# Mini-COBOL

*By* P. Giles*

**A detailed specification of a subset of COBOL for teaching use with all compilers is described in a format similar to the COBOL 1965 report. The subset has been designed so that it will compile on almost all existing compilers with the very minimum of alterations. It has also been biased, where possible, towards the inclusion of features demonstrating the basic principles of COBOL and of clear and well structured programs. A test program and further details are available on request from U.K. Giro Account 13 493 0002, for a transfer of five shillings.**

(Received January 1969)



The major problem in the efficient utilisation of computers in commercial and administrative areas lies in the inability to state the detailed application in such a way that computer programs are developed and maintained with a minimum of time and programming effort (Harder, 1968). Besides ameliorating this problem, COBOL was intended to speed up the training of personnel in the design of data processing systems, and in the development of computer programs for such systems. At first, the running time of an object program was naturally compared unfavourably with that of the corresponding program coded directly in assembly language. As a result, efficiency in the compiled object code took a high place in the order of priorities of the compiler writers. The natural development of the language was therefore somewhat biased towards the efficiency requirements of users and a wide variety of options was provided, thus increasing the points of difference between machines and making the practical application of compatibility more difficult. If basic COBOL can be taught to systems people as well as to programmers, the communication gap between the two will be much reduced and file structures will be more precisely defined in terms of the problem to be solved.

The national bodies that examine students are concerned with the problem of compatibility between colleges. A completely compatible language—even at a very low level—would ease some of the problems of teaching and make COBOL appear to the novice to be much easier to learn. It would also establish a standard by which competency in elementary programming could be measured. This subset has been designed with these aims in view, and after considerable discussion and criticism within the COBOL Specialist Group of the British Computer Society.

## Documentation and ease of understanding

Data description at the elementary level has been restricted to the use of the PICTURE clause (COBOL 65 Manual), but the COPY clause has been dropped because it is not available in several small compilers and is of little value where students are writing many short disconnected programs. The frequent use of NOTE should ensure that all programs are readily understood and should buttress the reduction in readability caused by limiting data-names to six characters for very small machines. For the same reason, qualification and sections have been dropped, although they are valuable aids to understanding. The problem of compatible printer format was incapable of a complete solution, except by an insistence on uniform spacing for every WRITE verb. By permitting spacing from 0 to 2 lines the impression is given that any desired number could be obtained in a large configuration.

## Compatibility

One of the main aims of Mini-COBOL is compatibility across all existing compilers, subject only to the restriction that every feature included must be good COBOL. For this reason, research started with two existing small compilers and discussion centred round the highest common factor of these two. Anything that exemplified points of principle in programming practice was retained wherever possible—for example the labelling of magnetic tape, or other files. Features that add polish, or permit the programmer to write less well-structured programs more easily, as does ALTER, were dropped. Arithmetic expressions and condition names were left out with regret. This resulted in a simple and straightforward structure in the Procedure Division, entirely compatible except for the verb WRITE. In the Data Division some difficulty was experienced with LABEL RECORDS STANDARD, because COBOL 65 makes provision for greater operating flexibility by permitting the label to be specified directly to the operating system, without appearing in the program itself anywhere. However, it was found that in practice, in one large much-used compiler, the use of VALUE OF, although not required, did not prevent a successful compilation. The caution diagnostic produced could be ignored with safety. For good documentation, it would seem helpful to specify the contents of the label record within the program; and it seemed a pity to omit the whole subject of file labels just because of a more advanced development in the flexibility of operation of large machines. For the same reason, it was thought to to be good documentation practice to insist on a statement of the memory size required, expressed in a standard unit, within the Configuration Section. In any case, this section must be rewritten on any change of machine, so a conversion from CHARACTERS to WORDS can then be carried out if necessary.

* *Scottish Amicable Life Assurance Society, P.O. Box* 13 *Craigforth, Stirling*

Any remaining incompatibilities have been listed in the Appendix together with details of the machines and compilers concerned.

**Specification**

The following specification has been drawn from the COBOL, Edition 1965, Report published by the U.S. Government Printing Office.

*Chapter 1—Notation used in Format*

A General Format is the specific arrangement of the elements of a clause, a statement, or a paragraph.

Elements, which make up a clause or a statement, consist of upper-case words, lower-case words, level numbers, brackets, braces and special characters.

