AD-A035 212 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
JAN 77
UNCLASSIFIED CCVS68-VSR165 NL

1 OF 1
ADA035212

END

DATE
FILMED
3 - 77

FEDERAL COBOL COMPILER TESTING SERVICE

VALIDATION SUMMARY REPORT

Department of the Navy
(ADPESO)

Washington, D.C.
20376

(10)

(6) COBOL COMPILER
VALIDATION SUMMARY REPORT

(14)

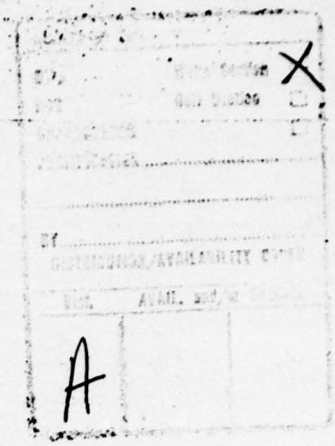
VALIDATION NUMBER CCVS68-VSR165

(11) 28 Jan 77 (12) 42p.

Prepared By:

FEDERAL COBOL COMPILER TESTING SERVICE
DEPARTMENT OF THE NAVY (ADPESO)
WASHINGTON, D.C. 20376

408 438



COBOL COMPILER VALIDATION

1.	Validation Number	CCVS68-VSR165
2.	Vendor	BURROUGHS
3.	Mainframe	B6700/B7700
4.	Compiler Identification	II.8
5.	Operating System Identification	MCP II.8
6.	CCVS Version	6.2
7.	Federal Information Processing Standard Publication	21

*Information regarding this compiler can be obtained from:

Mr. Jay Wolf
Burroughs Corporation
Federal and Special Systems Group
P.O. Box 517
Paoli, Pennsylvania 19301

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1. Purpose of the Validation Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose usage does not conform to Federal Standard COBOL as adopted from American National Standard COBOL (X3.23-1968) by Federal Information Processing Standards Publication 21 (FIPS PUB 21).

1.2. Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System (CCVS). The COBOL Compiler Validation System consists of audit routines containing features of Federal Standard COBOL, their related data, and an executive routine (VP-routine) which prepares the audit routines for compilation. Each audit routine is a COBOL program which includes tests and supporting procedures used to indicate the results of the tests.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The report produced by each routine indicates whether the compiler passed or failed the tests contained in the routine.

If the compiler rejects some language elements by terminating compilation, giving fatal diagnostic messages, or terminating execution abnormally, then the test containing the code the compiler was unable to process is deleted and the audit routine compilation and execution repeated.

The compilation listings and the output reports of the audit routines constitute the raw data from which the members of the Federal COBOL Compiler Testing Service produce a Validation Summary Report.

1.3. Organization of the VSR

The Validation Summary Report is made up of several sections the contents of which are described below:

a. Section 2 summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System.

Section 2 is subdivided into Section 2.1, syntax not accepted by the compiler, and Section 2.2, semantic differences between the compiler implementation and the language specification. All errors are grouped by processing module.

b. FIPS PUB 21 defines four levels of Federal Standard COBOL. Section 3 of the VSR lists the discrepancies described in Section 2 by the level in which the problem occurs.

c. Section 4 contains information obtained during the validation process

which, while not impacting the status of the compiler with respect to the defined Federal COBOL levels, may be of interest to a user of the compiler.

d. Section 5 contains information which describes the software environment in which the compiler was tested. This includes the name and version of the compiler; the name and version of the operating system; the implementor-names which were used in the Environment Division of the programs comprising the CCVS; the options used with the compiler; and if applicable, information regarding the use of compiler optimization features.

e. Appendix A is the Validation Summary Working Document, a working paper resulting from the compilation and execution of the CCVS, and from which the VSR is derived.

1.4. Compliance to FIPS PUB 21

Definition of an Implementation of USA Standard COBOL

(excerpts from American National Standard COBOL X3.23-1968, Chapter 1)

Throughout the USA Standard COBOL specifications, there are certain language elements that depend, for their implementation, on particular types of hardware components. The implementor specifies the minimum hardware configuration required for a given implementation of USA Standard COBOL and the hardware components that it supports. Any language elements that depend on hardware components for which support is claimed must be implemented. Language elements that are not implemented because of their dependence on hardware components whose support is not claimed for an implementation must be so specified and their absence will not render the implementation nonstandard.

When a facility is provided that accomplishes the function specified by any particular COBOL element, it may be unnecessary to include that particular element within the COBOL source program. If such an unnecessary element does appear in the source program, it must be accepted by the compiler. However, if the element does not lead to the production of object code, no substitute statements shall be required within the COBOL source program to accomplish this function.

By the same token, the inclusion of language elements or of functions that are not a part of the USA Standard COBOL specifications will not render an implementation of USA Standard COBOL nonstandard. This is true even though it may imply the extension of the list of reserved words by the implementor, and prevent proper compilation of some programs which conform to the Standard.

