

Specification for

**Computer
programming language
CORAL 66**

UDC 681.3.06CORAL 66

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI
Documents

AVA

Document Sharing Hub

Cooperating organizations

The Data Processing Standards Committee, under whose direction this British Standard was prepared, consists of representatives from the following Government departments and scientific and industrial organizations:

- British Computer Society Ltd.*
- British Paper and Board Industry Federation (PIF)
- British Printing Industries Federation
- Business Equipment Trade Association*
- Central Computer Agency (Civil Service Department)*
- Committee of London Clearing Bankers on behalf of the Committee of Scottish Clearing Bankers, Co-operative Bank, Central Trustee Savings Bank and Yorkshire Bank
- Department of Industry (Computers Systems and Electronics)
- Department of Industry (National Physical Laboratory)*
- Electricity Supply Industry in England and Wales*
- Government Communications Headquarters
- HM Customs and Excise
- Institute of Cost and Management Accountants
- Institute of Purchasing and Supply
- Institution of Electrical Engineers
- Institution of Mechanical Engineers
- Inter-university Committee on Computing
- London Transport Executive
- Ministry of Defence*
- National Computer Users' Forum
- National Computing Centre Ltd.*
- National Research Development Corporation
- Post Office*
- Society of British Aerospace Companies Limited

The organizations marked with an asterisk in the above list, together with the following, were directly represented on the committee entrusted with the preparation of this British Standard:

- Association for Literary and Linguistic Computing
- Association of Computer Units in Colleges of Higher Education (ACUCHE)
- British Gas Corporation
- Computing Services Association
- Control and Automation Manufacturer's Association (BEAMA)
- Edinburgh Regional Computing Centre
- Engineering Equipment Users' Association
- Hatfield Polytechnic
- University of London

This British Standard, having been prepared under the direction of the Data Processing Standards Committee, was published under the authority of the Executive Board and comes into effect on 31 October 1980

© BSI 04-2000

The following BSI references relate to the work on this standard:
Committee reference IST/5
Draft for comment 78/63605 DC

ISBN 0 580 11442 2

Amendments issued since publication

Amd. No.	Date of issue	Comments
5079	October 1986	Indicated by a sideline in the margin

Contents

	Page
Cooperating organizations	Inside front cover
Foreword	ii
1 Scope	1
2 References	1
3 Definitions	1
4 Syntactic metalanguage	1
5 Compliance	2
5.1 Implementations	2
5.2 Programs	8
6 Requirements	8
6.1 The CORAL 66 program	8
6.2 Scoping	9
6.3 References to data	10
6.4 Place references: switches	15
6.5 Expressions	15
6.6 Statements	19
6.7 Procedures	23
6.8 Communicators	26
6.9 Names and constants	27
6.10 Processing text in a program	29
6.11 List of language symbols and character set	30
6.12 Permissible options	31
Appendix A Unspecified features	33
Appendix B Implementation	33
Table 1 — Alphabetical list of syntax rules	2
Table 2 — Parameters of procedures	24
Table 3 — Language words	30
Table 4 — Other symbols	31
Publications referred to	Inside back cover

Foreword

This British Standard has been prepared under the direction of the Data Processing Standards Committee and is based on the “Official definition of CORAL 66”, first published in 1970 by Her Majesty’s Stationery Office, and reproduces material taken from that publication. This standard follows some ten years after the “Official definition” and the BSI committee believes that the standard should not contain requirements incompatible with the very large number of existing implementations. For this reason, although a number of proposals for extensions of the language have been considered, their adoption has, in general, been reserved until a future revision of the standard.

Two exceptions have, however, been made in this first edition:

- a) the number denotation HEX has been included (see **6.9.2**);
- b) the language words CORAL and SEGMENT have been included to give a formal syntax to a multisegment program (see **6.1.3**).

Future revisions may make fuller provision for byte-addressable machines with a byte-oriented, but otherwise conformable, language definition. The current requirements assume a word-addressable virtual machine, but some provision for such byte-addressable machines is made in Appendix B.