Lower-case words must conform to the rules for the formation of COBOL words, except for statement (Chapter 2) and character-string (Chapter 6 under PICTURE).

Level numbers, appearing within the framework of Data Division entries, are required when that entry is used.

When a portion of a general format is enclosed in square brackets, [ ], that portion may be included or omitted at the user's choice. Braces, { }, enclosing a portion of a general format, mean that a selection of one of the options contained within the braces must be made. In both cases, a choice is indicated by vertically stacking the possibilities.

In the general format, the ellipsis represents where repetition may occur at the user's option. The portion of the format that may be repeated is determined as follows. Given ... in a clause or statement format, scanning from right to left, determine the ] immediately to the left of the ... ; continue scanning right to left and determine the logically matching [ ; the ... applies to the words between the determined pair of delimiters.

*Chapter 2—Definitions that differ from those of COBOL 65*

Character Set The complete character set consists of the 41 characters described as numeric, alphabetic, editing, or punctuation characters.

| EDITING FUNCTION | CHARACTER GRAPHIC | PURPOSE |
|---|---|---|
| Sign control symbol | – | Fixed insertion in numeric edited items or literals. |
| Zero suppression | Z | Suppression of leading zeros by substitution for 9 but only to the left of the decimal point. |
| Decimal point | · | Special insertion in numeric edited items serving both alignment and physical representation. |

| PUNCTUATION FUNCTION | CHARACTER GRAPHIC | PURPOSE |
|---|---|---|
| Period or full stop | . | To terminate a source program entry. |
| Quotation mark | " | To bound a nonnumeric literal. |
| Left parenthesis | ( | To bound subscripts, or repetition in the Picture clause. |
| Right parenthesis | ) | |
| Space | | To act as the word separator. |

Condition A simple condition only, that is, any single relation condition.

Constant, Figurative Either the reserved word that represents a numeric value (ZERO), or the reserved word that represents a string of one or more blank characters (SPACES).

Data-Name, Subscripted An identifier that is composed of a data-name followed by one subscript enclosed in parentheses.

Level Number A number composed of two digits that must have one of the values from 01 to 05 inclusive.

Literal A string of not more than 30 characters bounded by quotation marks; or composed of not more than ten numeric characters, that may contain either or both a decimal point, that cannot be the rightmost character, and a unary operator that must be the leftmost character.

Operator, Unary A minus (—) sign that is prefixed to a literal. A plus sign is implied in the absence of a minus sign.

Paragraph A sentence containing an EXIT statement must form a paragraph on its own.

Sentence A sequence of one or more statements, the last of which is terminated by a period followed by a space. A compiler directing statement, or the statement EXIT must each form a sentence on its own.

Statement A compiler directing statement begins with the verb NOTE. A conditional statement is either an IF statement, a READ statement, or a statement which specifies a SIZE ERROR option.

Table There must not be more than 30 tables in one program and no one table must require more than 4,000 characters.

Word A string of not more than six characters, of which the first is alphabetic, from the set of 37.

Word, Reserved Besides the words listed in the COBOL 65 Manual, those listed by each implementor should also be avoided.

*Chapter 3—Language concepts*

The complete character set for Mini-COBOL consists of the 41 characters defined in Chapter 2. Where necessary character substitutions may be made, such as apostrophe for quotation mark. Similarly, implementors may provide for words exceeding six characters in length, but all words used in a Mini-COBOL program must be limited to the definition given in Chapter 2, in order to ensure that all compilers distinguish between different words by the same criteria. Consecutive spaces may be used to improve readability.

A figurative constant may be used wherever a literal appears in a format, except that numeric literals do not permit the insertion of SPACES. The figurative constant consists of one character, when it is the operand of a DISPLAY or STOP verb; elsewhere it consists of a string of characters equal in length to the operand with which the statement associates it.

*Chapter 4—Identification Division*

The program-name must conform to the rules for a word and must not include hyphen. In some implementations, the last two characters must be numeric and will then form a priority number.

