GC28-6473-3

Program Product

IBM DOS/VS COBOL Compiler and Library General Information

Program Numbers: 5746-CB1 (Compiler and Library)
5746-LM4 (Library)

IBM

Preface

This publication contains information to aid data systems planners and analysts in evaluating and planning for the use of the DOS/VS COBOL Compiler and Library Program Product. The DOS/VS COBOL Subroutine Library can also be ordered separately.

DOS/VS COBOL allows source programs written in American National Standard COBOL to be processed in a DOS virtual environment. DOS/VS COBOL is compatible with the highest level of American National Standard COBOL, X3.23-1968, and with international standard ISO/R 1989-1972 Programming Language COBOL. CODASYL-Specified and IBM-Specified extensions are also implemented.

The DOS/VS COBOL Compiler and Library Program Product operates under control of DOS/VS. DOS/VS can operate as an independent system or under control of VM/370.

This publication is not a specification manual. Proposed specifications for the IBM DOS/VS COBOL Program Products are given in the publication:

Program Product Design Objectives: IBM DOS/VS COBOL Compiler and Library, GC28-6474

Fourth Edition (April 1976)

This edition, as amended by technical newsletters GN20-9140, GN20-9134, and GN20-9292, applies to Release 2 of the IBM DOS/VS COBOL Compiler and Library, program numbers 5746-CB1 and 5746-LM4, and to any subsequent releases unless otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Amendments" following the table of contents. Specific changes made as of this publishing date are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are continually made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography,* GC20-0001, for the editions that are applicable and current.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

Comments may be addressed to IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California U. S. A. 95150. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Contents

Figures

Tables

SUMMARY OF AMENDMENTS

Number 3

Support for fixed block devices is provided under DOS/VSE with VSE/Advanced Function, Release 1.

Number 2

Support has been added for the 3330-11 Disk Storage and 3350 Direct Access Storage devices.

Number 1

Support has been added to run DOS/VS COBOL under control of VM/370 CMS Release 3.

DOS/VS COBOL programs can be compiled in CMS and then executed in a DOS virtual machine, or under a DOS system.

The following restrictions apply to execution of DOS/VS COBOL programs in CMS:

1. Indexed files (DTFIS) are not supported. Various clauses and statements are therefore invalid: RECORD KEY, APPLY CYL-OVERFLOW, NOMINAL KEY, APPLY MASTER/CYL-INDEX, TRACK-AREA, APPLY CORE-INDEX and START.
2. Creating direct files is restricted as follows:
   —For U and V recording modes, access mode must be sequential.
   —For ACCESS IS SEQUENTIAL, track identifier must not be modified.
3. None of the user label-handling functions are supported. Therefore, the label-handling format of USE is invalid. The data-name option of the LABEL RECORDS clause is invalid.
4. There is no Sort or Segmentation feature.
5. ASCII-encoded tape files are not supported.
6. Spanned records (S-mode) processing is not available. This means that the S-mode default (block size smaller than record size) cannot be specified, and that the RECORDING MODE IS S clause cannot be specified.

In addition, multitasking, multipartition operation, and teleprocessing functions are not supported when executing under CMS.

Introduction

This publication describes briefly the DOS/VS COBOL Compiler and Library, a Program Product that offers advanced programming facilities to reduce program development time and increase programmer productivity. Separate chapters describe:

- DOS/VS COBOL Compiler features, grouped by the improved functions they make possible:
  - — Virtual System Support
  - — Advanced Application Programs
  - — Programmer Productivity Aids
  - — Efficient Object-Time Performance
  - — Improved Compile-Time Performance
- DOS/VS COBOL Subroutine Library Features
- DOS/VS COBOL Language Base

In addition, appendixes list the other COBOL Program Products available, and DOS/VS COBOL additions to the reserved word list.

Data Set Compatibility

The VSAM Compatibility Interface allows the user to perform all ISAM-like functions on VSAM data sets using existing COBOL ISAM programs. By converting old ISAM data sets to VSAM format, the user can continue present operation with only minor JCL changes. The resulting VSAM data set usually requires less frequent reorganization than the ISAM data set, because VSAM uses free space within the data set for updating.

The data format of VSAM files is common to both DOS/VS and OS/VS. Thus a data set created for one system can be used without change by programs written to interface with the other.

Data set compatibility exists between the DOS/VS COBOL Compiler and the other IBM DOS COBOL Compilers: all versions of the Full American National Standard COBOL Compiler, the Subset American National Standard COBOL Compiler, and the COBOL D Compiler. That is, data sets created by a program compiled on one of these compilers can be processed by a program compiled on the DOS/VS COBOL Compiler.

Programming Compatibility

The IBM DOS/VS COBOL Compiler is compatible with all versions of the IBM DOS Full American National Standard COBOL Compiler, and with the Subset American National Standard COBOL Compiler.

A number of reserved words as defined in CODASYL COBOL have been added to the reserved word list (see Appendix B). None of these words should be specified as user-defined names in a COBOL source program. The publication *IBM DOS Full American National Standard COBOL,* Order No. GC28-6394, gives the complete list of reserved words.

Source programs written for the COBOL D Compiler must be converted before compiling them on the DOS/VS COBOL Compiler. The Language Conversion Program described in the publication *IBM System/360 Conversion Aids: COBOL-to-American National Standard COBOL Language Conversion Program,* Order No. GC28-6400, facilitates such conversions.

The DOS/VS COBOL Library (5746-LM4) will operate with programs compiled by the DOS Full American National Standard COBOL, Version 3 (5736-CB2), DOS Full American National Standard COBOL, Version 2 (360N-CB-482) and DOS Subset American National Standard COBOL (5736-CB1) compilers. Thus a user having such programs need not recompile when going to the DOS/VS COBOL Program Product.

Object Modules from the above mentioned compilers may be linked together in a program when operating under the DOS/VS Operating System.

System Requirements

Compile-time Machine Requirements	System/370 Models 115, 125, 135, 145, 155II, or 158 4 Required Work Files: SYS001 (Disk), SYS002-SYS004 (Tape or Disk) 3 Optional Work Files: SYSLNK (Disk, for CATAL or LINK options); SYS005 (Tape or Disk, for Symbolic Debug); SYS006 (Disk, for FIPS Flagger)
Object-time Machine Requirements	System/370 1 Work File (Tape or Disk) when Symbolic Debug is used Input/Output Devices used by the object program
Virtual Memory	64K Compiler
Devices Supported	3203 132-character Printer 5203 96/120/132-character Printer 3886 Optical Character Reader 3330-11 Disk Storage 3340 Direct Access Storage 3350 Direct Access Storage 3540 Diskette Input/Output Unit 5425 Multifunction Card Unit Fixed block direct access storage devices (Plus all devices supported by DOS Full American National Standard COBOL, Version 3)

IBM DOS/VS COBOL Compiler

The DOS/VS COBOL Compiler makes it possible to process American National Standard COBOL programs in a DOS virtual environment. The language implemented is compatible with the highest level of American National Standard COBOL, X3.23-1968, and with international standard ISO/R 1989-1972 Programming Language COBOL. CODASYL-specified and IBM-specified extensions to these standards are also implemented.

The added and improved functions of the DOS/VS COBOL Compiler make possible advanced program applications. Existing applications can obtain improved performance when updated to take advantage of DOS/VS COBOL features. Other DOS/VS COBOL features make program development easy, and give efficient compile-time and object-time performance.

VIRTUAL SYSTEM SUPPORT, through DOS/VS, uses the performance improvements and large storage capacity of the Virtual System.

ADVANCED APPLICATION PROGRAMS can be achieved when the following DOS/VS COBOL features are used:

- *VSAM Support* — VSAM is a high-performance access method with a high degree of data security. Through an index, access can be sequential or random; physical sequential access is also provided.
- *Advanced System/370 Device Support* — multifunction card devices, advanced printers, optical mark readers, optical character readers, high-speed disk facilities, and the diskette input/output unit are all supported.
- *MERGE Support* — The MERGE verb enables two or more identically-sequenced input files to be combined into a single output file by specifying a set of keys. Both standard sequential and sequentially accessed VSAM files can be designated as input or output.

PROGRAMMER PRODUCTIVITY AIDS are provided through high-level language debugging and efficiency check features. (Surveys show that in the past up to 30% of a programmer's development time was spent debugging.)

- *Syntax-Check Feature* — optionally provides a quick scan of the source program without producing object code. Syntax checking can be unconditional or conditional.
- *FIPS (Federal Information Processing Standard) Flagger* — ensures that DOS/VS COBOL source programs can be written to conform to the Federal Information Processing Standard, by identifying nonstandard clauses and statements.
- *Lister (Release 2 only)* — Provides specially formatted source listing with embedded cross references for increased intelligibility and ease of use. Reformatted source deck is available as an option.