The Federal COBOL Standard

The Federal COBOL Standard is the same as American National Standard COBOL (X3.23-1968) with two exceptions:

> The Federal Standard defines 4 levels and the ANSI Standard defines only the minimum COBOL implementation and the full standard. Low and High levels of the Federal COBOL Standard (see 1.5) correspond to the above two ANSI levels (minus the Report Writer module). Two additional levels, low-intermediate and high-intermediate have been included in the Federal Standard between the highest and lowest subsets. These additional levels accommodate hardware which cannot support the full standard, but which is capable of implementing more than the minimum standard.
>
> The Report Writer Module has been dropped from the Federal Standard and is not included in the validation process.

The Federal Standard requires that a compiler contain as a *minimum* the elements specified in at least one of the Federal levels. No restrictions are imposed on the inclusion of selected features from higher levels or even unique vendor extensions. Compatibility among various implementations of a given level containing additional features must be controlled by management imposed standards and restrictions.

1.5. Federal Standard COBOL Levels

The Federal Government has defined four nested levels of Standard COBOL X3.23-1968. All compilers acquired by a Federal agency must contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs.

	Low Level	Low-Intermediate Level	High-Intermediate Level	High Level
Nucleus.....	1	2	2	2
FPM				
Table Handling.....	1	2	2	3
Sequential Access....	1	2	2	2
Random Access.....	-	2	2	2
Sort.....	-	-	1	2
Segmentation.....	-	1	1	2
Library.....	-	1	1	2

1.6. Use of the VSR

The Federal COBOL Compiler Testing Service may make full and free public disclosure of the Validation Summary Report (VSR) in accordance with the "Freedom of Information Act" (5 U.S.C. #552). The results of the validation are only for the purpose of satisfying United States Government requirements, and apply only to the computer system, operating system release, and compiler version identified in the VSR.

The COBOL Compiler Validation System is used to determine, insofar as is practical, the degree to which the subject compiler conforms to the COBOL Standard. Thus, the VSR is necessarily discretionary and judgmental. The United States Government does not represent or warrant that the statements, or any one of them, set forth in the VSR are accurate or complete. The VSR is not meant to be used for the purpose of publicizing the findings summarized therein.

1.7. Sources of Additional Information

FIPS PUB 21 defines the Federal COBOL Language Standard. This publication is available from the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

The detailed COBOL language specifications are given in the publication "USA Standard COBOL, X3.23-1968", available from American National Standards

Institute, 1430 Broadway, New York, New York 10018.

An explanation of the COBOL Compiler Validation System is contained in the CCVS User's Guide. This document explains how to run the compiler validation system. The User's Guide and a magnetic tape source image copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. Specify AD772600 for the Users Guide and AD772601 for the CCVS tape.

1.8. Requests for Interpretation

Questions regarding this VSR or the CCVS should be forwarded to the FCCTS. If any problem cannot be adequately resolved through the FCCTS, the request for interpretation will be forwarded to the Federal COBOL Interpretation Committee for final resolution.

A brochure describing the Validation process including the procedures for requesting a validation and resolution of questions involving interpretation of the current Federal Standard is available from the Department of the Navy, Federal COBOL Compiler Testing Service, Washington, D.C. 20376.

1.9. Federal Standard COBOL Approved Intrepretation

The National Bureau of Standards published in the Federal Register Vol. 41 No. 179, September 14, 1976, an approved interpretation of Federal Standard COBOL as pertains to the evaluation of arithmetic expressions in the COMPUTE statements. This interpretation states that "size of the intermediate result field is implementor-defined."

Since the results of evaluating arithmetic expressions are not predictable, all COMPUTE statements and IF statements containing arithmetic expressions have been removed from the COBOL Compiler Validation System.

SECTION 2. DETAILED EVALUATION OF ERRORS

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System. The section is subdivided into two sections: Section 2.1. lists the language syntax not accepted by the compiler; Section 2.2. explains the semantic differences between the compiler implementation and the language specification. All errors within each section are grouped by processing module.

Each program in the COBOL Compiler Validation System is identified by a 5-character program name. The name associates the routine with the functional processing module and level of ANS COBOL tested within the program.

The five character name has the general format XXNMM. The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

LB - Library
NC - Nucleus
RC - Random Access
SG - Segmentation
SQ - Sequential
ST - Sort
TH - Table Handling

The third character of the audit routine name is either a 1, 2, or 3, and identifies the level of the functional module being tested. Each module and level is represented by several programs. The fourth and fifth characters of the program name are sequence numbers for programs which test features in the same level of the same functional processing module.

As an example, the program name NC21D is the tenth program in the series of routines which test the second level of the Nucleus module.