*Chapter 5—Environment Division*

In this division, which is composed of two sections, all names other than file-names must be specified by the implementor. In the Configuration Section, CHARACTERS should be used in preference to WORDS where the implementor permits, since this provides a more accurate measure of the memory size required by the object program. The size should not make provision for the executive or supervisor, unless the implementor specifies otherwise. Either or both of these may require to be altered whenever the Environment Division is rewritten for a different configuration. In the Input-Output Section, all files named must bear a one-to-one correspondence to the files described in the Data Division. The number of files must not exceed six, and each must be assigned to a different hardware unit. If more than one format is to be handled by one hardware unit, then each must be described as a separate DATA RECORD within the file assigned to that unit.

*Chapter 6—Data Division*

This division is subdivided into the File Section and the Working-Storage Section. The File Section defines the contents of data files stored on an external medium. Each file is defined by a file description followed by a record description or a series of record descriptions. The file description entry consists of the level indicator FD followed by a data-name and a set of independent clauses. These clauses specify the relation between logical and physical records, the name of the label record contained in the file, and the names of the data records which comprise the file.

A record description has hierarchical structure and therefore the clauses used with an entry may vary considerably, depending whether or not it is followed by subordinate entries. Every entry of file or data description is terminated by a period. Any clauses used must be present in the order specified.

The Working-Storage Section describes records which are not part of external data files but are developed and processed internally. They include constants whose value is assigned in the source program but does not change during the execution of the object program.

BLOCK integer RECORDS must be omitted for a file that is assigned to a card reader, punch, or printer, or when integer takes the value 1. The implementor must specify when it may be used and when LABEL RECORDS STANDARD may be used. The literal must be nonnumeric and composed of not more than eight characters. Both a LABEL RECORD clause and a DATA RECORDS clause are required for every file-name. Every record-name must appear in a data description entry under level 01 and the order of appearance should be the same as in the preceding file description. The presence of more than one record-name indicates that the file contains more than one type of data record and that they all share the same area of memory. Any input operation on the file will overwrite data in *all* record descriptions for that file.

Level number may have any value from 01 to 05 inclusive and the following clauses must occur in the order given. FILLER, REDEFINES, and OCCURS are not permitted at level 01. Neither REDEFINES nor OCCURS are permitted in an entry which is subordinate to an entry containing an OCCURS clause, and they must not both be present in the same entry. PICTURE must be used for every elementary item and nowhere else. VALUE must be used for every constant elementary item within the Working-Storage Section and nowhere else. No VALUE clause may form part of an entry subordinate to an entry containing REDEFINES or OCCURS. The literal in a VALUE clause must satisfy the rules for a MOVE into the item in which it is used.

When a REDEFINES clause is used, the entries describing data-name must immediately follow the entries describing previous-data-name, and the level of the first entry for each must be the same. Previous-data-name may not be subscripted nor subject to REDEFINES. The memory areas described by previous-data-name and by data-name must be the same, and any sub-division into elementary items must subdivide the area in the same way. Redefinition starts at previous-data-name and ends when the same level number is encountered in the entry containing REDEFINES. Every OCCURS clause must specify an integer larger than zero as a numeric literal. The OCCURS clause is used to define tables which are described in Chapter 3. The data-name which is the subject of an entry containing OCCURS must be subscripted whenever it is used as an operand, as must any subordinate data-name.

The PICTURE clause must not contain more than 15 characters in the string. The size of the data item thus defined must not exceed 10 characters and sign if numeric or 30 characters in any other case. There are three categories of data that can be described with a PICTURE clause—alphanumeric (X), numeric (9 S V), and numeric edited (Z 9 . V –). Apart from these characters specified no other may appear in the character string except a digit within parentheses specifying recurrence of the previous character a stated number of times. S indicating the presence of an operational sign and V indicating the assumed decimal point may not occur more than once each in any PICTURE. In a numeric edited PICTURE any of the editing characters may appear but X or S must not appear. Only one decimal point—either V or · — may be present and the sign symbol – must be either the first or last character of the PICTURE if it is present. The size of a data item defined by a PICTURE consists of the number of characters present, or implied by recurrence, in the PICTURE excluding S or V. Every PICTURE must be terminated by a period which indicates the completion of that entry.

*Chapter 7—Procedure Division*

This division is composed of paragraphs, sentences, and statements. An imperative sentence is a sequence of one or more imperative statements, terminated by a period and a space. Wherever an imperative statement is permitted in a format, a series of imperative statements is also permitted. Wherever a literal appears in a format, a figurative constant may be used. Every word in this division of the source program must be either a reserved word, a word defined in the Environment or Data Division, a paragraph-name not used in any other division, a literal, or a figurative constant.