- *SORT-OPTION Clause (Release 2 Only)* — gives the COBOL programmer greater control over operations of the Sort/Merge program.
- *Verb Profiles (Release 2 only)* — Facilitates identifying and locating verbs in the COBOL source program. Options provide verb summary or a verb cross reference listing which includes the verb summary.
- *Symbolic Debug* — during program execution, COBOL-formatted snapshots and maps of the Data Division can be obtained, either at specified points during execution or at abnormal termination. Any number of debugging runs can be executed without recompilation, and without source language changes.
- *Flow Trace* — shows program flow up to the point of abnormal termination. The path of execution within and between user-specified modules can be traced. No COBOL source language changes are needed.
- *Statement Number Option* — provides information about the COBOL statement being executed at abnormal termination.
- *Execution Time Statistics (Release 2 only)* — Maintains a count of the number of times each verb in the COBOL source program is executed during an individual program execution.
- *WHEN-COMPILED Special Register* — Provides a means of associating a compilation listing with both the object program and the output produced at execution time.
- *Detailed Data and Procedure Division Information* — through expanded functions of the condensed listing (CLIST) or symbolic map (SYM).

EFFICIENT OBJECT-TIME PERFORMANCE can be achieved with DOS/VS COBOL. The following features make it possible:

- *COBOL Optimizer* — when requested, can reduce generated Procedure Division code by as much as 30%, as compared with code produced without requesting the feature.
- *System/370 Support* — automatic generation of System/370 instructions saves object program space and speeds up execution. High-performance System/370 devices also speed up execution, as well as allowing advanced applications.

IMPROVED COMPILE-TIME PERFORMANCE is achieved with the sorted cross-reference feature described below.

- *Speedy Sorted Cross-Reference (SXREF)* — gives compile-time performance up to 2 times faster than for Version 2 ANS COBOL source ordered cross-reference (XREF). Alphabetized cross-reference listings make it easier to find references to user-specified names.

Each of these DOS/VS COBOL features is described in greater detail in the following sections. In addition, all features of DOS Full American National standard COBOL, Version 3, are still available with DOS/VS COBOL, including the following:

- ASCII Support
- Separately Signed Numeric Data
- Double-buffered ISAM
- Improvements in the MOVE Statement and in Comparisons

Virtual System Support

The DOS/VS COBOL Compiler and Library operate under control of DOS/VS. DOS/VS makes available as much as 16 million bytes of Virtual Storage, and the ability to use high-performance VSAM (Virtual Storage Access Method). DOS/VS operates as an independent system or under control of VM/370. Powerful System/370 instructions are automatically generated and the performance improvements of advanced System/370 devices can be exploited.

VSAM (Virtual Storage Access Method)

VSAM is a high-performance access method of DOS/VS for use with direct access storage. VSAM offers high-speed retrieval and storage of data, flexible data organization, ease of use — including simplified job control statements — data protection against unauthorized access, central control of data management functions, cross-system compatibility, device independence (freedom from consideration of block sizes, control information, record deblocking, etc.), and ease of conversion from other access methods. VSAM uses a multifunction utility program known as Access Method Services to define a VSAM data set, to load records into it, to convert an existing indexed or sequential data set to VSAM format, and to perform other tasks as well. VSAM allows key-sequenced and entry-sequenced data sets.

In a *key-sequenced* data set, records are stored in the ascending collating sequence of an embedded key field. Records can be retrieved sequentially in key sequence; they can also be retrieved randomly according to the value of the desired key. VSAM uses the contents of the key field and optional free space (space in the data set not occupied by data) to insert new records in their key sequence.

In an *entry-sequenced* data set, the records are stored in the order in which they are presented for inclusion in the data set. New records are stored at the end of the data set. In COBOL, record retrieval must be sequential.

DOS/VS COBOL makes possible two types of file processing through VSAM: indexed and sequential.

Indexed File Processing

A VSAM indexed (key-sequenced) file is ordered in the ascending sequence of its record key — which is embedded in the record and whose value must not change. The physical sequence of the records in the file corresponds to

the ascending collating sequence of the record key. Records can be processed either sequentially or randomly, and can be fixed or variable in length.

In sequential processing, records are accessed in the ascending order of their record key values.

In random processing the sequence of record retrieval is controlled by the programmer. The desired record is accessed through the value placed in its RECORD KEY data item. A full or partial key value can be used.

Performance Considerations: In a VSAM data set, inserted records are stored and addressed the same way as original records; thus access speed after many insertions is equivalent to access speed before insertions. Free space allows efficient automatic reorganization of the data set by the access method; thus, there is seldom need to reorganize the data set offline.

Sequential File Processing

In a VSAM sequential (entry-sequenced) file, data is stored in the order in which it is received. In COBOL, records can be retrieved only in the same order in which they were stored. There is no key field. New records are always stored at the end of the data set. Records can be fixed or variable in length.

Programming Considerations: A record cannot be deleted from the file; however, its space can be reused for a new record of the same length.

If file reorganization becomes necessary, then a new file must be created, using records from the presently existing file.

System/370 Device Support

DOS/VS COBOL supports the following advanced System/370 devices, making possible new programming applications and enhanced performance.

3203 Advanced Printer — with 132-character print line

5203 Advanced Printer — with 120/132 character print line at compile time, and 96/120/132 character print line for user files at object time

3886 Optical Character Reader — reads multiline alphanumeric or numeric machine-printed documents or numeric hand-printed documents, with stacker selection

High-speed, large-capacity mass storage facilities:
- 3330-11 Disk Storage
- 3340 Direct Access Storage
- 3350 Direct Access Storage
- Fixed Block Direct Access Storage devices

3540 Diskette Input/Output Unit — transfers IBM diskette data directly to the System/370, and transfers System/370 data directly to IBM diskettes (Data for initial entry via diskette is prepared using the IBM 3740 Data Entry System)

5425 Multifunction Card Unit (MFCU) — For 96-column cards, with read/punch/print/select features and combined function processing. Without the combined function features, the 5425 MFCU can be used as a reader or as a punch.

All devices supported by DOS Full American National Standard COBOL Version 3 are supported by DOS/VS COBOL. These include: 3504/3505/3881 (with OMR), 3525 (with RCE and combined function processing), 2560 MFCM, 3410/3420 tapes, and 3330 disk. (To specify the new System/370 devices for an existing COBOL program, the user must modify the source program and recompile.)

MERGE Support

The MERGE verb allows the COBOL user to combine two or more identically ordered input files into one output file according to embedded key(s) in the record. Special processing of merged records can also be specified. More than one MERGE statement can be executed in one program. Both standard sequential files and sequentially accessed VSAM files can be designated as input or output. The Program Product DOS/VS Sort-Merge (Program Number 5746-SM1) is designed to be used with the MERGE statement.

Syntax-Check Feature

This feature optionally produces a quick scan of the source program for syntax errors without producing object code. Syntax-checking can be unconditional or conditional.

When unconditional syntax-checking is requested, the compiler scans the source text for syntax errors, and generates the appropriate error messages, but does not generate object text.

When conditional syntax-checking is requested, the compiler scans the source text for syntax errors, and generates the appropriate error messages. If no message exceeds the W or C level, a full compilation is produced. If one or more E or D level messages are produced, the compiler generates the messages, but does not generate object text.

(A few syntax errors may not be detected when a syntax-checking compilation is requested. When the compiler is released, a list of such errors will be made available.)

When object text is not produced, all of the following compile-time options, if specified, are suppressed:

OPTION control statement: LINK, DECK, XREF

CBL statement: SXREF, CLIST, LISTX, PMAP, SYMDMP, TRUNC, OPTIMIZE, FLOW, STATE

Unconditional syntax-checking is assumed if all of the following compile-time options are specified:

OPTION control statement: NOLINK, NOXREF, NODECK, NOLIST

CBL statement: SUPMAP (and CLIST, XREF, and PMAP are *not* specified)

If neither unconditional nor conditional syntax-checking is specified, or if unconditional syntax-checking is not assumed, a full compilation — including error messages, object text, and all other specified (or default) options — is produced.

Performance Considerations: Compile time is significantly reduced when object code is not produced.

FIPS (Federal Information Processing Standard) Flagger

The Federal Information Processing Standard is a compatible subset of American National Standard COBOL, X3.23-1968. FIPS recognizes four language levels: low, low-intermediate, high-intermediate, and full. When the FIPS Flagger is used, source clauses and statements that do not conform to the specified level of FIPS are identified. This assists the user in creating DOS/VS COBOL programs which conform to that standard.

Programming Considerations: At system generation time, no flagging, NOLVL (which is the system generation default), or flagging at a specified FIPS level, LVL=A/B/C/D, can be specified as the installation default option. At compile time, the programmer can override any of these options by specifying another level of FIPS flagging; if he specifies NOLVL, however, the option is ignored.