Each error noted in this section makes reference to a program or functional COBOL module in the Validation Summary Working Document (APPENDIX A). This reference provides the documented results of an occurrence of errors detected during the running of the COBOL Compiler Validation System using the compiler being validated. The Validation Summary Working Document is presented in sequence by functional module, functional module level and program number as defined above.

2.1 Syntactical Errors.

No errors.

2.2 Semantic Errors.

2.2.1 Nucleus Module.

2.2.1.1 Leading SPACE(s) are truncated when ACCEPTing data from the SPO (standard console ACCEPT and DISPLAY device).

See NC109.
See NC204.

2.2.1.2 The compiler does not replace the entire comment entry of the DATE-COMPILED paragraph with the current date. It only replaces the first sentence in the paragraph.

See NC203.

2.2.2 Random Access Module.

2.2.2.1 Label DECLARATIVE procedures for random access mass storage devices do not appear to be supported.

See RC201.
See RC202.
See RC203.

2.2.2.2 A file can be opened and read after a CLOSE WITH LOCK statement has been executed for the file.

See RC201.

2.2.2.3 The statement VALUE OF data-name IS has not been implemented.

See RC203.

2.2.3 Segmentation Module.

2.2.3.1 Independent segments are not provided in their initial state each time these segments are made available to the program.

See SG102.
See SG103.
See SG201.
See SG203.

2.2.4 Sequential Access Module.

2.2.4.1 For tape files, the ENDING LABEL procedures execute following a CLOSE even though the file is not at an end-of-file condition.

See SQ203.

See SQ217.

2.2.4.2 Label DECLARATIVE procedures do not appear to be supported on sequential access mass storage devices.

See SQ205.
See SQ208.
See SQ209.
See SQ213.
See SQ216.

SECTION 3. COMPILER STATUS

Section 1.5 explains the four levels of Federal Standard COBOL. This section lists the discrepancies described in Section 2 by the Federal level in which the problem occurs. All errors listed for a lower level are also errors in any higher level, even though they are listed only in the lower level. The paragraph number from Section 2 is used to reference the errors in each Federal level.

3.1. Low Level

2.2.1.1 ACCEPT - leading blanks truncated when accepting from the console.

3.2. Low-Intermediate Level

2.2.1.2 DATE-COMPILED - date does not replace the entire paragraph.
2.2.2.1 USE procedures - Random Access.
2.2.2.2 CLOSE WITH LOCK - Random Access.
2.2.2.3 VALUE OF data-name - Random Access.
2.2.3.1 Segmentation - independent segments not in initial state.
2.2.4.1 Ending label procedures executed prior to end-of-file.
2.2.4.2 USE procedures - Sequential Access mass storage.

3.3 High-Intermediate Level

None

3.4 High Level

None

SECTION 4. INFORMATION ITEMS

Section 4.1 covers the cases where the results of a given statement are not defined in the language specifications.

Section 4.2. gives other information about the compiler which was discovered during validation. Included in this section are items which are hardware dependent, and situations where implementor defined variations are permitted.

Section 4.3. gives information on hardware dependent features that are not supported by the subject COBOL Compiler.

Section 4.4 gives information concerning the use of facilities outside the COBOL program that accomplish functions defined in COBOL (see 1.4 of this VSR).

The COBOL language specification "American National Standard COBOL X3.23-1968" as defined by FIPS Pub 21 is used as the reference document.

4.1. Information Tests

4.1.1. Nucleus Module

4.1.1.1. CLOSE followed by an OPEN OUTPUT of a printer destined file (LB103):

COPY-TEST-2 of LB103 closed and then reopened the printer file. The result of closing and then reopening a unit record output file is not specifically addressed in the language specifications. The object of this test is to determine whether the file is repositioned to its beginning and all previous results overwritten, or whether all current output is preserved with additional output concatenated to it.

The results for this compiler:

All printed output was provided.

4.1.1.2. IF...Signed Numeric item equals Alphanumeric item (NC103):

IF-TEST-65 compares two elementary items with the first item having PIC S9(18) VALUE 111111111111111111 and the second item having a PIC X(18) VALUE "111111111111111111". The items are compared by the form:

IF identifer-1 EQUAL TO identifier-2...

The results for this compiler:

The two elementary items did not compare equal.

4.1.1.3. IF...NUMERIC Condition tests (NC103):

CLASS-TEST-17 and CLASS-TEST-22 of NC103 are information tests where data items with PICTURE S9 contain -1 and +1 respectively. Each data item is redefined as an alphanumeric data item PICTURE X. The redefined data item is then

referenced in a NUMERIC test to determine whether the signed data item contains an indicator to represent the sign, i.e. an overpunch. If both data items are determined to be numeric, then the system does not use an indicator technique for the sign. If both data items are determined to be not numeric then the system uses an indicator method for both positive and negative numbers. If one of the data items is numeric and the other is not, then the system uses the indicator method for one case but not the other. The sign is not addressed by the COBOL language specifications and consequently is left to the discretion of the implementors of COBOL compilers as to its representation.