A condition causes the object program to select between alternate paths of control, depending upon the truth value of a test. Conditions are used in IF state-

ments and cause a comparison of two operands, which must be either both numeric or both alphabetic. They can be literals composed of the corresponding characters and bounded by quotation marks only if alphabetic, or an appropriate figurative constant. For numeric operands the algebraic values are compared. Zero is considered a unique value regardless of the sign. Unsigned numeric operands are treated as positive for purposes of comparison. Alphabetic operands must be of the same length. Z is at the high end of the alphabet and space is lower than A.

The ROUNDED option and the SIZE ERROR option may either or both be used in any of the five arithmetic statements. The resultant is that identifier associated with the result of an arithmetic statement. When rounding is requested the absolute value of the resultant is increased by one in the least significant digit whenever the most significant truncated digit is greater than or equal to five. If, after decimal point alignment, the value of a result exceeds the largest value that can be contained in the resultant, then a size error condition exists. Rounding takes place before checking for size error. Division by zero always causes this condition. If SIZE ERROR is specified, then the resultant is not altered by the statement but the imperative statement in the option is executed. Otherwise the resultant is unpredictable.

The READ statement makes available the next logical record from an input file and specifies performance of imperative statement(s) when end of file is detected. Only one area is allocated to each file, even if the file contains more than one type of record. Each READ statement causes a new record to replace the one read by the previous READ statement. An OPEN statement must be executed for a file prior to the execution of the first READ statement for that file. After the execution of the imperative statement of the AT END phrase, a READ statement for that file must not be given without prior execution of a CLOSE statement and an OPEN statement for that file.

In the five ADD, SUBTRACT, MULTIPLY, and DIVIDE statements every identifier must refer to a numeric elementary item, except that the identifier immediately following GIVING may refer to a numeric edited elementary item. Each literal must be numeric and ZERO must not be used. Decimal point alignment is automatic and truncation of digits at either end of the resultant may occur. The SIZE ERROR and ROUNDED options are provided for such a contingency. Truncation at the most significant end may otherwise produce an unpredictable resultant. The operational sign may be truncated.

Any MOVE statement in which the sending and receiving items are both elementary items is an elementary MOVE. Both such items must have the same category specified in their PICTURE clause, except that where the receiving item is numeric edited the sending item must not be other than numeric, and where the receiving item is alphanumeric the sending item may have any category other than numeric. Where the sending item is numeric, alignment by decimal point and any necessary zero filling takes place except where zeros are replaced because of editing requirements. If the sending item has more digits to the left or right of the decimal point than the receiving item can contain, the excess digits are truncated. If the receiving item has no operational sign, the absolute value of the sending item is used. Where the sending item is not numeric, its characters are successively transferred to those of the receiving field starting with the leftmost. The remainder of the receiving item, if any, is filled with spaces. The receiving field must not be too small, since truncation would occur. A numeric literal or ZERO belongs to the category numeric. A nonnumeric literal or SPACES belongs to the category alphanumeric. In any MOVE that is not an elementary MOVE, both the sending and receiving items must be composed of corresponding elementary items specified in the same order in the PICTURE of each group item. Each pair of corresponding elementary items must be equal in size and must be alphanumeric. The sending and receiving items of each pair must satisfy the rules for an elementary MOVE.

DISPLAY may operate on an elementary item, a group item, or a literal provided that its length does not exceed 30 characters. If a figurative constant is specified only one character will be displayed.

A WRITE statement cannot be executed for a file unless the file is open. The record-name is the name of a logical record in that file, which is no longer available after the WRITE statement has been executed. All output records for a single file occupy the same area of storage. Vertical spacing of a record on the printed page is specified by integer, which may only take the value 0, 1 or 2.

The OPEN statement for a file must be executed prior to the first READ or WRITE statement for that file. A second OPEN statement for a file cannot be executed prior to the execution of a CLOSE statement for that file. The OPEN statement initiates the processing of both input and output files. It performs checking and/or writing of labels and other input-output operations. It does not obtain or release the first data record.