Lister (Release 2 Only)

The Lister option permits the user to obtain a reformatted detailed source code listing that contains complete cross-reference information at the *source* level. Each COBOL statement is begun on a new line, and indented in a manner that makes the logic of the program readily apparent, by highlighting level numbers, nested IF statements, etc. Each line of the reformatted source listing contains one or more references to other source statements, specifying the statement number, and indicating the type of reference. Figure 1 gives an example of Lister output.

```
GRANTZ                                                          10/06/73  PAGE 1


    1 IDENTIFICATION DIVISION.
    2   PROGRAM-ID. GRANTZ.
    3 ENVIRONMENT DIVISION.
    4 CONFIGURATION SECTION.
    5   SOURCE-COMPUTER. IBM-370.
    6   OBJECT-COMPUTER. IBM-370.
    7 INPUT-OUTPUT SECTION.
    8   FILE-CONTROL.
    9       SELECT INPUT-BUFFER ASSIGN TO SYS005-UR-2540R-S.          21
   11       SELECT OUTPUT-BUFFER ASSIGN TO SYS006-UR-1403-S.          29
   13       SELECT FILEX-DIAGNOSTICS ASSIGN TO SYS007-UT-2400-S.      38
   15       SELECT FILEY-WORKFILE ASSIGN TO SYS008-UT-2400-S.         53
   17       SELECT FILEZ-WORKFILE ASSIGN TO SYS009-UT-2400-S.         64
```

```
GRANTZ                                                          10/06/73  PAGE 2


   19 DATA DIVISION.
   20 FILE SECTION.
   21 FD  INPUT-BUFFER                                          9E,318E,328R
          LABEL RECORDS ARE OMITTED.
   23   01  INPUT-CARD.                                        333U
   24     02  INPUT-BYTE          OCCURS 72 TIMES INDEXED BY   337Q,659Q,664U,675Q,678Q,681Q,684Q,687Q,690Q,693Q,696Q,A
   25           INPUT-INDEX              PIC X.                322C,336C,657Q,659X,660C,664X,6660,675X,678X,681X,684X,B
   26     02  FILLER                    PIC X(8).
    *
    *
```

❶ Reference to FD statement number.

❷ FD referred to by SELECT statement.

❸ SELECT statement; (E) denotes Environment Division reference.

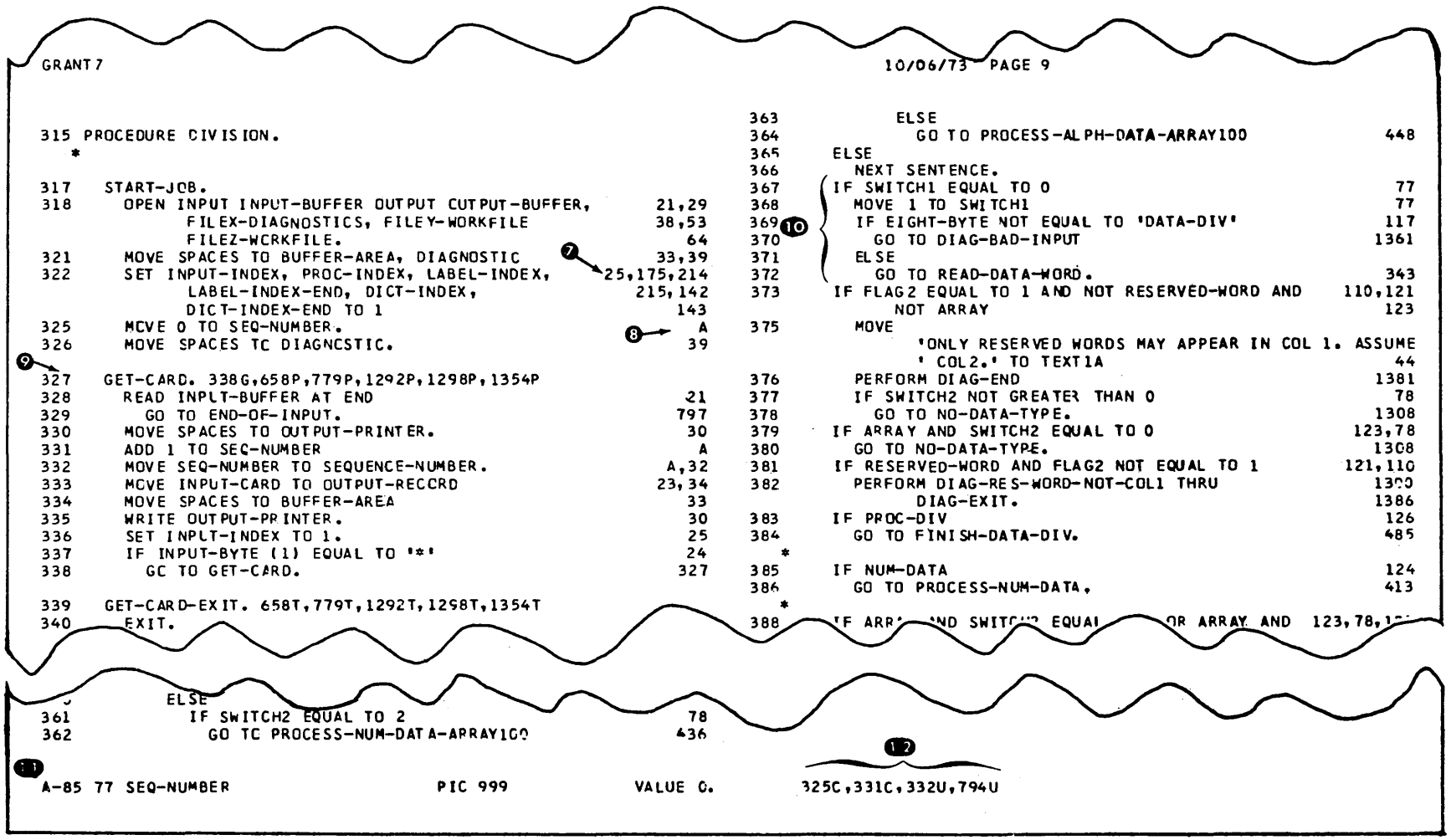
❹ Procedure Division statement; (R) denotes that statement 328 reads this data item.

❺ Data item changed by statement 322.

❻ Footnotes for additional references at bottom of this page.

Figure 1. Example of Lister Option Output (Part 1 of 2)

Figure 1. Example of Lister Option Output (Part 2 of 2)



```
GRANT 7                                                    10/06/73  PAGE 9


315 PROCEDURE DIVISION.
    *
317   START-JOB.
318     OPEN INPUT INPUT-BUFFER OUTPUT OUTPUT-BUFFER,          21,29
              FILEX-DIAGNOSTICS, FILEY-WORKFILE              38,53
              FILEZ-WORKFILE.                                   64
321     MOVE SPACES TO BUFFER-AREA, DIAGNOSTIC               33,39
322     SET INPUT-INDEX, PROC-INDEX, LABEL-INDEX,         25,175,214
              LABEL-INDEX-END, DICT-INDEX,                 215,142
              DICT-INDEX-END TO 1                             143
325     MOVE 0 TO SEQ-NUMBER.                                   A
326     MOVE SPACES TO DIAGNOSTIC.                             39
    *
327   GET-CARD. 338G,658P,779P,1292P,1298P,1354P
328     READ INPUT-BUFFER AT END                               21
329       GO TO END-OF-INPUT.                                 797
330     MOVE SPACES TO OUTPUT-PRINTER.                         30
331     ADD 1 TO SEQ-NUMBER                                     A
332     MOVE SEQ-NUMBER TO SEQUENCE-NUMBER.                   A,32
333     MOVE INPUT-CARD TO OUTPUT-RECORD                     23,34
334     MOVE SPACES TO BUFFER-AREA                             33
335     WRITE OUTPUT-PRINTER.                                  30
336     SET INPUT-INDEX TO 1.                                  25
337     IF INPUT-BYTE (1) EQUAL TO '*'                         24
338       GO TO GET-CARD.                                     327
    *
339   GET-CARD-EXIT. 658T,779T,1292T,1298T,1354T
340     EXIT.

363           ELSE
364             GO TO PROCESS-ALPH-DATA-ARRAY100                448
365       ELSE
366         NEXT SENTENCE.
367       IF SWITCH1 EQUAL TO 0                                  77
368         MOVE 1 TO SWITCH1                                    77
369         IF EIGHT-BYTE NOT EQUAL TO 'DATA-DIV'               117
370           GO TO DIAG-BAD-INPUT                             1361
371         ELSE
372           GO TO READ-DATA-WORD.                             343
373       IF FLAG2 EQUAL TO 1 AND NOT RESERVED-WORD AND     110,121
              NOT ARRAY                                        123
375         MOVE
                'ONLY RESERVED WORDS MAY APPEAR IN COL 1. ASSUME
                ' COL2.' TO TEXT1A                              44
376         PERFORM DIAG-END                                 1381
377         IF SWITCH2 NOT GREATER THAN 0                      78
378           GO TO NO-DATA-TYPE.                            1308
379       IF ARRAY AND SWITCH2 EQUAL TO 0                   123,78
380         GO TO NO-DATA-TYPE.                              1308
381       IF RESERVED-WORD AND FLAG2 NOT EQUAL TO 1         121,110
382         PERFORM DIAG-RES-WORD-NOT-COL1 THRU              1309
                DIAG-EXIT.                                   1386
383       IF PROC-DIV                                        126
384         GO TO FINISH-DATA-DIV.                           485
    *
385       IF NUM-DATA                                        124
386         GO TO PROCESS-NUM-DATA.                          413
    *
388       IF ARR[illegible]ND SWITCH2 EQUAL[illegible]OR ARRAY AND  123,78,1[illegible]

          ELSE
361         IF SWITCH2 EQUAL TO 2                            78
362           GO TO PROCESS-NUM-DATA-ARRAY100                436

A-85 77 SEQ-NUMBER                  PIC 999          VALUE 0.      325C,331C,332U,794U
```

Note two-column Procedure Division listing.

❼ Data items.

❽ Footnoted data item.