The results for this compiler:

Negative data redefined as unsigned data:

The data item tested NOT NUMERIC.

Positive data redefined as unsigned data:

The data item tested NOT NUMERIC.

4.1.1.4 Move signed numeric field to alphanumeric field (NC105):

MOVE-TEST-148 and MOVE-TEST-149 in NC105 move signed numeric fields with PICTURE S9(5) to fields with PICTURE X(5). The source fields had contents of +60666 and -70717 for MOVE-TESTS-148 and 149 respectively. The language specification is not definitive as to whether the absolute value is to be moved (i.e. the sign is stripped) or whether the contents are moved without regard to the sign. The results of these tests should be considered only in light of the results of the information test in 4.1.1.2.

The results for this compiler:

MOVE +60666 TO X(5)

The absolute value was moved; the sign was stripped.

MOVE -70717 TO X(5)

The absolute value was moved; the sign was stripped.

4.1.1.5. ADD without ON SIZE ERROR (NC106):

ADD-TEST-17 in NC106 checks the effect of a size error on the result of an ADD statement which does not have the ON SIZE ERROR phrase. The statement used was adding SV9(5), S9(6)V9(6), SV9(5) GIVING S9(5) ROUNDED. Identifier S9(6)V9(6) contained the value 333333.333333. The SV9(5) descriptions were the same identifier and contained the value .11111. The language specifications do not define the results of an arithmetic operation without the SIZE ERROR phrase, when the result cannot be contained in the resultant-identifier which causes truncation of significant digits. The results of this test require 6 digits and the resultant-identifier contains only 5. For a further discussion, see X3.23-1968 American National Standard COBOL page 2-28, Section

6.2.2(1), The SIZE ERROR clause.

The results for this compiler:

The result was stored in the resultant-identifier with the high order digit truncated.

4.1.1.6. Abbreviated Combined Relation Conditions (NC201):

IF--TEST-110 in NC201 utilizes an abbreviated combined relation condition to determine the direction the compiler will take in a case where the use of the word NOT is indeed confusing. The language specification states on page 2-73, section 5.2.1.1 that when the construct AND NOT LESS THAN is used in a source program, the word NOT is to be treated as a logical operator. This becomes relevant when taken in the context of the later abbreviation of the relational operator in the statement. For example, A = B AND NOT LESS THAN C OR D. would expand to the following: A = B AND NOT A LESS THAN C OR A LESS THAN D. The test in question contains the following: A GREATER THAN B AND IS NOT LESS THAN C OR D which strongly suggests that the word NOT in this case is relational in nature and not logical. The concept of an optional word (IS) changing the meaning of a statement is contrary to the philosophy of COBOL and as a result this test is presented as information only.

The values for the variables in the condition are A=5, B=7, C=1 and D=6. Using these values in place of the variables, the condition expands as the following if NOT is treated as relational:

((5 GREATER THAN 7) AND (5 NOT LESS THAN 1)) OR (5 NOT LESS THAN 6)

In this case the condition is false.

If the NOT is logical, the condition expands as the following:

((5 GREATER THAN 7 AND NOT (5 LESS THAN 1)) OR (5 LESS THAN 6)

In this case the condition is true.

The results for this compiler:

The word NOT was treated as a relational operator.

4.1.1.7 The Synchronized Clause (NC108)

The language specification does not specify the results of the use of the synchronized clause without the specification of either LEFT or RIGHT. In the event that such results are provided during the validation process, they are included here as information.

The results for this compiler:

The compiler did not indicate how the synchronization would be done.

4.1.1.8 The ACCEPT Statement (NC109):

The language specification does not specify the results of ACCEPT when the hardware is capable of transferring the amount of data as specified in the receiving field and more characters are entered for transfer on the device than were expected by the receiving field.

The results of this compiler:

The leftmost character(s) are truncated when the number of characters ACCEPTed from the SPO exceeds the size of the receiving field.

4.1.2. Random Access Module

4.1.2.1 Multiple length records (RC102):

Program RC102 produces a file containing records of multiple sizes. The purpose of this test is to determine whether multiple size records are written or whether the maximum size record is always written. As the file is created, records of both length are written; and as the smaller records are built, information uniquely identifying each record is placed in the portion of the larger record that would not logically be written to the external media. When the file is later retrieved the record area is examined to determine whether the extra portion beyond the end of the smaller record is available when each of the smaller records is read. The language specificaion does not readily suggest that the external media will be impacted by the size of the logical record as defined in the COBOL program.

The results for this compiler are:

The compiler supports fixed length records.