This standard aims to achieve an overall economy, in terms of human effort, in the development of computer-based systems for real-time applications and to protect the interests of the user by encouraging the use of computers that comply with this standard.

CORAL 66 provides a means of increasing the implementation efficiency of computer applications in environments where the input/output communication requirements may not have been standardized and that are time-critical. CORAL 66 is a kernel high-level computer language intended to replace a high proportion of assembly code in each specific application.

Items not explicitly specified in this standard should be clearly understood as being unspecified. These unspecified items result from an original design objective that CORAL 66 should favour all computer architectures as equally as possible and should allow implementors to exploit hardware features as efficiently as possible. The objective is to promote the use of a common form of expression whenever it is expedient to do so. The language specification thus explicitly includes both the insertion of machine code statements and anonymous referencing, so as to ensure that special machine features and peripherals can be handled efficiently at the cost of reduced program portability. Furthermore, implementors are thus able to adopt the hardware conventions of the target machine with regard to numeric representation and computation. The parameterized macro facility enables the source text to retain a high level of readability. CORAL 66 therefore aims to minimize, rather than to eliminate, the consequent machine dependency of programs and to maximize the portability of programming and software maintenance staff.

It is virtually impossible to design a standard language such that programs will run with equally high efficiency in all types of computer and in any applications. Much of the design of CORAL 66 reflects this difficulty. For example, the language permits the use of non-standard “code statements” for any parts of a program where it may be important to exploit particular hardware facilities. A special feature is scaled fixed-point arithmetic for use in small fixed-point machines; the floating-point facilities of the language can be omitted when hardware limitations make the use of floating-point arithmetic uneconomical. Other features can also be omitted without reducing the power of the language to an unacceptably low level. Major features that may be omitted are listed in **6.12**.

A clear distinction needs to be made between general-purpose languages and more limited languages designed to incorporate the inbuilt assumptions of specialized applications or to make direct computer access practical for the non-specialist user. CORAL 66 belongs to the first category. Languages in this class are suitable for writing compilers and interpreters as well as for direct application. Special-purpose languages can therefore be implemented by means of software written in CORAL 66, backed up as required with suites of specialized macros or procedures. It is largely for this reason that the facilities for using procedures have been kept as general as possible. The main differences between CORAL 66 procedures and those of ALGOL 60 lie in the replacement of the ALGOL 60 dynamic “name parameter” by the more efficient “location” or reference parameter used in FORTRAN, and the requirement to declare recursive procedures explicitly as such, again in the interest of object-code efficiency.

The theory and structure of programming for real-time computer applications has not yet advanced to a point where a particular choice of language facilities is inevitable. Furthermore, the design of real-time languages is handicapped by the lack of agreed standard software interfaces for applications programmers or compiler writers. This does not imply that real-time programs cannot yet be written in high-level language. The use of CORAL 66 in real-time applications implies the presence of a supervisory system for the control of communications, which may have been designed independently of the compiler. The programmer’s control over external events, and the computer’s reaction to them, is expressed by the use of procedures or macros that communicate with the outside world indirectly through the agency of the supervisory software. No requirements are specified in this standard regarding the names or action of such calls on the supervisor.

Editorial note. It is normal convention in British Standards to use italic type for algebraic quantities. Since the status of such quantities contained in this standard may or may not directly represent true variable quantities, this convention has not been adopted in this standard.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, pages i to iv, pages 1 to 34, an inside back cover and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

1 Scope

This British Standard specifies the semantics and syntax of the computer programming language CORAL 66 by specifying requirements for a compiler and for a conforming program.

NOTE 1 Any feature that is not explicitly specified in this standard has been intentionally left unspecified. Appendix A lists some unspecified features.

Appendix B gives additional information regarding implementation.

NOTE 2 The examples given among the specification requirements are not intended to add to or detract from the requirements, but are included purely to aid understanding.

2 References

The titles of the publications referred to in this standard are listed on the inside back cover.

3 Definitions

For the purposes of this British Standard, the following definitions, together with those for other terms given in BS 3527, apply.