❾ Paragraph Gone to (G) and Performed (P) by referenced statements.

❿ Nested IFs indented progressively.

⓫ Footnote.

⓬ Procedure Division statements referencing this data item.

Thus, when reading the Data Division, the user can identify immediately the procedure division statements that read, write, or inspect a given data item. File descriptions are comprehended quickly because of uniform indenting conventions that are imposed by the Lister option. When reading the Procedure Division, the user can see references to statements in the Data Division, showing use of the data item, and also to other statements in the Procedure Division, simplifying the tracing of program logic. The lister option further facilitates the tracing of program logic by printing the Procedure Division in multi-column format, so that fewer page turnings are required, and more logic appears on a page. Optionally, the user may obtain a new source deck that reflects the reformatted source listing, with the exception of embedded cross-reference information.

The Lister option also produces a summary listing that contains selected statements from the source program, plus a condensation of the detailed information from the reformatted source listing. The total number of each type of reference for each File Description, and for each Section in the Procedure Division is shown.

Performance Considerations: Although use of the Lister option necessarily adds time to the compilation run, the lister option does not alter the source code. Therefore, such source code takes no longer to compile than if the Lister option had not been invoked. However, to save time on initial compilations, the lister option should not be invoked until the COBOL source program is known to be substantially free of syntax errors. Once an up-to-date reformatted source listing and new source deck reflecting the listing is obtained, the Lister option can be omitted on subsequent compilations. When large COBOL programs are to be listed and compiled, the user may be able to obtain lower run times by electing to use Lister cross reference information in place of XREF or SXREF.

SORT-OPTION Clause (Release 2 Only)

The SORT-OPTION clause gives the COBOL programmer greater control over operations of the Sort/Merge program. In the sort-file-description entry, the SORT-OPTION clause specifies a Working-Storage area which at object time will contain an OPTION control statement for the Sort/Merge program (Program Product 5746-SM1). Options not specified default to the Sort/Merge defaults and not to the COBOL program defaults.

Verb Profiles (Release 2 Only)

The verb profiles option produces a list of all verbs contained in the COBOL source program. Each different COBOL verb in the program is shown, followed by a count representing the number of times it appears in the source program. Optionally, the count is followed by the associated statement numbers. Figure 3 gives an example of Verb Profile/Statistics output.

```
VERBS                          OCCURS     REFERENCE

CLOSE                          000001     000081
GENERATE                       000001     000078
GO                             000002     000077   000079
INITIATE                       000001     000069
OPEN                           000001     000068
READY                          000001     000067
RESET                          000001     000074
RETURN                         000001     000077
SORT                           000001     000070
STOP                           000001     000075
TERMINATE                      000001     000080


[1]Statement number references can be requested optionally in
 addition to the verb count.
```

Figure 2. Example of Verb Profile Output

Symbolic Debug

This feature reduces program development time by making debugging information available in a COBOL format instead of a hexadecimal dump format. No source language changes are needed; the debugging information is requested through object-time control cards. Thus, multiple debugging runs can be made without recompilation.

When a program terminates abnormally, the user receives a COBOL-formatted map of his Data Division. Each data area is identified by its source name, and its contents are easily readable. The user can also request snapshots of the Data Division at any point during program execution.

If two or more COBOL programs are link-edited together and one of them terminates abnormally, the user is provided with such COBOL-formatted information for the program causing termination and its callers, up to and including the main program. Abnormal termination information is provided in the following parts:

1. Abnormal termination message, including the number of the statement and of the verb being executed at the time of abnormal termination.
2. Selected areas in the Task Global Table.
3. COBOL-formatted map of the Data Division, including:
   a. for SD's, the sort or merge record.
   b. for FD's, the data record (and, for VSAM files, the file status).

c. for RD's, the report line, page counter, and line counter.

d. Working-Storage.

e. The Linkage Section.

A dynamic map can be requested at any point in the Procedure Division. If such a map is requested at a STOP RUN, EXIT PROGRAM, or GOBACK statement, an "end-of-job" map results.

No source language changes are needed to use this feature. At compile time, an option tells the compiler to create a debug file, an additional data set which contains descriptions of data items (the dictionary) and other debugging information. At object time, symbolic debug output is requested through control cards. An additional work file, SYS005, is required at compile time and object time.

Several COBOL programs link-edited together must have separate debug files if they are compiled with the SYMDMP option.

Performance Considerations: When this feature is used, optimized object code cannot be requested.

Load module size is increased by 3 factors, although in a virtual system, additional real storage for these elements is not needed:

1. Space needed for the SYMDMP routine.
2. Additional inline instructions in the object program for each branch out of the object code main line (such as branches to object-time subroutines, data management routines, etc.).
3. Data and table space, which depends on the number of SYMDMP control cards specified.

Figure 3 gives portions of a sample program compiled using the SYMDMP option, and shows an example of the abnormal termination map that can be requested.

Flow Trace

The flow trace option allows the user to receive a formatted trace (that is, a list containing the program identification and statement numbers) corresponding to a variable number of procedures executed prior to an abnormal termination. The number of procedures to be traced is specified by the user. The flow trace option requires no source language changes.

A flow trace is printed only in the event of an abnormal termination. The number of procedures to be traced may be specified at compile time or execution time.

Figure 3. Example of Symbolic Debugging Output

Portions of TESTRUN source program, compiled using the Symbolic Debugging Feature

```
                                .
                                .
                                .
00038   000370   WORKING-STORAGE SECTION.

                                .
                                .
                                .
00055   000530   01  RECORDA.
00056   000535       02 A PIC S9(4) VALUE 1234.
00057   000540       02 B REDEFINES A PIC S9(7) COMPUTATIONAL-3.
                                .
                                .
                                .
00059   000550   PROCEDURE DIVISION.
                                .
                                .
                                .
00066   000620   STEP-1.  OPEN OUTPUT FILE-1. MOVE ZERO TO KOUNT, NUMBR.
00067   000630   STEP-2.  ADD 1 TO KOUNT, NUMBR.
00068   000640       MOVE ALPHA (KOUNT) TO NO-OF-DEPENDENTS.
00069   000650       COMPUTE B = B + 1.
                                .
                                .
                                .
```

Field B does not contain valid COMPUTATIONAL-3 data.

Therefore, source statement 00069 is in error.

Portions of symbolic formatted map produced at abnormal termination.

Message giving source statement and verb number causing abnormal termination.

Portion of Data Division map

Data present in RECORDA.

Fields A & B. (Invalid numeric positions in field B shown as asterisks (*).

```
                                                    COBOL ABEND DIAGNOSTIC AIDS

INTERRUPT CODE 7   LAST PSW ADDR BEFORE ABEND E00044A2

PROGRAM TESTRUN

LAST CARD NUMBER/VERB NUMBER EXECUTED -- CARD NUMBER 000069/VERB NUMBER 01.
                                .
                                .
                                .
        000055  01  RECORDA                                  (HEX)       F1F2F3C4
003E40                                                            ND-OT   +1234
003E40  000056  02  A                                            NP-S    +*1*2*3*
003E40  000057  02  B
```

ND-OT = numeric display overpunch sign trailing

NP-S = numeric packed decimal–signed

Note: In the complete map, an explanation of the data codes used, and selected areas of the TGT are printed after the statement number message and before the Data Division map.

Performance Considerations: When the feature is used, optimized object code cannot be requested. An additional 6 bytes are generated for each procedure-name in the program. A trace table is built; its length is 80 bytes plus 4 times the number of procedures specified for the table. Execution time is increased by the time necessary to process these instructions and to build the trace table. Library modules of approximately 2700 and 1200 bytes are needed. The 2700-byte routine may be shared with the Statement Number Option. In a virtual system, additional real storage is not needed for these elements.

Statement Number Option

This option facilitates debugging by providing information about the statement being executed at the time of abnormal termination. At abnormal termination, the statement number is printed; if there are two or more verbs in the source statement, the verb being executed is identified. The program containing the statement is also identified. This option requires no source language changes.

Performance Considerations: When this feature is used, optimized object code cannot be requested. There are 5 additional bytes generated for each procedure-name in the program, and 5 to 17 for each verb.

Execution Time Statistics (Release 2 Only)

The execution time statistics option provides the programmer with information that aids user program optimization by identifying heavily-used portions of the COBOL source program. It is also useful to the programmer in debugging by providing verification that all portions of a program have been executed. During execution of the object program, a count is maintained for each verb in the source program. Just prior to program termination, the system prints the accumulated execution count showing the verb, the name of the procedure in which it appears, and the number of the statement in which it appears. Figure 4 gives an example of Verb Profile/Statistics output.

WHEN-COMPILED Special Register

The WHEN-COMPILED special register makes available to the object program the date-and-time compiled constant carried in the object module. WHEN-COMPILED provides a means of associating a compilation listing with both the object program and the output produced at execution time.

