IBM 360 COBOL CONVERSION

by
Norman H. Leven

There are areas of major difference when converting existing COBOL programs to IBM 360 COBOL (E) programs. This article considers these major areas, which are concerned with program preparation and language differences.

When converting to a third generation IBM System 360, modification is necessary of existing COBOL programs to IBM 360 COBOL (E) programs, using the Full Operating System. Many of the problems inherent in this type of conversion also relate to each of IBM's operating systems and to future released versions of COBOL.

Considerable time can be saved and maximum utilization of personnel accomplished if the converting installation is aware of those problems that others have already experienced in a COBOL conversion. This article reflects lessons learned in several successful conversions to System 360. It covers the pros and cons of alternate approaches to a COBOL system conversion, and the major COBOL language differences created by the IBM System 360 machine characteristics. No attempt is made to specify all of the actions necessary for conversion but rather to note those areas that can create the greatest conversion problems.

In approaching the conversion phase, decisions must be made regarding training of programers in machine language, alternate program testing methods, use of translational and name-cross-referenced programs, availability of test data, storage of test data sets, optimization of programs, use of cataloged procedures, conversion of master file data, and on-line printing capabilities for the testing phase.

Program Training

The ability to read a core dump and to understand data formats and assembly language instructions can be of great help in assisting the COBOL programer to "debug" his converted program. These capabilities will enable the programer to find the exact machine instruction that caused his program error and to identify the format of the data field that the instruction was manipulating at that point.

From this information, the programer can more easily determine the nature of his error. Without these abilities the programer can, at best, limit the source of error to several data fields and a general section of the program. Supplementary programing efforts and machine rerun time will then be necessary to define the source of error exactly.

If the programer has had previous experience in a one-for-one type language, a short introductory course in 360 Basic Assembly Language and core dump format is highly advisable prior to conversion. This basic machine language training will provide extensive savings in programer and machine time during the program conversion phase.

Basic assembly language training for COBOL programers inexperienced in a one-for-one type language consumes an unjustifiable amount of time and effort. These programers should be provided training only in supplemental 360 COBOL programing.

Alternate Program Testing Methods

Alternative testing methods available to the programers that are not trained in the 360 assembly language, are the use of the IBM COBOL debug packet or an installation-written Abnormal Interrupt Interpreting Routine. Use of IBM's debug packet will eliminate recompilation after successful program checkout, but requires extra programing effort, and often requires another computer run to obtain more information. Also, unless extreme care is used, the debugging language procedures will greatly extend computer run time.

An Abnormal Interrupt Interpreting Routine can be written by the installation's system-programers or obtained from another installation. Because of the complex internal structure of the operating system, the modification of an existing routine or conception of a new routine will require a major effort by the system-programer staff. Also, preliminary attempts have not provided a completely acceptable Interrupt Interpreting Routine capable of handling all possible interrupt conditions.

Translational and Name-Cross-Referencing

A duplication of the initial COBOL source program decks should be the first physical step in the conversion phase. The original source decks should then be filed in a safe location for backup protection. Under no conditions, should modification or processing take place with the original source program deck.

A translational-language conversion program (LCP) has been developed by IBM. This program will recode characters to 360 EBCDIC representation, change the environment division to reflect the 360 configuration, and either change or indicate required changes for those COBOL statements that are incompatible with the 360 COBOL language. Although the LCP program will not identify all of the COBOL statements that may cause conversion problems, its use will greatly reduce repetitious examination, and human copying errors.

Concurrently, a name-cross-reference program should be run. This cross-reference program provides a where-used list for each data and paragraph name used in a COBOL program. Programing efforts will be considerably reduced by reference to the cross-reference list, instead of searching through the entire COBOL program listing. However, the cross-reference list should be generated in advance because programers generally will not delay consideration of a program once the "debugging" phase has started.

Availability of Test Data

If the data used to test the original programs is available, it should be used for the initial checkout of the converted programs. The benefits of using the original test data are the time savings resulting from the handling and processing of only small quantities of data and the comprehensive test of the multiple conditions of the program.

If original test data is not available or when the converted program has satisfactorily processed the original data, current actual data should then be used for program checkout. A converted program should not be considered in a production status until it has successfully processed at least two complete sets of historical data. Even these extensive test procedures will not completely ensure the accuracy of the conversion and the occurrence of a small number of errors during initial production runs is still likely.

Storage of Test Data Sets

If small batches of initial test data are available, and if a disk storage space is available, it is suggested that the initial test data sets for each program be loaded onto disk and retained there until successful program checkout. This procedure eliminates the loading, cataloging and deletion of each data set, every time a test run is made.

The possibility of having the data deck scrambled or lost also is significantly decreased by semipermanent disk storage. However, if test data is stored on disk, extra care must be taken when deleting or modifying any of the cataloged data sets. Historical files used for testing, because of their size can quickly consume all available disk space; therefore, historical files should be deleted after each test run.

Optimization of Programs

Though COBOL programs can be optimized to run more efficiently on the 360 by making use of specific characteristics of the hardware and of the 360 COBOL language, it is suggested that optimization of the programs be delayed until all direct conversion steps are complete. The only exception to this is conversion to computational format, of those fields used in mathematical calculation. Concurrent optimization and conversion will propagate increases in computer recoding and logic errors, and increase the number of changes that the programer must analyze for each error occurrence.