4.1.3. Sequential Access Module

4.1.3.1. CLOSE REEL for Output Files (SQ101):

The language specifications are not clear as to the first record written to a new reel after a CLOSE REEL statement has been executed. In the program SQ101, a multi-reel file is created using the CLOSE REEL statement. INFO-ON-TEST-5 reads the first record of the second reel.

The results for this compiler:

The first record read on the second reel was the first record written following the CLOSE REEL.

4.1.3.2. Multiple Length Records for Tape Files (SQ102):

The audit routine SQ102 creates a tape file with multiple length records. The file is then read and the record area examined to determine whether fixed length

records were written. For a discussion of this operation, refer to 4.1.2.1. under the discussion of Random Access.

The results for this compiler are:

The compiler appears to support multiple length records.

4.1.3.3. Multiple length records for sequential mass storage files (SQ104):

The audit routine SQ104 creates a sequential mass storage file with multiple length records. The file is then read and the records are examined to determine whether fixed length records are written. Refer to 4.1.2.1 under the discussion of multiple records for Random Access Files for a description of a similar test.

The results for this compiler are:

The compiler appears to support fixed length records.

4.1.3.4. FILE-LIMITS Clause

The FILE-LIMITS clause need not be used by the compiler and/or operating system to allocate file space. The language specification allows an external source or function to accomplish some functions described in the COBOL language, in particular, file facilities management functions.

The results for this compiler are:

The FILE-LIMITS clause is used by this compiler in defining the relative limits of the file.

4.1.3.5 RERUN Clause Option

The language specification requires that the implementor must provide at least one of the specified forms of the RERUN clause.

The RERUN clause used for this compiler was:

    RERUN ON DISK EVERY 5 RECORDS OF file-name-1.

4.2. Other Information

4.2.1. Library Source Text

The language specification is somewhat vague as to the content of the library text prior to being copied. As a result there are a multitude of techniques that have been noted for establishing the source text for subsequent use by a COPY statement. This information is provided in order to understand the technique used by this system.

The results for this compiler are:

There were no modifications required for the library text.

4.2.2. Normal Print Positioning (SQ218):

The language specification does not provide the default specifications for the use of the WRITE statement in relation to files destined for a printer when the ADVANCING phrase is not specified by the user. The purpose of this test is to observe the default taken by the compiler.

The results for this compiler are:

This compiler assumes BEFORE ADVANCING 1 when no advancing phrase is used.

4.3 Hardware dependent features.

- o Hardware Switches

This system does not support hardware switches and as a result, all tests related to the testing of switches were deleted.

- o UNIT Option

The logical concept of UNIT is not defined by this system for mass storage devices. As a result all tests related to testing of the UNIT option were deleted.

4.4 Transparent Implementation

User labels. The syntax of the statement VALUE OF data-name IS produces a warning message but not a fatal compile time error. The system has the SYSTEM/RTABLGEN function to accomplish the semantics for this COBOL language statement external to the COBOL program.

SECTION 5 SOFTWARE ENVIRONMENT.

The compiler referenced in this document was validated using the software environment described in this section. When using a modification of the described environment, the compiler may or may not continue to conform to the standard. It should be noted that during the validation process, an attempt is made to validate as many different options as possible.

The use of compiler options, implementor-names in the Environment Division and any form of optimization which is not described in this report could cause the compiler to produce a program that does not perform according to the specifications of Standard COBOL. Only the environment described in this document has been used with this compiler to satisfy the requirements of FIPS PUB 21 and FPMR 101-32.1305.1a. (Any deviations which must be corrected as per the referenced FPMR are described in Sections 2 and 3 of this report.)

1. Options or parameters used on the processor call statement for the compiler: The following options/parameters were used during the validation.

Options specified:

```
$ SET OPTIMIZE (also ran without OPTIMIZE)
      STACK
      LISTP
      LISTDELETED
      USASI
      RESET B6700
```

Options defaulted:

See Burroughs COBOL Reference Manual No. 5000656 Chapter 13.

2. Environment Division implementor-names.

Printer destined files

PRINTER

Tape files

TAPE

Sequential mass storage files

DISK

Random Access files

DISK

Sort files (SD)

SORT DISK

Switch names

Not supported by the compiler.

Source Computer names

B6700-B7700

Object Computer names

B6700-B7700

3. Optimization. The compiler may or may not have optimization features. If there was an optimization feature available, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

The optimization feature for this compiler is invoked through the compiler statement ($SET). See 1. above. There was no differnce in the execution of the programs when the optimization feature was invoked.

4. Compiler.

Burroughs COBOL Version II.8

5. Operating system.

Burroughs MCP II.8

APPENDIX A

VALIDATION SUMMARY WORKING DOCUMENT

A-1 This appendix is a working paper produced during the validation and documents the results of the compilation and execution of each of the programs comprising the CCVS. The results contained herein are based on the use of the compiler within the Validation Environment identified in this appendix. This appendix (*Validation Summary Working Document*) is not part of the official Validation Summary Report (VSR) and is not intended to reflect in any way the compiler's usefulness or degree of conformance to the language specifications.