STATEMENT	PROCEDURE NAME	L[1] B	VERB ID	VERB COUNT	PERCENT
119	PAGE-HEAD-RTN				
120			USE		
121	PAGE-HEAD-RTN-SWITCH				
122			GO	6	2.343
123	PAGE-HEAD-RTN-TEST				
124			IF	5	1.953
124			MOVE	2	.781
125			ELSE		
125			MOVE	3	1.171
126			MOVE	3	1.171
127			GO	5	1.953
128	PAGE-HEAD-RTN-ALTER				
129			ALTER	1	.390
130	PAGE-HEAD-RTN-SUPPRESS				
131			MOVE	1	.390
132	PAGE-HEAD-RTN-EXIT				
133			EXIT	6	2.290
136	OPEN-FILES				
136			OPEN	1	.390
137			INITIATE	1	.390
138	READATA				
139			READ	73	28.515
139			GO	1	.390
140			GENERATE	72	28.125
141			GO	72	28.125
142	COMPLETE				
143			PERFORM	1	.390
144			TERMINATE	1	.390
145			CLOSE	1	.390
146		*	STOP	1	.390

[1] Last Block (this column contains an asterisk (*) for the last block in each subroutine)

VERB	STATIC COUNT	DYNAMIC COUNT	VERB EXECUTIONS	PERCENT
ALTER	1	1	1	.390
CLOSE	1	1	1	.390
EXIT	1	1	6	2.343
GENERATE	1	1	72	28.125
GO	4	4	84	32.812
IF	1	1	5	1.953
INITIATE	1	1	1	.390
MOVE	4	4	9	3.515
OPEN	1	1	1	.390
PERFORM	1	1	1	.390
READ	1	1	73	28.515
STOP	1	1	1	.390
TERMINATE	1	1	1	.390

Figure 4. Examples of Execution Statistics Output

Detailed Data and Procedure Division Information

Additional information is provided when the Condensed Listing (CLIST) or Symbolic Map (SYM) compiler options are specified. Global tables, literal pools, register assignments, PBL assignments, and Working-Storage location and size (in hexadecimal notation) are all given in addition to the information previously provided with these options.

Optimized Object Code

With this feature the user can obtain optimized object code as output from the DOS/VS COBOL Compiler. When this feature is requested, new phases of the Compiler generate more efficient code, considerably reducing the amount of space required by the object program.

The code thus produced is considerably better (with respect to object program space) than that produced by Version 2 of the American National Standard COBOL Compiler. The time required to execute this code is equal to or less than the time required to execute object code produced by Version 2 of the American National Standard COBOL Compiler.

The optimized object code feature is requested at compile time via the CBL card. This feature may not be used if the symbolic debug feature, the statement number option, or the flow trace option is requested.

Performance Considerations: Excluding user's data, the space saved as compared with Version 2 can be up to 30% of the generated code. The percentage of space saved is dependent on the nature of the program. For small programs with few data areas the savings will be small. As a general rule, however, the space saved increases with the number of referenced paragraph-names and branches, and the number of 01-level data-names in the program.

System/370 Instruction Generation

System/370 instructions are provided automatically by DOS/VS COBOL. (When IBM-360 is specified in the OBJECT-COMPUTER paragraph, the compiler generates System/370 instructions, and issues a warning message.) The Compiler generates instructions from the System/370 set, including Move Long (MVCL), Compare Logical Long (CLCL), and Shift And Round Decimal (SRP) that are particularly useful to COBOL. These System/370 instructions replace object-time subroutines and instructions that the Version 3 Compiler generates under System/360 including routines and instructions to handle decimal arithmetic scaling (where operands have a different number of decimal places) and rounding. System/370 support also gives much improved processing of variable length fields.

Since System/370 does not require boundary alignment for COMPUTATIONAL, COMPUTATIONAL-1, and COMPUTATIONAL-2 items, no moves are generated for items that are not SYNCHRONIZED.

Performance Considerations: Space occupied by a DOS/VS COBOL program is decreased, particularly when calls to object-time subroutines are no longer necessary. Such calls are always generated in System/360 for variable-length moves and comparisons. If there is at least one variable-length alphanumeric move in the source program, System/370 support reduces the size of the object program by at least 484 bytes; if there is also at least one variable-length alphanumeric comparison, System/370 support reduces the size of the object program by at least an additional 498 bytes.

Table 1 gives comparative figures, without right justification, for fixed-length and variable-length MOVE statements, and for fixed-length and variable-length comparisons.

Table 2 gives comparative figures for Shift And Round Decimal generation; the savings shown are made for each such operation in the object program.

Number of Bytes in Each Move or Comparison	*For Each Alphnumeric Move: Object-program Instructions*		*For Each Comparison (in a conditional expression): Object-program Instructions*	
	System/360 Bytes Needed	*System/370 Bytes Needed*	*System/360 Bytes Needed*	*System/370 Bytes Needed*
variable length	26 + 480*	14-22	26 + 496*	16-24
fixed length				
1-256	6-16	6-16	8-26	8-26
257-512	12-22	12-22	16-36	16-24
513-768	18-28	14-22	24-46	16-24
769-1024	24-34	14-22	32-56	16-24
1025-1280	30-40	14-22	40-66	16-24
1281-1536	36-46	14-22	48-76	16-24
...	...	...	...	...
>4096	26 + 480*	14-22	26 + 496*	16-24

*Bytes needed to invoke object-time subroutine, plus size of subroutine itself.

Table 1. Alphanumeric Moves and Comparisons—System/360 vs. System/370

Function	*System/360 Bytes Needed*	*System/370 Bytes Needed*
Rounding	39 + literal*	6
Left Scaling	6 + literal*	6
Right Scaling	12	6

* As used for decimal point alignment; the literal varies in length with size of data-item, number of decimal positions defined, and/or scaling positions defined.

Table 2. Shift And Round Decimal (SRP)—System/360 vs. System/370

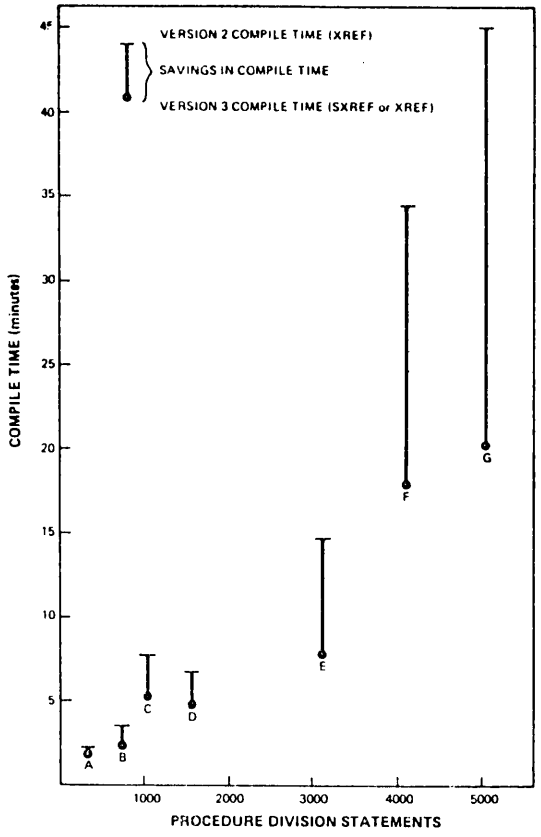
Speedy Sorted Cross-Reference

If an alphabetized cross-reference list is requested, the DOS/VS COBOL compiler produces a cross-reference dictionary in which data-names, file-names, and procedure-names are sorted alphanumerically into two groups. One group consists of data-names and file-names; the second consists of procedure-names. Each group is preceded by a subheading.

This option is requested at compile time via the CBL card.

Performance Considerations: Although actual figures are not the same, performance improvement for DOS/VS COBOL is in the same order of magnitude as for DOS Full American National Standard COBOL, Version 3, since the technique used is the same. Version 3 alphabetized cross-reference (SXREF) performs up to 25 times faster than previous source-ordered cross-reference (XREF). As a result, when SXREF is requested, compilation can be up to 2 times faster, depending on the options chosen, than Version 2 compilation using source-ordered XREF. Version 3 XREF is similarly improved in performance over Version 2. Figure 5 gives examples of actual test cases.

Note: A more comprehensive reformatted listing of the entire COBOL source program can be obtained by invoking the Lister option. The listing thus produced contains *embedded* cross-reference information and simplifies tracing program logic. (See the "Lister" section.)



Other Options in Effect.

```
// OPTION LOG,NODUMP,NOLINK,NODECK,NOLIST,NOLISTX,NOSYM,ERRS
  CBL BUF=256,SEQ,FLAGW,SPACE1,APOST,TRUNC,ZWB,PMAP=0,NOOPT
  CBL FLOW=99,NOSTATE,NOCLIST,NOSUPMAP
```

Compilers Used: Version 3 Release 2.1 PTF 7
Version 2 Release 27.1 PTF 9

System Used: 500 MPS

Source Program Sizes:

Test Case	*Source Cards*	*Data Division Entries*	*Procedure Division Statements*
A*	549	188	322
B	953	209	720
C*	2177	908	1032
D	2012	368	1586
E	4229	1003	3110
F*	7426	3195	4090
G*	8319	3195	5065

*programs marked with an asterisk use maintenance functions, so that CBL LIB was used.

Figure 5. SXREF/XREF Performance Comparisons — Version 3/Version 2

IBM DOS/VS COBOL Subroutine Library

COBOL library subroutines perform operations requiring extensive coding. For this reason, it is inefficient to place this coding in the object module produced by the DOS/VS COBOL Compiler each time it is needed. The COBOL object-time subroutines, now packaged as a separate Program Product, are located in the relocatable and core image libraries. Routines required to execute the problem program are combined with the object module produced by the DOS/VS COBOL Compiler at link-edit time or are dynamically fetched during program execution.

