Standard ECMA-55

Minimal BASIC

January 1978

Contents

BRIEF HISTORY

The first version of the language BASIC, acronym for Beginner's All-purpose Symbolic Instruction Code, was produced in 1964 at the Dartmouth College in the USA. This version of the language was oriented towards interactive use. Subsequently, a number of implementations of the language were prepared, that differed in part from the original one.

In 1974, the ECMA General assembly recognised the need for a standardised version of the language, and in September 1974 the first meeting of the ECMA Committee TC 21, BASIC, took place. In January 1974, a corresponding committee, X3J2, had been founded in the USA.

Through a strict co-operation it was possible to maintain full compatibility between the ANSI and the ECMA draft standards. The ANSI one was distributed for public comments in January 1976, and a number of comments were presented by ECMA.

A final version of the ECMA Standard was prepared at the meeting of June 1977 and adopted by the General Assembly of ECMA on Dec. 14, 1977 as Standrd ECMA-55.

1. SCOPE

This Standard ECMA-55 is designed to promote the interchangeability of BASIC programs among a variety of automatic data processing systems. Subsequent Standrds for the same prpose will describe extensions and enhancements to this Standard. Programs conforming to this Standard, as opposed to extensions or enhancements of this Standard, will be said to be written in "Minimal BASIC".

This standard establishes:

- the syntax of a program written in Minimal BASIC.
- The formats of data and the precision and range of numeric representations which are acceptable as input to a automatic data processing system being controlled by a program written in Minimal BASIC.
- The formats of data and the precision and range of numeric representations which are acceptable as output to a automatic data processing system being controlled by a program written in Minimal BASIC.
- The semantic rules for interpreting the meaning of a program written in Minimal BASIC.
- The errors and exceptional circumstances which shall be detected and also the manner in which such errors and exceptional circumstances shall be handled.

Although the BASIC language was originally designed primarily for interactive use, the Standard describes a language that is not so restricted.

The organisation of the Standard is outlines in Appendix 1. Thet method of syntax specification is explained in Appendix 2.

2. REFERENCES

ECMA-6 : 7-Bit Input/Output Coded Character Set, 4th Edition

ECMA-53 : Representation of Source Programs

3. DEFINITIONS

For the purposes of this Standard, the following terms have the meanings indicated.

3.1 BASIC

A term applied as a name to members of a special class of languages which possess similar syntaxes and semantic meanings; acronym for Beginner's All-purpose Symbolic Instruction Code.

3.2 Batch-mode

The processing of programs in an environment where no provision is made for user interaction.

3.3 End-of-line

The characters or indicator which identifies the termination of a line. Lines of three kinds may be identified in Minimal BASIC: program lines, print lines and input reply lines. End-of-line may vary between the three cases and may also vary depending upon context. Thus, for example, an end of input line may vary on a given system depending on the terminal being used in interactive or batch mode.

Typical examples of end-of-line are carriage-return, carriage-return line-feed, and end of record (such as end of card).

3.4 Error

A flaw in the syntax of a program which causes the program to be incorrect.

3.5 Exception

A circumstance arising in the course of executing a program which results from faulty data or computations or from exceeding some resource constraint. Where indicated certain exceptions (non-fatal exceptions) may be handled by the specified procedures; if no procedure is given (fatal exceptions) or if restrictions imposed by the hardware or operating environment make it impossible to follow the given procedure, then the exception shall be handled by terminating the program.

3.6 Identifier

A character string used to name a variable or a function.

3.7 Interactive mode

The processing of programs in an environment which permits the user to respond directly to the actions of individual programs and to control the commencement and termination of these programs.

3.8 Keyword

A character string, usually with the spelling of a commonly used or mnemonic word, which provides a distinctive identification of a statement or a component of a statement of a programming language.

The keywords in Minimal BASIC are: BASE, DATA, DEF, DIM, END, FOR, GO, GOSUB, GOTO, IF, INPUT, LET, NEXT, ON, OPTION, PRINT, RANDOMIZE, READ, REM, RESTORE, RETURN, STEP, STOP, SUB, THEN, and TO.

3.9 Line

A single transmission of characters which terminates with an end-of-line.

3.10 Nesting

A set of statements is nested within another set of statements when:

- the nested set is physically contiguous, and
- the nesting set (divided by the nested set) is non-null.

3.11 Print zone

A contiguous set of character positions in a printed output line which may contain an evaluated print statement element.

3.12 Rounding

The process by which the presentations of a value with lower precision is generated froma representation of higher precision taking into account the value of that portion of the original number which is to be omitted.

3.13 Significant digits

The contiguous sequence of digits ...