If a CLOSE statement has been executed for a file, a READ or WRITE statement for that file must not be executed unless an intervening OPEN statement for that file is executed. A file must be opened before it can be closed. Only one CLOSE can be given for a file for each time that it has been opened. When a file is closed, the data area reserved for the file is released, the reel is rewound if the file is on magnetic tape, and other closing conventions specified by the implementor are performed. If LABEL RECORDS STANDARD is specified and if ending labels have been implemented, then they are written on output files, but only checked on input files if the imperative statement in the AT END phrase has been executed since the preceding OPEN statement.

The GO TO statement must be the only or the last statement in a sentence. An automatic return to the statement following a PERFORM statement is established after the specified exit-paragraph, or after the last statement of the performed paragraph, if the THRU option has not been employed. This last statement must not be a GO TO statement. If control passes to these paragraphs by means other than a PERFORM statement, control passes through the last statement of the procedure to the following statement, as if no PERFORM statement mentioned this procedure. There is no necessary relationship between paragraph-name and exit-paragraph-name except that a consecutive

sequence of operations is to be executed commencing with the first and terminating with the EXIT. A procedure to be performed may include GO TO and/or PERFORM statements except as stated above. If it is possible to replace two GO TO statements by a PERFORM statement this should be done in order to simplify and clarify the logical sequence of operations thus defined. If a procedure referred to by a PERFORM statement includes another PERFORM statement, the procedures specified by the included PERFORM must each either be totally included in, or totally excluded from any procedure specified by the including PERFORM. Thus, if two or more procedures, each called by a separate PERFORM, have both commenced their execution but not reached their corresponding exit paragraphs at any point of their execution, then any procedure that commences within another procedure must not contain or specify the exit paragraph of that other procedure.

The single word statement EXIT must appear in a paragraph by itself. The NOTE statement alone must form the last sentence of the paragraph in which a commentary is required. The commentary will be produced on the listing, but will not be compiled. Any

```
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. computer-name.
OBJECT-COMPUTER. computer-name MEMORY integer {CHARACTERS / WORDS}.
INPUT-OUTPUT SECTION.
FILE-CONTROL. SELECT file-name-1 ASSIGN implementor-name-1.
    [ SELECT file-name-2 ASSIGN implementor-name-2. ] . . .

DATA DIVISION.

FILE SECTION.
FD file-name [ BLOCK integer RECORDS ]
    {LABEL RECORDS OMITTED / LABEL RECORDS STANDARD VALUE OF IDENTIFICATION IS literal}
    DATA RECORDS record-name-1 [ record-name-2 ] . . .  .
level-number {data-name / FILLER} [ REDEFINES previous-data-name ]
    [ OCCURS integer TIMES ] PICTURE character string.

[ WORKING-STORAGE SECTION.
level-number {data-name / FILLER} [ REDEFINES previous-data-name ]
    [ OCCURS integer TIMES ] PICTURE character string [ VALUE literal ] . ]

PROCEDURE DIVISION.
paragraph-name.
      sentence. [ sentence. ] . . .
[ exit-paragraph-name.
      EXIT. ]
[ paragraph-name.
    [[ imperative-statement [ imperative-statement ]. . . . ] . . .
     [ conditional-statement. ] . . .      ] . . .
     [ NOTE character string. ] ] . . .
```

**Format 1**

combination of the characters from the allowable character set, excluding only the period, may be included in the character string.

STOP literal displays the literal to the operator and halts the program temporarily. The program may be restarted at the next statement. STOP RUN institutes the program ending procedure specified by the implementor.