3.1

layout characters

the six “layout characters” and the space character specified in BS 4730

3.2

printing characters

the graphic characters specified in BS 4730

NOTE To draw attention to the language concepts, some terms are printed in *italics* on their first mention in this standard.

4 Syntactic metalanguage¹⁾

The widespread use of syntax-driven methods of compilation lends increasing importance to syntax methods of language description. This standard specifies the syntax of CORAL 66, and therefore starts with broad structure, working downwards to finer detail. For reasons of legibility, the customary Backus notation has been abandoned in favour of a system relying on typographical layout. Each syntax rule has on its left-hand side a class name, such as “Statement”. Such names appear in lower case without spaces, and with an initial capital letter. On the right-hand side of a rule are the various alternative expansions for the class name. These alternatives are either each printed on a new line or separated by “or”, as appropriate. Where a single alternative spreads over more than one line of print, the additional lines are inset in relation to the starting position of the alternative. Each alternative expansion consists of a sequence of items separated by spaces. The items themselves are either further class names or terminal symbols, such as BEGIN. The class name “Void” is used for an empty class. For example, a typical pair of rules might be

Specimen	=	ALPHA Sign
		BETA Sign
Sign	=	+
		—
		Void

Examples of specimens complying with these rules are ALPHA + and BETA.

The equals sign is used to separate the left-hand side from the right, except after its first appearance in a rule.

Certain constructs are defined informally in this standard; these are underlined.

The syntax rules are listed in Table 1.

¹⁾ A British Standard syntactic metalanguage is at an early stage of preparation.

5 Compliance

5.1 Implementations

5.1.1 Any implementation of CORAL 66 that can accept all of the features of the language specified in clause 6 and with the meanings defined in clause 6 complies with the requirements of this standard.

5.1.2 Except as specifically allowed by 6.12, an implementation that omits any of the features of the language specified in clause 6 or that includes such a feature, but with a meaning altered in any way from that specified in clause 6, does not comply with the requirements of this standard.

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Absolutecommunicator	= <u>defined in a particular implementation to conform to the style of the commoncommunicator</u>	6.8.5
Actual	= Expression Wordreference Destination Name	6.6.4
Actuallist	= Actual Actual, Actuallist	6.6.4
Addoperator	= + —	6.5.2
Alternative	= Statement	6.6.10
Answerspec	= Numbertype Void	6.7.2
Answerstatement	= ANSWER Expression	6.6.5
Arraydec	= Numbertype ARRAY Arraylist Presetlist	6.3.3
Arrayitem	= Idlist [Sizelist]	6.3.3
Arraylist	= Arrayitem Arrayitem, Arraylist	6.3.3
Assignmentstatement	= Variable ← Expression	6.6.2
Base	= Id Id [Signedinteger]	6.3.8
Bitposition	= Integer	6.3.4.3.1
Block	= BEGIN Declist; Statementlist END	6.2.1
Booleanword	= Booleanword2 Booleanword4 DIFFER Booleanword5	6.5.4
Booleanword2	= Booleanword3 Booleanword5 UNION Booleanword6	6.5.4
Booleanword3	= Booleanword6 MASK Typedprimary	6.5.4
Booleanword4	= Booleanword Typedprimary	6.5.4
Booleanword5	= Booleanword2 Typedprimary	6.5.4
Booleanword6	= Booleanword3 Typedprimary	6.5.4
Bracketedcomment	= <u>(any sequence of characters in which round brackets are matched)</u>	6.10.3
Codesequences	= <u>defined in a particular implementation</u>	6.6.6
Codestatement	= CODE BEGIN Codesequences END	6.6.6