The reader of this appendix should keep in mind that the same problem area may appear in more than one program, but is considered only as one single discrepancy and as such is reflected only once in the body of the VSR. (The VSR will in turn only reference the first occurrence of the problem in the appendix.)

This appendix is divided into two parts. The first part describes the Validation Environment. The second part of the document is divided into categories of information: compilation and execution results.

The reference document for COBOL is FIPS PUB 21 (X3.23-1968).

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	B6700/B7700 Version II.8
COMPUTER SYSTEM:	Burroughs B6700/B7700
OPERATING SYSTEM:	Burroughs MCP II.8

LIBRARY MODULE LEVEL 1

L3101 through L3107

A. Compilation:
No errors.

B. Execution:
No errors.

LIBRARY MODULE LEVEL 2

L3201 through L3205

A. Compilation:
No errors.

B. Execution:
No errors.

NUCLEUS MODULE LEVEL 1

NC101 through NC108

A. Compilation:
No errors.

B. Execution:
No errors.

NC109

A. Compilation:
No errors.

B. Execution:

ACC-TEST-6 accepts 20 characters from the SPO (standard console ACCEPT and DISPLAY device) and compares the characters it has ACCEPTed into an elementary item PIC A(20) with another elementary item PIC A(20). The fields did not compare equal because the leading space was truncated from the characters transmitted from the SPO.

Computed result: ABC XYZ .
Expected result: ABC XYZ .

ACC-TEST-7 accepts 9 characters from the SPO into an elementary numeric item PIC 9(9) and then compares this item with another elementary numeric item PIC 9(9) VALUE 012345678. The fields did not compare equal. It is important to understand that a total of ten (10) characters were transmitted from the SPO. When the transferred data exceeds the size of the receiving item, only the leftmost characters of the data are stored in the receiving data item.

See Page 2-29 6.3.3 (4) b. The ACCEPT Statement.

NC110 through NC113

A. COMPILATION:
No errors.

B. Execution:
No errors.

NUCLEUS MODULE LEVEL 2

NC201 through NC202

A. Compilation:
No errors.

B. Execution:
No errors.

NC203

A. Compilation:

The DATE-COMPILED paragraph in this program tests: (a) the acceptance of a comment entry within the DATE-COMPILED paragraph and (b) replacement of the date compiled paragraph during compilation with a paragraph of the form:

DATE-COMPILED. Current-date.

This compiler inserted the current-date into the paragraph rather than replace the comment entry.

See Page 2-26, Paragraph 2.4.3(1) The DATE-COMPILED PARAGRAPH.

B. Execution:
No errors.

NC204

A. Compilation:
No errors.

B. Execution:

ACC-TEST-5 accepts 20 characters from the SPO and compares the characters it has ACCEPTed into an elementary item PIC A(20) with another elementary item PIC A(20). The fields did not compare equal because the leading SPACE was truncated from the characters transmitted from the SPO.

Computed result: ABC XYZ .
Expected result: ABC XYZ .

NC205 through NC212

A. Compilation
No errors.

B. Execution
No errors.

RANDOM ACCESS MODULE LEVEL 1

RC101 through RC105

A. Compilation:
No errors.

B. Execution:
No errors.

RANDOM ACCESS MODULE LEVEL 2

RC201

A. Compilation:

The following DECLARATIVE procedures produced the compile time message:

WARNING : ROUTINE NOT APPLICABLE TO ANY FILE***

USE AFTER BEGINNING FILE LABEL PROCEDURE ON OUTPUT
USE BEFORE STANDARD BEGINNING FILE LABEL PROCEDURE ON I-O
USE BEFORE ENDING FILE LABEL PROCEDURE ON OUTPUT
USE AFTER ENDING FILE LABEL PROCEDURE ON I-O
USE AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON INPUT
USE BEFORE STANDARD ENDING FILE LABEL PROCEDURE ON INPUT

B. Execution:

CLOSE TEST-01 and CLOSE TEST-02 fail because DECLARATIVE procedure was not properly executed "AFTER BEGINNING FILE LABEL PROCEDURE ON OUTPUT".

CLOSE TEST-09 and CLOSE TEST-10 fail because DECLARATIVE procedure was not properly executed "BEFORE ENDING FILE LABEL PROCEDURE OUTPUT".

OPEN INPUT TEST-11 fails because DECLARATIVE procedure was not properly executed "AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON INPUT".

OPEN I-O TEST-11 fails because DECLARATIVE procedure was not properly executed "BEFORE STANDARD BEGINNING FILE LABEL PROCEDURE ON I-O".

CLOSE TEST-25 fails because DECLARATIVE procedure

was not properly executed "BEFORE STANDARD ENDING FILE LABEL PROCEDURE ON INPUT".

