SOURCE LANGUAGE

A SIMULA program which is to be processed by the compiler should be contained in the job-input stream or put in a sequential data set with fixed or fixed blocked record format and 80 byte record length. It can also be a member of a partitioned data set with the same record characteristics. The program must be contained in columns 1-72 of the lines, and columns 73-80 may be used for sequence numbers (these columns will not be processed, but they will appear on the program listing). The EBCDIC code is used.

Basic symbols.

A basic symbol is represented as a keyword, a special character or a sequence of special characters (table A.1).

Identifiers.

Identifiers should follow the syntax of ALGOL 60, and the syntatic class <letter> consists both of lower and upper case letters. Lower case and upper case letters are not distinguished, except in text or character constants. 12 characters are significant in an identifier. An identifier must not be a keyword used to represent a basic symbol.

Constants.

Arithmetic constants follow the ALGOL 60 syntax. If an integer constant is out of integer range (3.1, p.2) it is regarded as a real constant. A constant can be given in hexadecimal as 1 to 16 hexadecimal digits, preceded by #. The constant is regarded as real if it is followed by an R.

A character constant is represented as the desired character, enclosed in single quotes (').

A text constant (<string>) is represented as the desired sequence of characters, enclosed in double quotes ("). A text constant may not contain a double quote without the use of a special convention: If a text string contains two double quotes in sequence, this is interpreted as one double quote within the text, and not the ending double qoute.

Any basic symbol, identifier or constant must be contained in one line, and it must not contain interspersed blanks. A text constant may, however, continue over several lines. Adjacent identifiers or basic symbols represented by keywords must be separated by line shift or at least one blank.

Comments.

A comments consists of the basic symbol COMMENT, followed by a nonalphameric character, and the successive characters up to and including the next semicolon (; or $).

An end comment is the string of characters following the basic symbol END up to, but not including the next semicolon, END, ELSE, WHEN or OTHERWISE.

Comments and end comments are printed on the program listing, but they are not further processed by the compiler.

A warning message is issued if an end comment contains the basic symbols ":=", ":-", "(", or "GOTO".

| basic symbol | representations (EBCDIC) | basic symbol | representations (EBCDIC) |
|---|---|---|---|
| = | = EQ | == | == |
| | <> NE | =/= | =/= |
| > | > GT | : | : |
| < | < LT | := | := |
| | >= GE | . | . |
| | <= LE | | |
| | OR | ; | ; $ |
| & | AND | ( | ( |
| | IMP | ) | ) |
| | EQV | | ( |
| | _ NOT | | ) |
| + | + | , | , |
| - | - | | & |
| * | * | | |
| | ** | | |
| / | / | | |
| | // | | |

Table A.1 Basic symbol representations.

Basic symbols consisting of boldface or underlined keywords in the reference language are represented as the corresponding word in capital letters. The basic symbol go to can be punched with or without space(s) between GO and TO.

Listing and input control lines.

The compilation listing and compiler input can be controlled by special lines put in the compiler input data set. Such lines are identified by a % sign in column 1, immediately followed by a control word. The allowed control lines are:

%TITLE text

The contents of columns 10-55 of this line are put in printer positions 24-69 of the headings of subsequent pages. Any old title text is overwritten. Source program listing continues with the next line on a new page. The %TITLE line itself is not listed.

%PAGE

The program listing continues with the next line listed on a new page. The %PAGE line is not listed.

%NOSOURCE

Suppress listing of the source program. The lines following this line up to the next %SOURCE line (if any) will not be listed. The line numbers will, however, be updated. THE %NOSOURCE line is not listed.

%SOURCE

Resume listing of the source program. The %SOURCE line is not listed.

%COPY text

The %COPY control line will obtain source-language coding from a partitioned or sequential data set and insert it into the program currently being compiled. The coding to be inserted is identified by the text on the line, which is interpreted as the name of a member in the partitioned data set.

The partitioned data set is identified by the SYSLIB DD-statement supplied to the compilation step.

If this member cannot be located, or if SYSLIB is not specified, the system will try to locate a sequential data set with the ddname identical to what was expected to be the member name.

The %COPY card will be listed.

%ENDCOPY

This line signals the end of predefined source-language coding obtained by a %COPY line. It must be the last line of any member in a partitioned data set which will be obtained by a %COPY line. The data set should have blocksize not greater than 1600. The code to be copied is inserted in the library by the IEBUPDTE or IEBUPDAT utility program (see (9)).

The %ENDCOPY line will be listed.

%RESWD=n

This has the same effect as the compiler parameter RESWD, see section 2.2.1.1.

%INDENT=n

This has the same effect as the compiler parameter INDENT, see section 2.2.1.1.

%NOINDENT

Suppress indentation of subsequent lines in the compilation listing, e.g. enabling visual distinction of label declarations in an otherwise automatically indented listing.

%INDENT

Resume the indentation at the earlier level.