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Commentsentence	= COMMENT <u>any sequence of characters not including a semicolon;</u>	6.10.3
Commoncommunicator	= COMMON (Commonitemlist)	6.8.2
Commonitem	= Datadec Overlaydec Placespec Procedurespec Void	6.8.2
Commonitemlist	= Commonitem Commonitem; Commonitemlist	6.8.2
Comparator	= < or ≤ or = or ≥ or > or ≠	6.5.6.2
Comparison	= Simpleexpression Comparator Simpleexpression	6.5.6.2
Compoundstatement	= BEGIN Statementlist END	6.6.7
Condition	= Condition OR Subcondition Subcondition	6.5.6.2
Conditionalexpression	= IF Condition THEN Expression ELSE Expression	6.5.6.1
Conditionalstatement	= IF Condition THEN Consequence IF Condition THEN Consequence ELSE Alternative	6.6.10
Consequence	= Simplestatement Label: Consequence	6.6.10
Constant	= Number Addoperator Number	6.9.2
Constantlist	= Group Group, Constantlist	6.3.6.2
Datadec	= Numberdec Arraydec Tabledec	6.2.1
Dec	= Datadec Overlaydec Switchdec Proceduredec	6.2.1
Declist	= Dec Dec; Declist	6.2.1
Destination	= Label Switch [Index]	6.6.3
Digit	= 0 or 1 or 2 or 3 or 4 or 5 or 6 or 7 or 8 or 9	6.9.1
Digitlist	= Digit Digit Digitlist	6.9.2
Dimension	= Lowerbound : Upperbound	6.3.3
Dummystatement	= Void	6.6.9
Elementdec	= Id Numbertype Wordposition Id Partwordtype Wordposition, Bitposition	6.3.4.3.1

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Elementdeclist	= Elementdec Elementdec; Elementdeclist	6.3.4.2
Elementpresetlist	= PRESET Constantlist Void	6.3.6.3
Elementscale	= (Totalbits, Fractionbits) (Totalbits)	6.3.4.3.3
Endcomment	= Id	6.10.3
Expression	= Unconditionalexpression Conditionalexpression	6.5.1
Externalcommunicator	= <u>defined in a particular implementation to conform to the style of the commoncommunicator</u>	6.8.4
Factor	= Primary Booleanword	6.5.3.1
Forelement	= Expression Expression WHILE Condition Expression STEP Expression UNTIL Expression	6.6.11.1
Forlist	= Forelement Forelement, Forlist	6.6.11.1
Forstatement	= FOR Wordreference ← Forlist DO Statement	6.6.11.1
Fractionbits	= Signedinteger	6.3.1
Gotostatement	= GOTO Destination	6.6.3
Group	= Constant (Constantlist) Void	6.3.6.2
Hexdigit	= Digit A or B or C or D or E or F	6.9.2
Hexlist	= Hexdigit Hexdigit Hexlist	6.9.2
Id	= Letter Letterdigitstring	6.9.1
Idlist	= Id Id, Idlist	6.3.2
Index	= Expression	6.5.3.4
Integer	= Digitlist HEX (Hexlist) OCTAL (Octallist) LITERAL (<u>printing character</u>)	6.9.2
Label	= Id	6.4
Labellist	= Label Label, Labellist	6.4
Length	= Integer	6.3.4.2
Letter	= a or b or c or d or e or f or g or h or i or j or k or l or m or n or o or p or q or r or s or t or u or v or w or x or y or z	6.9.1
Letterdigitstring	= Letter Letterdigitstring Digit Letterdigitstring Void	6.9.1