There are several major categories into which the Subroutine Library can be divided:

- Input/output
- Conversion
- Arithmetic verbs
- Sort/Merge interfaces
- Checkpoint (RERUN)
- Segmentation Feature
- Debugging
- Other verbs
- VSAM Interface
- 3886 processing

Note that the Subroutine Library, because it is for use with programs compiled by the DOS/VS COBOL Compiler, necessarily contains all subroutines needed to support the improvements provided in the DOS/VS COBOL Compiler. These improvements are described in the section "DOS/VS COBOL Compiler."

The DOS/VS COBOL Subroutine Library is part of the Disk Operating System relocatable and core image libraries. These libraries must reside on a disk storage device. The Subroutine Library is designed for use under the IBM Disk Operating System, with object modules produced by the DOS/VS COBOL Compiler.

IBM DOS/VS COBOL Language Base

DOS/VS COBOL is compatible with the highest level of American National Standard COBOL, X3.23-1968, and with international standard ISO/R 1989-1972 Programming Language COBOL. DOS/VS COBOL is also a subset of the complete CODASYL description of the COBOL language as documented in *CODASYL COBOL Journal of Development.* IBM-specified language capabilities are also implemented. DOS/VS COBOL supports all COBOL features supported by DOS Full American National Standard COBOL.

The following functional description of the DOS/VS COBOL implementation indicates which items are *Standard COBOL,* which items are *CODASYL-Specified,* and which items are *IBM-Specified.* The descriptions are grouped according to processing function.

Those items not available in previous versions of IBM DOS COBOL are identified as follows:

[1] not available in COBOL D

[2] not available in DOS Full ANS COBOL Version 2

[3] not available in DOS Full ANS COBOL Version 3

Basic Internal Processing (Nucleus)

Allows the user to specify internal data processing within the framework of the four COBOL program divisions.

Identification Division

Standard COBOL: The programmer can name the COBOL program and write comment entries. He can request that the actual compilation date[1] be printed in the source listing (via the DATE-COMPILED paragraph).

Environment Division

Standard COBOL: The programmer can document source and object computer configuration. IBM function-names with a fixed meaning can be equated to user-specified mnemonic-names. Switches can be specified for use with the switch-status condition. Another character may be substituted for the dollar sign, [1] and the functions of the decimal point and comma in PICTURE strings and numeric literals can be interchanged.

Data Division

Standard COBOL: The programmer can describe the data to be used in the program. Data can be hierarchically grouped in records, or it can consist of independent noncontiguous data items. Data items can be assigned a specific size and value, can be described as alphabetic, alphanumeric, or fixed-point numeric (zoned decimal or binary), as being aligned upon necessary boundaries, as having alternate attributes, and as being regrouped for specific processing needs. [1]

IBM-Specified Items: Numeric items may be defined as internal decimal, as internal or external floating-point, or as having a sterling currency format.

Procedure Division

Standard COBOL: A simple or nested conditional statement allows a simple or compound condition to be tested. (Conditions allowed are: class, condition-name, relation, sign, switch-status.) Arithmetic operations allowed are: addition, subtraction, multiplication, division, exponentation, or a combination of any or all; multiple receiving fields[1] are allowed in addition and subtraction operations. Unconditional and conditional branching statements and program switches are allowed. Data can be moved from one storage location to another, and a data item can be inspected to count and/or replace single characters or groups of characters within it. Low-volume data can be moved into or out of the computer.

Sequential Access

Allows the user to process records in the physical order in which they are placed in the file.

Environment Division

Standard COBOL: The programmer can name the file and document the device upon which it resides; he can specify sequential access for the file as well as any shared memory areas and checkpoint instructions.

IBM-Specified Items: Indexed files can be specified, and access can be sequential.

Data Division

Standard COBO The FD entry allows the user to specify fixed or variable length physical a logical records, the presence of system standard, user standard, or non dard labels, to document the contents of labels, and to

identify records associated with the file.

IBM-Specified Items:[1] The programmer can specify the recording mode for the file.

Procedure Division

Standard COBOL: The programmer can create, retrieve, and update a sequential file; he can create and retrieve a direct file. When an input/output error occurs, special routines can be executed. Label Declarative procedures to process user standard or nonstandard labels can be specified.[1]

IBM-Specified Items: The programmer can create, retrieve, and update an indexed file. Processing can begin at any record,[1] or class of records,[1 2] in the file.

Random Access

Allows the user to process records in a mass storage file by means of a programmer-specified key. The order of processing is the order in which keys are presented.

Environment Division

Standard COBOL: The programmer can name the file and document the device upon which it resides; he can specify random access for the file, as well as any shared memory areas and checkpoint instructions.

IBM-Specified Items: Indexed files can be specified, and access can be random.

Data Division

Standard COBOL: The FD entry allows the user to specify fixed or variable length physical and logical records, the presence of system standard or user standard labels, to document the contents of such labels, and to identify records associated with the file.

IBM-Specified Items:[1] The programmer can specify the recording mode for the file.

Procedure Division

Standard COBOL: The programmer can create, retrieve, update and add records to a direct file. When an input/output error occurs, special routines can be executed. Label Declarative procedures to process user standard labels can be specified.[1]

IBM-Specified Items: Through a search key the programmer can retrieve, update, or add records to an indexed file.

Table Handling

Allows the user to define and process fixed or variable length tables of up to three dimensions. Subscripting (giving the ordinal position of the table entry) and indexing (giving the displacement of the table entry)[1] are provided.

Data Division

Standard COBOL: One elementary variable-length table is allowed. A table may be ordered on some ascending or descending key.[1] Data items can be defined to hold index values without conversion.[1]

IBM-Specified Items:[1] Three nested levels of variable-length tables can be defined.

Procedure Division

Standard COBOL:[1] Sequential or binary search of a table is provided. Index-name values can be initialized before a table search begins, and can be equated to literals, data-names or other index-name values.

Sort [1]

Allows the user to order a file of records one or more times according to the ascending/descending order of embedded key(s) in each record. More than one file may be sorted.

Environment Division

Standard COBOL: Names and identifies the sort file, documents the hardware devices to be used. Shared memory areas can be specified.

IBM-Specified Items: Checkpoints can be taken during a sorting operation.

Data Division

Standard COBOL: The SD entry specifies the logical sort record size and the names of the records to be sorted.

IBM-Specified Items: The recording mode of the records to be sorted can be specified. Sort work file label information is accepted.

Procedure Division

Standard COBOL: The programmer can specify that records are to be accepted from an input file, sorted, and placed on an output file. Input and/or output records can be modified before/after sort processing.

IBM-Specified Items: Control can be passed outside of the sort procedures during modification of input and/or output records. Sort special registers allow optimization of the sort processing.

Report Writer [1]

Allows the user to specify the format and logic of a printed report in the Data Division, thus minimizing the Procedure Division coding necessary. Detail and/or Summary reporting may be specified.

Standard COBOL: The RD entry describes the physical aspects of the report, including identifying code, control data items and page format. The report group description entry identifies a report group and defines its format (line and column), and characteristics: whether it is a heading, detail, or footing item, if it is an item used as a source or as a sum counter, when it is to be reinitialized, etc. Procedure Division statements specify that report data items are to be initialized, that report processing (detail or summary) is to be performed, and that report processing is to be ended. Reporting Declaratives for special processing situations can be specified. One report may be written to two different files.

IBM-Specified Items: Report group printing can be suppressed at specific points during program execution.

Segmentation [1]

Allows the user to specify object program overlay requirements through specification of segment numbers for Procedure Division sections.

Standard COBOL: All sections with the same segment number constitute one program segment. Segments with different segment numbers may be intermixed in the Procedure Division. Permanent segments that cannot be overlaid and permanent segments that can be overlaid (both always made available in their last-used state) are provided. Independent segments (always made available in their initial state) can overlay or be overlaid by other segments. In the Environment Division, the programmer can specify those permanent segments which can and those which cannot be overlaid by independent segments.

Source Program Library

Allows the user to specify text that is to be copied into a source program from a library.

Standard COBOL: Library text, with word substitution in the text to be copied,[1] can be copied into the COBOL source program. Text may be: Configuration Section paragraphs, Input-Output Section paragraphs, FD and SD[1] entries, record description entries, Procedure Division sections and paragraphs.

IBM-Specified Items:[1] Copied text need not appear on the source listing. Complete programs copied from a library can be used as input for a compilation; statements can be added and/or deleted before compilation begins.

VSAM Mass Storage Processing[1, 2, 3]

Allows the user to process records sequentially or randomly in a VSAM mass storage file. Records can be placed in the file in physical sequential order, or in the ascending order of an embedded record key, which contains an unchanging unique value.

Physical sequential files must be accessed sequentially. Indexed files can be accessed sequentially in order of ascending record keys, and randomly through the value of each record key presented.

Environment Division

CODASYL-Specified Items: The programmer can name the file and document the device upon which it resides; for physical sequential files he can specify sequential access; for indexed files he can specify sequential, random, or dynamic access. For indexed files he also specifies the record key. For both types of files, he specifies file organization, buffer allocation, shared memory areas between files, and checkpoint instructions.