Catalog COBOL Test Procedures

Use of catalog COBOL test procedures, similar to those specified in "IBM System 360 Operating System COBOL (E) Programer's Guide," will contribute to the standardization of the installation use of the operating system, and will enable the COBOL programers to concentrate on the understanding of a limited number of control card options.

Most attempts to learn quickly all areas of the operating system usually result in confusion and failure, and responsibility for complete understanding of the operating system should be delegated to the software-utility programers. Thus, the COBOL programer needs only to call the test procedures and to describe the data used by the program through use of the Job Card, Job Step Card and the LABEL, VOLUME, UNIT, DS NAME, DISK POSITION, SPACE, DCB and * option of the Data Definition Card.

Conversion of Master File Data

Although conversion of master file data requires consideration, the various mediums of data storage, various formats of both converting and converted files, and the numerous possible coding structures that the files can be represented in, make a generalized solution to the master file conversion problem impossible.

Line Printer Capabilities

The installation of an on-line printer on the main processor will be of considerable benefit during the conversion period. The use of a printer to record the system and compiler messages, rather than a system output tape, will enable the computer operators to immediately ascertain if the job is running correctly.

The operator will initially be unfamiliar with the 360 output formats, and the printing of the output during the job run will help the operators to separate correctly the output listings. Also, having the output printed directly will prevent the loss of prior job run outputs due to system abnormal end occurrences. Thus, ON-LINE printer during the program conversion period will decrease the number of operator errors, program turn-around time, and lost program run data.

COBOL Language Conversion Problems

Due to its uniqueness of hardware and software design, the 360 has certain characteristics that differ from those of other computers and correspondingly from other COBOL languages. Many COBOL programers, either to take advantage of the unique computer and language characteristics of other machines or in error have programed certain COBOL functions in a manner which is not acceptable to the 360.

Clearing of Memory Core

One of the major differences between the 360 and other computers is the clearing or blanking of core memory prior to initial program loading. The 360, because of its program relocation capabilities, does not automatically clear core. Thus, when a 360 COBOL program is loaded, those data areas that were not given initial values, by the VALUE clause or by the initial COBOL procedure statements, contain the values remaining from some prior program.

Thus, the programer must carefully verify that all alpha areas are initialized or cleared to blanks, and that all computational areas are cleared to zero or initialized with a correctly formatted numeric value.

A large percentage of the errors that occur during production runs are due to areas not initialized which, by chance, contained satisfactory values during the program test phase.

Problems of Computational-3 Format

Another area of frequent error occurrence is that of representing and processing numerical data in Com-

ABOUT THE AUTHOR

Norman H. Leven is an Analyst with Computer Usage Development Corporation's Los Angeles office. Prior to joining CUC, he worked as a Management Consultant with Norris & Gottfried, Inc., and as a Systems Engineer for IBM. He was involved in some of the first installations of 360's in the Los Angeles area. Mr. Leven holds a B.S. in Engineering from the University of Illinois.

putational-3 format (packed format). Numerical data is represented in this format because of the space and processing efficiencies it provides. The packed format consists of a valid sign in the right-most four bits of the low order byte, and numeric values (0-9) in all other positions.

The packing of a blank character (hexidecimal-40) provides an invalid sign representation since 4 becomes the sign character. The movement of a numeric zero to group level items that contain one or more computational-3 items can cause an invalid zero bit configuration to appear in the sign position of the computational-3 fields. The moving of a field defined as X format to a computational-3 field *does not* cause the data to be packed.

The following COBOL programing procedures should be implemented when using computational-3 format numeric fields.

1. Add numeric value fields only to computational-3 fields, not blanks or alpha format fields.
2. Examine all input numeric fields that are to be used in computations for blanks. If the field is defined as 9's format, redefine the field as X format and then examine the field, replacing or transforming all blanks to 0's.
3. Do not move an X-formatted field to a computational-3 field.
4. Do not group-level move an item containing computational-3 fields to another group-level area unless it has the same format.
5. Do not move zero to a group-level or *occurs* area containing computational-3 fields.
6. Do not compare any unpacked 9's fields possibly containing blanks (such as account number) with a computational-3 field.
7. Do not use figurative constants for computations.

Other COBOL Language Problems

The 360 COBOL processor is very stringent in its programing rules. The following is a list of major liberties that are not allowable on the 360.

1. No exit from the middle of a PERFORM LOOP.
2. No division by zero.
3. No initial subscript value to zero.
4. No special names for switch settings (the 360 does not contain switches, and leading cards should be used for program condition settings).
5. No implied subjects or implied objects. (If A = B or C is incorrect, if A =B or A = C is correct.)
6. No implied at End-Next-Sentence condition; the statement must be programed.
7. Different characters for printer spacing.
8. The WRITE AFTER option must be used for every printer line written.
9. The words AND or OR are not interpreted as meaningless English language connectors, but are used as logical operations.

Quirks of "DeBug" Trace Package

The following are some peculiarities of the debugging package.

1. The system output file must be opened before the TRACE option can be specified.
2. Because the ABEND program takes priority over all other queued I/O tasks, the last few COBOL paragraph names that were executed by the TRACE option prior to error may not be printed.
3. The TRACE and Exhibit option can consume large amounts of computer run time and should be used sparingly.
4. A count of each master tape record will prove helpful in later locating the erroneous record.