> NOTE: The Standard requires that the ability of a conforming implementation to accept numeric representations be measured in terms of significant digits rather than the actual number of digits (that is including leading or trailing zeroes) in the representation.

3.14 Truncation

The process by which the representation of a value with lower precision is generated from a representation of higher precision by merely deleting the unwanted low order digits of the original representation.

4. CHARACTERS AND STRINGS

4.1 General Description

The character set for BASIC is contained in the ECMA 7-bit coded character set. Strings are sequences of characters and are used in BASIC programs as comments (see 19), as string constants (see 6), or as data (see 15).

4.2 Syntax

1. letter

A/B/C/D/E/F/G/H/J/K/L/M/N/O/P/Q/R/S/T/U/V/W/X/Y/Z

2. digit

0/1/2/3/4/5/6/7/8/9

3. string-character

quotation-mark / quoted-string-character

4. quoted-string-character

exclamation-mark / number-sign / dollar-sign / percent-sign / ampersand / apostrophe / left-parenthesis / right-parenthesis / asterisk / comma / solidus / colon / semi-colon / less-than-sign / equals-sign / greater-than-sgn / question-mark / circumflex-accent / underline / unquoted-string-character

5. unquoted-string-character

space / plain-string-character

6. plain-string-character

plus-sign / minus-sign / full-stop / digit / letter

7. remark-string

string-character*

8. quoted-string

quotation-mark quoted-string-character* quotation-mark

9. unquoted-string

plain-string character / plain-string-character unquoted-string-character* plain-string-character

4.3 Examples

```
ANY CHARACTERS AT ALL (?!*!!) CAN BE USED IN A "REMARK".

"SPACES AND COMMAS CAN OCCUR IN QUOTED STRINGS."

COMMAS CANNOT OCCUR IN UNQUOTED STRINGS.
```

4.4 Semantics

The letters shall be the set of upper-case Roman letters contained in the ECMA 7-bit coded character set in positions 4/1 to 5/10.

The digits shall be the set of arabic digits contained in the ECMA 7-bit code...

...

The names of characters are specified in Table 1.

The coding of characters is specified in Table 2; however this coding applies only when programs and/or input/output data are exchanged by means of coded media.

4.5 Exceptions

None.

4.6 Remarks

...

5. PROGRAMS

5.1 General Description

BASIC is a line-oriented language. A BASIC program is a sequence of lines, the last of which shall be an end-line and each of which contains a keyword. Each line shall contain a unique line-number which serves as a label for the statement contained in that line.

5.2 Syntax

1. program

block* end-line

2. block

(line/for-block)*

3. line

line-number statement end-of-line

4. line-number

digit digit? digit? digit?

5. end-of-line

[implementation-defined]

6. end-line

line-number end-statement end-of-line

7. end-statement

END

8. statement

data-statement / def-statement / dimension-statement / gosub-statement / goto-statement / if-then-statement / input-statement / let-statement / on-goto-statement / option-statement / print-statement / randomize-statement / read-statement / remark-statement / restore-statement / return-statement / stop-statement

5.3 Examples

```
999 END
```

5.4 Semantics

A BASIC program shall be composed of a sequence of lines ordered by line-numbers, the last of which contains an end-statement. Program lines shall be executed in sequential order, starting with the first line, until

- some other action is dictated by a control statement, or
- an exception condition occurs, which results in a termination of the program, or
- a stop-statement or end-statement is executed.

Special conventions shall be observed regarding spaces. With the following exceptions, spaces occur anywhere in a BASIC program without affecting the execution of that program and may be used to improve the appearance and readability of the program.

Spaces shall not appear:

- at the beginning of a line
- within keywords
- within numeric constants
- within line numbers
- within function or variable names
- within two-character relation symbols

All keywords in a program shall be preceded by at least one space and, if not at the end of a line, shall be followed by at least one space.

Each line shall begin with a line-number. The values of the integers represented by the line-numbers shall be positive nonzero; leading zeroes shall have no effect. Statements shall occur in ascending line-number order.

The manner in which the end of a statement line is detected is determined by the implementation; e.g. the end-of-line may be a carriage-return character, a carriage-return character followed by a line-feed character, or the end of a physical record.

Lines in a standard-conforming program may contain up to 72 characters; the end-of-line indicator is not included within this 72 character limit.

The end-statement serves both to mark the physical end of the main body of a program and to terminate the execution of the program when encountered.

5.4 Exceptions

None.

5.6 Remarks

Local editing facilities may allow for the entry of statement lines in any order and also allow for duplicate line-numbers and lines containing only a line-number. Such editing facilities usually sort the program into the proper order and in the case of duplicate line numbers, the last line entered with that line-number is retained. In many implementations, a line containing only a line-number (without trailing spaces) is usually deleted from the program.