CLOSE TEST-25 fails because DECLARATIVE procedure was not properly executed "AFTER ENDING FILE LABEL PROCEDURE ON I-O".

OPEN AFTER LOCK TEST-27 fails because a file is made available after it was CLOSED WITH LOCK.

See Pages 2-155 through 2-182 Random Access Level 2.

RC202

A. Compilation:

The following DECLARATIVE procedures produced the compile time messages:

WARNING : FILE NOT ASSIGNED TO TAPE***
WARNING : ROUTINE NOT APPLICABLE TO ANY FILE ***

USE BEFORE BEGINNING FILE LABEL PROCEDURE ON file-name
USE AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON file-name
USE AFTER ENDING FILE LABEL PROCEDURE file-name
USE BEFORE ENDING FILE LABEL PROCEDURE ON file-name

B. Execution:

OPEN OUTPUT TEST-01 fails because the DECLARATIVE procedure was not properly executed "BEFORE BEGINNING FILE LABEL PROCEDURE ON file-name".

OPEN OUTPUT TEST-02 fails because the DECLARATIVE procedure was not properly executed "AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON file-name".

CLOSE TEST-09 fails because the DECLARATIVE procedure was not properly executed "AFTER ENDING FILE LABEL PROCEDURE file-name".

CLOSE TEST-10 fails because the DECLARATIVE procedure was not properly executed "BEFORE ENDING FILE LABEL PROCEDURE ON file-name".

OPEN INPUT TEST-11 fails because the DECLARATIVE procedure was not properly executed "AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON file-name".

OPEN I-O TEST-11 fails because the DECLARATIVE procedure was not properly executed "BEFORE BEGINNING FILE LABEL PROCEDURE

ON file-name".

CLOSE TEST-27 fails because the DECLARATIVE procedure was not properly executed "BEFORE ENDING FILE LABEL PROCEDURE ON file-name".

CLOSE TEST-28 fails because the DECLARATIVE procedure was not properly executed "AFTER ENDING FILE LABEL PROCEDURE file-name".

See Page 2-175 4.1.4 (2) A. Standard Close File.
See Page 2-177 4.2.4 (4) and 4.2.4 (7) The OPEN Statement.

RC203

A. Compilation:

In the FD section the following construct produced the compile time message WARNING : CONSTRUCT NOT IMPLEMENTED ***

VALUE OF USER-ID-C IS "RANDOM-FILE-3".

The following DECLARATIVE procedure produced the compile time message WARNING : FILE NOT ASSIGNED TO TAPE ***
WARNING : ROUTINE NOT APPLICABLE TO ANY FILE ***

USE BEFORE STANDARD BEGINNING FILE LABEL PROCEDURE ON file-name
USE AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON file-name
USE BEFORE STANDARD ENDING FILE LABEL PROCEDURE ON file-name
USE AFTER STANDARD ENDING FILE LABL PROCEDURE ON file-name

B. Execution:

All tests in RC203 failed to execute properly because of the failure to execute the DECLARATIVE procedures associated with the files.

See Pages 2-155 through 2-182 Random Access Level 2.

SEGMENTAION MODULE LEVEL 1

SS101

A. Compilation:
No errors.

B. Execution:
No errors.

SS102

A. Compilation:
No errors.

B. Execution:

SEG-TEST-5 fails because independent segments are not in their initial state each time these segments are made available to the program.

See Page 2-241 2.2.3 Independent Segments.

SS103

A. Compilation:
No errors.

B. Execution:

INITIAL-STATE-TEST-1, FALL-THRU-TEST-6, and GO-TO-ALTER-IND-TEST-7 all fail because independent segments are not provided in their initial state each time they are made available to the program.

See Page 2-241 2.2.3 Independent Segments.

SEGMENTAION MODULE LEVEL 2

S3201

A. Compilation:
No errors.

B. Execution:

The following tests fail because independent segments are not provided in their initial state each time they are made available to the program:

SEG-TEST-22
SEG-TEST-23
SEG-TEST-24
SEG-TEST-25
SEG-TEST-26
SEG-TEST-27
SEG-TEST-28
SEG-TEST-29
SEG-TEST-30
SEG-TEST-31
SEG-TEST-32
SEG-TEST-33
SEG-TEST-34
SEG-TEST-35
SEG-TEST-36
SEG-TEST-37
SEG-TEST-38
SEG-TEST-39
SEG-TEST-40
SEG-TEST-41
SEG-TEST-42
SEG-TEST-43
SEG-TEST-64
SEG-TEST-65

See Page 2-246 2.2.3 Independent Segments.

S3202

A. Compilation:
No errors.

B. Execution:
No errors.

SS203

A. Compilation:
No errors.

B. Execution:

The following tests fail because the independent segments are not provided in their initial state each time they are made available to the program:

INITIAL STATE PARA-43B (TEST 14)
INITIAL STATE PARA-68C (TEST 15).