*Chapter 8—Reference Format*

This is the standard method of describing COBOL source programs in terms of character positions within a line on an input or output medium. Six positions from the leftmost margin L are allocated to a sequence number, if required. Division and section names, level indicators, level 01 and paragraph names must begin at the eighth position—margin A—and each must occupy a line by itself, except for level indicators and numbers which form part of the following entry. This following entry, and every sentence in the Procedure Division must begin at the twelfth position—margin B—as must any continuation of an entry or sentence from the previous line. Words must not be broken and hyphenated to make continuation lines. No text may

```
IF identifier [ NOT ] { GREATER THAN | LESS THAN | EQUAL TO } { literal | identifier } imperative-statements.

READ file-name AT END imperative-statements.

ADD { literal | identifier } { literal | identifier } GIVING identifier [ ROUNDED ]
    [ ON SIZE ERROR imperative-statements. ]

ADD { literal | identifier } TO identifier [ ROUNDED ]
    [ ON SIZE ERROR imperative-statements. ]

SUBTRACT { literal | identifier } FROM { literal | identifier } [ GIVING identifier ][ ROUNDED ]
    [ ON SIZE ERROR imperative-statements. ]

MULTIPLY { literal | identifier } BY { literal | identifier } GIVING identifier [ ROUNDED ]
    [ ON SIZE ERROR imperative-statements. ]

DIVIDE { literal | identifier } INTO { literal | identifier } GIVING identifier [ ROUNDED ]
    [ ON SIZE ERROR imperative-statements. ]

MOVE { literal | identifier } TO identifier

DISPLAY { nonnumeric literal | identifier }

WRITE print-record-name { BEFORE | AFTER } integer

WRITE nonprint-record-name

OPEN { INPUT | OUTPUT } file-name

CLOSE file-name

GO TO paragraph-name.

PERFORM paragraph-name [ THRU exit-paragraph-name ]

EXIT.

NOTE character string.

STOP { literal | RUN. }
```

**Format 2**

continue beyond the rightmost position—margin R. It is preferable that each elementary statement should occupy a line by itself. This will simplify corrections.

The preferred positions for indentation of subordinate level numbers are positions 12, 14, 16, and 18. The preferred position for PICTURE clause is position 30.

**Acknowledgements and requests for additions to the Appendix**

Grateful acknowledgement is made to the considerable assistance provided by many friends, too numerous to mention individually, and often contacted through the COBOL Study Group and for much assistance with typing. All readers are earnestly requested to supply any further entries for the Appendix for compilers for new or old machines. Teachers do not always have access to modern machines, so entries for old compilers may be valuable to them. Any other comments on clarification or alteration in these specifications will be welcome and will be taken into account in future work in this field.

**Conclusion**

This Mini-COBOL specification has demonstrated that a complete standardisation of COBOL at a level useful for teaching purposes is practicable. The compilers which are already available demonstrate that it can be implemented on machines with a core store of about 8,000 characters, provided that some form of backing store such as discs or magnetic tape is available. The Appendix shows some of the non-standard features present in existing compilers and should assist other work at present being carried out towards a minimum common standard of COBOL implementation by the Nationalised Industries E.D.P. Committee. This Mini-COBOL specification, however, does not aim at providing an efficient object program, but rather at providing a tool for teaching a basic understanding of the main principles of COBOL.

# Appendix

**Compilers requiring changes to be made in a Mini-COBOL program**

(*a*) Compilers with no requirements for alterations—except for the Environment Division and possibly for interchange between BEFORE and AFTER for the WRITE verb or deletion of the VALUE OF IDENTIFICATION clause.

* Bull Gamma 30. † GE 400 and GE 115.

(*b*) Alterations compatible with COBOL 65.

ICT 1300 compilers—the use of REDEFINES is restricted and only GO TO is permitted following AT END or SIZE ERROR. Input files may require TYPE.

ICT Rapidwrite—Words are restricted to five characters.

Honeywell B and C—Replace SPACES by SPACE—otherwise fatal.

ICT Compact—Replace ZERO by ZEROS—otherwise fatal.

ICT Compilers—A record count Word must be inserted in all magnetic tape files.

IBM 360/30 (DOS)—RECORDING MODE is required for unit record files and card record descriptions must be padded up to 80 bytes.

(*c*) Alterations incompatible with COBOL 65.

ICT 1300 and 1900—all compilers—insert V adjacent to decimal point in all numeric edited items—otherwise handled as integer.

Honeywell L—Division by zero produces a fatal object time diagnostic.

* Tested by user.
† Tested by supplier.

**References**

HARDER, E. L. (1968). The Expanding World of Computers, *Comm. Assoc. Comp. Mach.*, Vol. 11, p. 234.
COBOL Report, Edition 1965—Department of Defence, Government Printing Office, Washington, D.C. 20402, $1.75.
Formal Definition of the Syntax of COBOL, European Computer Manufacturers Association, 114 Rue du Rhone, 1204 Geneva.