IBM-Specified Items: A password can be defined.

Data Division

CODASYL-Specified Items: For both sequential and indexed files, the FD entry allows the user to specify fixed or variable length logical records. Other clauses of the FD entry are accepted and treated as comments.

Procedure Division

CODASYL-Specified Items: For a sequential file, the programmer can create, retrieve, update, and add records in the file. For an indexed file, the programmer can create, retrieve, update, add, and delete records in the file; he can specify that sequential processing is to begin at some specific record or at some class of records. For both types of files, special routines can be executed when an input/output error occurs.

IBM-Specified Items: If a password has been defined, the file cannot be opened unless a valid password is present.

MERGE [1, 2, 3]

Allows the user to combine two or more identically sequenced files according to the ascending/descending order of embedded key(s) in each record. Up to 8 files may be combined in a single merge operation.

Environment Division

CODASYL-Specified Items: Names and identifies the merge file.

Data Division

CODASYL-Specified Items: The SD entry specifies the logical merge record size and the names of the records to be merged.

IBM-Specified Items: The recording mode of the records to be merged can be specified. Merge work file label information serves only as documentation.

Procedure Division

CODASYL-Specified Items: The programmer can specify that records are to be released from input files, merged, and placed on an output file. Output records can be modified after merge processing.

IBM-Specified Items: Control can be passed outside of the merge procedures during modification of output records. Certain sort special registers allow optimization of merge processing.

Advanced System/370 Device Processing[1, 2]

Allows the user to exploit the processing capabilities of the advanced System/370 devices.

Standard COBOL: Through standard COBOL language combined function processing for the System/370 card devices can be specified. Read/punch/print, read/punch, read/print, and punch/print functions can be specified.

IBM-Specified Items: Through standard COBOL source language and object-time control cards, Optical Mark Read processing for the 3505 and Read Column Eliminate Processing for the 3505 and 3525 can be specified. For the Optical Character Readers, predefined object-time library subroutines provide the following functions: opening and closing the file, reading, checking, controlling the document, and loading a new format record. After each request, a return code is passed back to the program so that any exceptional condition can be tested. Options are available to specify error and end-of-file processing.

Subprogram Linkage

Allows communication between object programs in one run unit.

CODASYL-Specified Items: Data can be shared between calling and called programs. Control can be transferred from a calling program to a called subprogram.

IBM-Specified Items: Alternate entry points into a called subprogram are allowed.

Debugging Facilities

Allows the user to describe conditions under which data items and/or procedures are to be monitored during object program execution.

IBM-Specified Items: A formatted display of selected data items or literals at specific points of program execution can be requested either conditionally or unconditionally. When a procedure is entered, an indication can be requested. A debugging procedure can be written at the end of the source program for insertion before any section or paragraph in the Procedure Division. Selective execution of a debugging statement may be requested depending on a count condition.

Appendix A: Other IBM COBOL Program Products

The following brief descriptions outline the features and functions of other IBM COBOL Program Products. There are Program Product COBOL Compilers for a wide range of systems. There are also other Program Products related to COBOL which make the compilation and debugging of IBM OS COBOL programs far quicker and easier than before.

Note: A Type 1 (no fee) Language Conversion Program (LCP) is available which converts COBOL D, E, and F programs to American National Standard COBOL. IBM DOS American National Standard COBOL is upwardly compatible with IBM DOS/VS COBOL — that is, any IBM DOS American National Standard COBOL program (Subset or Full) will run under DOS/VS without change.

1130 COBOL Compiler & Library

Program Number

5711-CB1

Features

- Spanned Records
- COPY Useable in 3 Divisions
- 3 Levels of Indexing, Subscripting
- Abbreviations Extensively Permitted
- Extensive Argument Passing CALL Statement
- Improved Compile Operations with
  1) Larger CPU (up to 32K words)
  2) 1442 Card Punch if 2501 Card Reader installed instead of Reader/Punch
  3) Additional 2310 Disk Drives
- Improved Execution Operations with the above plus a Second Printer
- Load on CALL, System Overlay on CALL
- Special Service Subroutines

Note: 1130 COBOL does not include

- ISAM
- Packed Decimal
  (except via service subroutine)
- Floating Point
- TRANSFORM
- Declaratives
- Segmentation

(All are included in DOS Subset ANS COBOL)

Debugging Aids

READY TRACE
RESET TRACE
Source Statement Number Retention
Callable Snapshot Dump

System Requirements

1130 Disk Monitor System Version 2

Compiler Storage Requirements

8K words

Data Set Compatibility

Any 1130 RPG sequentially created file
1130 FORTRAN files
(converted by service program)

Device Support

1132 Printer
1403 Printer 120-132 Print Positions
1442 Card Read/Punch
2501 Card Reader

Of Interest To

New Users

Availability

NOW

General Documentation

General Information GH20-0799

System/3 COBOL Compiler & Library

Program Number

5702-CB1

Features

Upward Compatible to DOS and OS COBOL
- Superset of 1130 COBOL: includes all 1130 Language Features
- Lowest Level Nucleus plus Extensions in Nucleus, Sequential and Random Access
- Extensive Options
- CODASYL Extensions
- Comprehensive and Precise Diagnostics
- ISAM
- Segmentation
- Table Handling
- Extensive COPY Usage
- Accepts 80 or 96-column Card Input
- Separate Sign Feature
- Multi-Function Card Processing

Note: System/3 COBOL does not include
- Floating Point
- TRANSFORM
- Declaratives

(All are included in DOS Subset ANS COBOL)

Debugging Aids

READY TRACE
RESET TRACE
EXHIBIT

System Requirements

Model 10 Disk System Control Program
Release 6

Compiler Storage Requirements

12K

Data Set Compatibility

Any System/3 file created through Disk System Management including RPG
80 column 1130 or System/360 card files

Device Support

Additional Models of 5444 Disk and 5203-1403 Printers.
5424 MFCU
1442 Card Read/Punch
5471 Console
5445 Disk
3410/3411 Tape

Of Interest To

New Users
Users Moving from 1130 COBOL to System/3
DOS Users (mixed shop with System/3)

Availability

Release 3 NOW

General Documentation

General Information GC28-6453

DOS Subset ANS COBOL Compiler & Library

Program Number

5736-CB1

Features

Highlights:
- Segmentation
- Double buffered ISAM
- Table Handling with Indexing

Comprehensive, Precise Diagnostics
Expanded Output Listings
- Assembler mnemonics in LISTX
- Expanded Data Division Map
- Comments in every Division

Expanded Function
- Spanned Records
- ADD, SUBTRACT & MOVE CORRESPONDING
- COPY Allowed in All Divisions
- Relative Track Addressing
- RENAMES
- VALUE THRU, VALUE IS series
- Separate Signed Numeric Data
- CICS

Compiler Design and Performance Improvements
- SYSLINK & Deck in one Run
- PROGSIZE Option

(See DOS Full ANS COBOL for items not included)

Debugging Aids

READY TRACE	EXHIBIT
RESET TRACE	Sorted XREF

System Requirements

DOS	Release 24
Double-buffered ISAM, 2319, 3211	Release 25
3330, 3410, 3420, 3505, 3525	Release 27

Compiler Storage Requirements

20K

Data Set Compatibility

DOS Full ANS COBOL Versions 2 & 3
COBOL D

Device Support

3211 Printer	3410, 3420 Tape
2319 Disk	3505 Reader (OMR)
3330 Disk	3525 Card Punch

Of Interest To

COBOL D
Small Storage or Multiprogramming User
S/370 user

Availability

Release 2	NOW
3881, 2560	6/73
5425	12/73

General Documentation

General Information GC28-6402

DOS Full ANS COBOL Compiler & Library Version 3

Program Number

5736-CB2 Compiler 5736-LM2 Library

Features

All DOS Subset ANS COBOL Features, plus:
- Sort Feature
- Report Writer
- Full Table Handling
  - SEARCH verb
  - 3-level variable-length tables
- Full Segmentation
  (overlayable fixed segments)
- Full Source Program Library
  (REPLACING option)
- Extended Source Program Library
  (BASIS/INSERT/DELETE cards)

Version 3 offers the following improvements over Version 2:
- Optimized Object Code
- Move and Compare Improvements
- Debugging Features Listed above

Debugging Aids

All DOS Subset ANS COBOL items, plus:

Batch Symbolic Debug (COBOL-formatted Data Division snapshots and listings)
WORKING-STORAGE location and size
Flow Trace
Statement number
Expanded CLIST & SYM
Relocation factor

System Requirements

DOS	Release 25
2319, 3211 3330, 3410, 3420, 3505, 3525	Release 27
3504, 2560, 3881	Release 28

Compiler Storage Requirements

54K

Data Set Compatibility

DOS Full ANS COBOL Version 2
ANS Subset COBOL
COBOL D

Device Support

3211 Printer
2319 Disk
3330 Disk
2560 Multifunction Card Machine (MFCM)
3410, 3420 Tape
3504, 3505 Reader (OMR/RCE)
3525 Card Punch (RCE)
3881 Optional Mark Reader (OMR)