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Librarycommunicator	= <u>defined in a particular implementation to conform to the style of the commoncommunicator</u>	6.8.3
Lowerbound	= Signedinteger	6.3.3
Macrobody	= <u>any sequence of characters in which string quotes are matched</u>	6.10.3 and 6.11
Macrocall	= Macroname Macroname (Macrostringlist)	6.10.3
Macrodefinition	= DEFINE Macroname † Macrobody †; DEFINE Macroname (Idlist) † Macrobody †;	6.10.3
Macrodeletion	= DELETE Macroname;	6.10.3
Macroname	= Id	6.10.3
Macrostring	= <u>any sequence of characters in which string quotes are matched, commas are protected by round or square brackets and in which such brackets are properly matched and nested</u>	6.10.3
Macrostringlist	= Macrostring Macrostring, Macrostringlist	6.10.3
Multoperator	= * /	6.5.2
Name	= Id	6.6.4
Number	= Real Integer	6.9.2
Numberdec	= Numbertype Idlist Presetlist	6.3.2
Numbertype	= FLOATING FIXED Scale INTEGER	6.3.1
Octaldigit	= 0 or 1 or 2 or 3 or 4 or 5 or 6 or 7	6.9.2
Octallist	= Octaldigit Octaldigit Octallist	6.9.2
Overlaydec	= OVERLAY Base WITH Datadec	6.3.8
Parameterspec	= Specifier Idlist Tablespec Procedurespec	6.7.4.1 and 6.7.4.10
Parameterspeclist	= Parameterspec Parameterspec; Parameterspeclist	6.7.3
Partword	= Id [Index] BITS [Totalbits, Bitposition] Typedprimary	6.5.3.5
Partwordreference	= Id [Index] BITS [Totalbits, Bitposition] Wordreference	6.6.2
Partwordtype	= Elementscale UNSIGNED Elementscale	6.3.4.3.3
Placespec	= LABEL Idlist SWITCH Idlist	6.8.2
Presetlist	= ‹ Constantlist Void	6.3.6.2

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Primary	= Untypedprimary Typedprimary	6.5.3.1
Procedurecall	= Id Id (Actuallist)	6.6.4
Proceduredec	= Answerspec PROCEDURE Procedureheading; Statement Answerspec RECURSIVE Procedureheading; Statement	6.7.1
Procedureheading	= Id Id (Parameterspeclist)	6.7.3
Procedurespec	= Answerspec PROCEDURE Procparamlist	6.7.4.9
Procparameter	= Id Id (Typelist)	6.7.4.9
Procparamlist	= Procparameter Procparameter, Procparamlist	6.7.4.9
Real	= Digitlist . Digitlist Digitlist ₁₀ Signedinteger ₁₀ Signedinteger Digitlist . Digitlist ₁₀ Signedinteger HEX (Hexlist . Hexlist) OCTAL (Octallist . Octallist)	6.9.2
Scale	= (Totalbits, Fractionbits)	6.3.1
Signedinteger	= Integer Addoperator Integer	6.9.2
Simpleexpression	= Term Addoperator Term Simpleexpression Addoperator Term	6.5.2
Simplestatement	= Assignmentstatement Gotostatement Procedurecall Answerstatement Codestatement Compoundstatement Block Dummystatement	6.6.1
Sizelist	= Dimension Dimension, Dimension	6.3.3
Specifier	= VALUE Numbertype LOCATION Numbertype Numbertype ARRAY LABEL SWITCH	6.7.4.1
Statement	= Label : Statement Simplestatement Conditionalstatement Forstatement	6.6.1

Table 1 — Alphabetical list of syntax rules

Syntax rule		Reference
Statementlist	= Statement Statement; Statementlist	6.6.7
String	= ‹ <u>sequence of characters with quotes matched</u> ›	6.9.4 and 6.11
Subcondition	= Subcondition AND Comparison Comparison	6.5.6.2
Switch	= Id	6.4
Switchdec	= SWITCH Switch ‹← Labellist	6.4
Tabledec	= TABLE Id [Width, Length] [Elementdeclist Elementpresetlist] Presetlist	6.3.4.2
Tablesdec	= TABLE Id [Width, Length] [Elementdeclist]	6.7.4.6
Term	= Factor Term Multoperator Factor	6.5.2
Totalbits	= Integer	6.3.1
Type	= Specifier TABLE Answerspec PROCEDURE	6.7.4.9
Typedprimary	= Wordreference Partword LOCATION (Wordreference) Numbertype (Expression) Procedurecall Integer	6.5.3.3
Typelist	= Type Type, Typelist	6.7.4.9
Unconditionalexpression	= Simpleexpression String	6.5.1
Untypedprimary	= Real (Expression)	6.5.3.2
Upperbound	= Signedinteger	6.3.3
Variable	= Wordreference Partwordreference	6.6.2
Width	= Integer	6.3.4.2
Wordposition	= Signedinteger	6.3.4.3.1
Wordreference	= Id Id [Index] Id [Index, Index] [Index]	6.5.3.4