6. CONSTANTS

6.1 General Description

Constants can denote both scalar numeric values and string values.

A numeric-constant is a decimal representation in positional notation of a number. There are four general syntactic forms of (optionally signed) numeric constants:

- implicit point representation

`sd...d`

- explicit point unscaled representation

`sd..drd..d`

- explicit point scaled representation

`sd..drd..dEsd..d`

- implicit point scaled representation

`sd..dEsd..d`

where:

`d` is a decimal digit,
`r` is a full stop
`s` is an optional sign, and
`E` is the explicit character E.

A string-constant is a character string enclosed in quotation marks (see 4).

6.2 Syntax

1. numeric-constant

sign? numeric-rep

2. sign

plus-sign / minus-sign

3. numeric-rep

significand exrad?

4. significand

integer full-stop? / integer? fraction

5. integer

digit digit*

6. fraction

full-stop digit digit*

7. exrad

E sign? integer

8. string-constant

quoted-string

6.3 Examples

```
1          500          -21.          .255          1E10
5E-1       .4E+1
"XYZ"              "X - 3B2"              "1E10"
```

6.4 Semantics

The value of a numeric-constant is a the number represented by that constant. "E" stands for "times ten to the power"; if no sign follows the symbol "E", then a plus sign is understood. Spaces shall not occur in numeric-constants.

A program may contain numeric representations which have an arbitrary number of digits, though implementations may round the values of such representations to an implementation-defined precision of not less than six significant decimal digits. Numeric constants can also have an arbitrary number of digits in the exrad, though nonzero constants whose magnitude is outside an implementation-defined range will be treated as exceptions. The implementation defined range shall be at least 1E-38 to 1E+38. Constants whose magnitudes are less than machine infinitesimal shall be replaced by zero, while constants whose magnitudes are larger than machine infinity shall be diagnosed as causing an overflow.

A string-contant has as its value the string of all characters between the quotation marks; spaces shall not be ignored. The length of a string-constant, i.e. the number of characters contained between the quotation-marks, is limited only by the length of a line.

6.5 Exceptions

The evaluation of a numeric constant causes an overflow (non-fatal, the recommended recovery procedure is to supply machine infinity with the appropriate sign and continue).

6.6 Remarks

Since this Standard does not require that strings with more than 18 characters be assignable to string variables (see 7), conforming programs can use string constants with more than 18 characters only as elements in a print-list.

It is recommended that implementations report constants whose magnitudes are less than machine infinitesimal as underflows and continue.

7. VARIABLES

7.1 General Description

Variables in BASIC are associated with either numeric or string values and, in the case of numeric variables, may be either simple variables or references to element of one or two dimensional arrays; such references are called subscripted variables.

Simple numeric variables shall be named by a letter followed by an optional digit.

Subscripted numeric variables shall be named by a letter followed by one or two numeric expressions enclosed within parentheses.

String variables shall be named by a letter followed by a dollar sign.

Explicit declarations of variable types are not required; a dollar-sign serves to distinguish string from numeric variables, and the presence of a subscript distinguishes a subscripted variable from a simple one.

7.2 Syntax

1. variable

numeric-variable / string-variable

2. numeric-variable

simple-numeric-variable / numeric-array-element

3. simple-numberic-variable

letter digit?

4. numeric-array-element

numeric-array-name subscript

5. numeric-array-name

letter

6. subscript

left-parenthesis numeric-expression (comma numeric-expression)? right-parenthesis

7. string-variable

letter dollar-sign

7.3 Examples

```
X          A5          V(3)          W(X,X+Y/2)
S$         C$
```

7.4 Semantics

At any instant in the execution of a program, a numeric-variable is associated with a single numeric value and a string-variable is associated with a single string value. The value associated with a variable may be changed by the execution of statements in the program.

The length of the character string associated with a string-variable can vary during the execution of a program from a length of zero characters (signifying the null or empty string) to 18 characters.

Simple-numeric-variables and string-variables are declared implicitly through their appearance in a program.

A subscripted variable refers to the element in the one or two dimensional array selected by the value(s) of the subscript(s). The value of each subscript is rounded to the nearest integer. Unless explicitly declared in a dimension statement, subscripted variables are implicitly declared by their first appearance in a program. In this case the range of each subscript is from zero to ten inclusive, unless the presence of an option-statement indicates that the range is from one to ten inclusive. Subscript expressions shall have values within the appropriate range (see 18).

The same letter shall not be the name of both a simple variable and an array, nor the name of both a one-dimensional and a two-dimensional array.

There is no relationship between a numeric-variable and a string-variable whose names agree except for the dollar-sign.

At the initiation of execution the values associated with all variables shall be implementation-defined.

7.5 Exceptions

A subsc