Of Interest To

DOS ANS Version 2
COBOL D Large Storage Users
S/370 user

Availability

NOW

General Documentation

General Information GC28-6421

OS Full ANS COBOL Compiler & Library Version 3

Program Number

5734-CB1

Features

Improvements over Version 2
- Time Sharing Option Support
- OPEN Statement Improvement
- Move & Compare Improvement
- Optional Allocation of Compiler Data Sets
- Batch Compilation
- Separate Signed Numeric Data
- Dynamic Record Length Specified
- Generic Keys for Indexed Files
- RERUN Facility at End-of-Volume
- ON Statement Improvement
- Error Declarative Enhancement
- Debugging Features listed above
- Sort interface improvements

Debugging Aids

Sorted XREF	Flow Trace
WORKING-STORAGE Location & Size	Statement Number
	Expanded CLIST & DMAP

System Requirements

	OS	OS/VS
Basic Support	Rel. 19	VS1, VS2
ASCII	Rel. 20.1	VS1, VS2
TSO	Rel. 20.1	VS2

Compiler Storage Requirements

80K MFT
86K VS1, MVT
92K VS2

Data Set Compatibility

OS Full ANS COBOL Version 2
COBOL E & F

Device Support

3211 Printer	2305 Fixed Head Storage
2319 Disk	3330 Disk

Of Interest To

OS Full ANS COBOL Version 2 users moving from DOS to OS COBOL E & F	All TSO users
	System/370 users

Availability

Release 3 NOW

General Documentation

General Information GC28-6407

OS Full ANS COBOL Compiler & Library Version 4

Program Number

5734-CB2 (Compiler & Library) 5734-LM2 (Library only)

Features

All Version 3 items plus:
- Optimized object code
- Library management facility (reentrant object library)
- Teleprocessing facility (TCAM)
- Dynamic subprogram linkage
- Syntax-checking compilation
- String Manipulation
- FBS (fixed block standard) support (improves 3330 performance)
- Supports COBOL Interactive Debug (see reverse side)

Debugging Aids

All OS ANS COBOL Version 3 items, plus:
Batch symbolic debug (COBOL-formatted dynamic Data Division snapshots and listings)
Syntax-checker
Associated COBOL Interactive Debug Program Product

System Requirements

	OS	OS/VS
Basic Support	Rel. 19	VS1, VS2
ASCII	Rel. 20.1	VS1, VS2
TSO	Rel. 20.1	VS2
Teleprocessing (TCAM)	Rel. 21.6	VS1, VS2

Compiler Storage Requirements

80K VS1, MFT, MVT
86K VS2

Data Set Compatibility

OS Full ANS COBOL Versions 2 & 3
COBOL E & F

Device Support

3211 Printer
2319 Disk
2305 Fixed Head Storage
3330 Disk
3410, 3420 Tape
3505 Reader (OMR/RCE)
3525 Card Punch (RCE)

Of Interest To

OS Full ANS COBOL Versions 2 & 3
Users moving from DOS to OS COBOL E & F
All TSO users
System/370 users

Availability

NOW

General Documentation

Planning Guide GC28-6431

TSO COBOL Prompter

Program Number

5734-CP1

Features

Simplifies Compilation of a COBOL Program through Conversational Method

COBOL Prompter Function

- Accepts Terminal Input
- Analyzes Input for Completeness and Correctness
- Prompts Terminal User for Required Operands
- Dynamically Allocates Necessary Data Sets
- Builds an Option List and DDLIST for Compiler
- Invokes the OS Version 3 Compiler, Program Product 5734-CB1, which contains the following options
  - Compilation Storage Availability (BUF option)
  - Buffer Storage Allocation (SIZE option)
  - Batch Compilation
  - Debugging Aids
  - Sorted Cross Reference
  - Terminal Output Facility – output data sets can be allocated to the terminal
- Invokes the OS Version 4 Compiler, Program Product 5734-CB2, which contains all features of the Version 3 Compiler, plus:
  - Batch Symbolic Debugging (may be allocated to the terminal)
  - Optimized Object Code
  - Teleprocessing Facility
  - Library Management Facility
  - Dynamic Subprogram Linkage

Dynamic Storage Requirements

80K ANS COBOL Compiler
20K TSO Service Routine
2K TSO COBOL Prompter

Of Interest To

TSO Users under OS or VS2

Availability

NOW

General Documentation

General Information GC28-6454

OS COBOL Interactive Debug

Program Number

5734-CB4

Features

Allows interactive debugging of a COBOL program from a TSO terminal, through dynamic monitoring of object program execution

Via a TSO terminal, a COBOL user can:

- Establish unconditional or conditional breakpoints (suspensions of execution)
- Prespecify automatic actions at breakpoints (that is, display of data, etc.)
- Dynamically display/compare/modify contents of data items
- Resume execution at any user-specified procedure-name
- Trace the path of execution
- Display selected COBOL source statements
- List status of files
- List or remove active breakpoints
- Obtain assistance in entering COBOL Interactive Debug commands (via HELP)
- Dynamically interrupt (via ATTENTION) and enter debugging commands during test session

Dynamic Storage Requirements

Fixed Region Work Space (all code reentrant, may be LPA resident)

COBOL Interactive Debug:		46K
TSO:	Parse	20K

Variable Work Space

COBOL Interactive Debug Dictionary	may be 12K per 500 elementary Data Division entries
Table Space	4K - 6K
Debug Working Storage	6K - 8K
TSO Control Blocks	12K
LSQA	8K

Object Module produced by companion Program Product 5734-CB2 (OS Full ANS COBOL Version 4)

Of Interest To

TSO users – under OS or VS2
OS Full ANS COBOL Version 4 users

Availability

NOW

General Documentation

General Information GC28-6454

Appendix B: IBM DOS/VS COBOL Changes in the Reserved Word List

For DOS/VS COBOL, the DOS American National Standard COBOL Version 3 reserved word list must include the following entries.

The keys preceding the entries, and their meanings, are:

(xa) before a word means that the word is a CODASYL-specified extension to American National Standard COBOL

(xac) before a word means that the word is an IBM-specified extension to both American National Standard COBOL and CODASYL COBOL

(ca) before a word means that the word is a CODASYL COBOL reserved word not incorporated in DOS/VS COBOL

(sp) before a word means that the word is an IBM function-name established in support of the SPECIAL-NAMES function.

(spn) before a word means that the word is used by an IBM Program Product COBOL compiler, but not this compiler

(ca)	ALPHANUMERIC
(ca)	ALPHANUMERIC-EDITED
(ca)	BOTTOM
(spn)	CANCEL
(spn)	CBL
(spn)	CD
(xa)	CHARACTER
(spn)	COMMUNICATION
(xa)	COMP-4
(xa)	COMPUTATIONAL-4
(spn)	COUNT
(spn)	DATE
(spn)	DAY
(ca)	DAY-OF-WEEK
(ca)	DEBUG-CONTENTS
(ca)	DEBUG-ITEM
(ca)	DEBUG-LINE
(ca)	DEBUG-NAME
(ca)	DEBUG-SUB-1
(ca)	DEBUG-SUB-2
(ca)	DEBUG-SUB-3
(ca)	DEBUGGING
(xa)	DELETE
(spn)	DELIMITED
(spn)	DELIMITER
(spn)	DEPTH
(spn)	DESTINATION
(ca)	DISABLE

(ca)	DUPLICATES
(xa)	DYNAMIC
(spn)	EGI
(spn)	EMI
(ca)	ENABLE
(spn)	ESI
(xa)	EXCEPTION
(xa)	EXTEND
(ca)	INITIAL
(ca)	INITIALIZE
(ca)	INSPECT
(spn)	LENGTH
(xa)	MERGE
(spn)	MESSAGE
(ca)	NUMERIC-EDITED
(xa)	ORGANIZATION
(spn)	OVERFLOW
(xac)	PASSWORD
(spn)	POINTER
(ca)	PROCEDURES
(spn)	QUEUE
(spn)	RECEIVE
(ca)	RELATIVE
(xac)	RELOAD
(ca)	REMOVAL
(xa)	REWRITE
(spn)	SEGMENT
(spn)	SEND
(xac)	SERVICE
(xa)	SORT-MERGE
(spn)	SORT-MESSAGE
(xac)	SORT-OPTION
(xa)	START
(spn)	STRING
(spn)	SUB-QUEUE-1
(spn)	SUB-QUEUE-2
(spn)	SUB-QUEUE-3
(xa)	SUPPRESS
(spn)	SYMBOLIC
(sp)	S03
(sp)	S04
(sp)	S05
(spn)	TABLE
(spn)	TEXT
(spn)	TIME
(ca)	TOP

(spn)	UNSTRING
(xac)	WHEN-COMPILED

The following words should be deleted from the list:

CONSTANT
KEYS
LOWER-BOUND
LOWER-BOUNDS
OH
OV
PREPARED
PRIORITY
RANGE
SELECTED

INDEX

IBM DOS/VS COBOL Compiler and
Library General Information
GC28-6473-3

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

GC28-6473-3

Fold and Staple

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
General Products Division
Programming Publishing—Department J57
1501 California Avenue
Palo Alto, California 94304

Fold and Staple

IBM DOS/VS COBOL G.I. Printed in U.S.A. GC28-6473-3

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

1026555460