See Page 2-245 2.2.3 Independent Segments.

SEQUENTIAL ACCESS MODULE LEVEL 1

SQ101 through SQ108

A. Compilation:
No errors.

B. Execution:
No errors.

SEQUENTIAL ACCESS MODULE LEVEL 2

SQ201 through SQ202

A. Compilation:
No errors.

B. Execution:
No errors.

SQ203

A. Compilation:
No errors.

B. Execution:

USE-TEST-9 fails to execute properly because the ENDING LABEL procedure executed on a file that had been OPENed for INPUT and this file was CLOSEd before the file reached the end-of-file condition.

See Page 2-150 4.4 The USE Statement.
See Page 2-143 4.1 The CLOSE Statement.

SQ204

A. Compilation:
No errors.

B. Execution:
No errors.

SQ205

A. Compilation:

The following statements produced the compile time message WARNING : CONSTRUCT NOT IMPLEMENTED ***

VALUE OF USER-ID-A IS "SEQUENTIAL-FILE-1"
VALUE OF USER-ID-C IS "SEQUENTIAL-FILE-3".

B. Execution:

LABEL-TEST-14, LABEL-TEST-15, LABEL-TEST-16, and LABEL-TEST-18 all fail because the VALUE OF data-name IS literal clause is not implemented.

See Page 2-150 4.4 The USE Statement.
See Page 2-142 3.7 The VALUE OF Clause.

SQ206 through SQ207

A. Compilation:
No errors.

B. Execution:
No errors.

SQ208

A. Compilation:

The following DECLARATIVES produced the compile time message:

WARNING : ROUTINE NOT APPLICABLE TO ANY FILE***

USE BEFORE BEGINNING FILE LABEL PROCEDURE ON I-O
USE AFTER BEGINNING FILE LABEL PROCEDURE ON I-O
USE BEFORE ENDING FILE LABEL PROCEDURE ON I-O
USE AFTER ENDING FILE LABEL PROCEDURE ON I-O.

B. Execution:

OPEN-I-O-TEST, LABEL PROCEDURE 07-USE-TEST, 08-USE-TEST, 09-USE-TEST, and 10-USE-TEST all failed because of improper execution of the DECLARATIVE procedures associated with the I-O files.

See Page 2-150 4.4 The USE Statement.

SQ209

A. Compilation:

The following DECLARATIVE procedures produced compile time messages:

WARNING : FILE NOT ASSIGNED TO TAPE***
WARNING : ROUTINE NOT APPLICABLE TO ANY FILE***

USE AFTER STANDARD ENDING UNIT LABEL PROCEDURE ON MASS-FILE-A
USE AFTER STANDARD BEGINNING UNIT LABEL PROCEDURE ON MASS-FILE-A
USE BEFORE STANDARD ENDING UNIT LABEL PROCEDURE ON MASS-FILE-A
USE BEFORE STANDARD ENDING UNIT LABEL PROCEDURE ON MASS-FILE-B
USE BEFORE STANDARD BEGINNING UNIT LABEL PROCEDURE ON MASS-FILE-B
USE BEFORE BEGINNING LABEL PROCEDURE ON DUMMY-I-O-FILE
USE BEFORE ENDING LABEL PROCEDURE ON DUMMY-I-O-FILE

B. Execution:

OPEN SERIES OPEN-TEST-1 fails because the associated DECLARATIVE procedures do not execute properly.

See Page 2-150 4.4 The USE Statement.

SA210

A. Compilation:

Compiler messages were produced for all DECLARATIVE procedures which referred to UNIT; however, these messages were WARNINGs and all CLOSE UNIT tests were deleted from the program so these messages were appropriate.

B. Execution:

All CLOSE UNIT tests were deleted so no errors.

SQ211 through SQ212

A. Compilation:
No errors.

B. Execution:
No errors.

SA213

A. Compilation:

The following DECLARATIVE procedures produced the compile time messages:

WARNING : FILE NOT ASSIGNED TO TAPE***
WARNING : ROUTINE NOT APPLICABLE TO ANY FILE***

```
USE AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON MASS-FILE
USE AFTER STANDARD BEGINNING FILE LABEL PROCEDURE ON DUMMY-MASS-FILE
USE AFTER STANDARD ENDING FILE LABEL PROCEDURE ON MASS-FILE
USE AFTER STANDARD ENDING FILE LABEL PROCEDURE ON DUMMY-MASS-FILE.
```

B. Execution:

D4-USE-TEST fails because the ending file label procedure is improperly executed when the file is CLOSEd; however, the end-of-file condition has not been reached.

See Page 2-144 4.1.4 (2)C Standard Close File.

D7-USE-TEST and D9-USE-TEST fail because the associated DECLARATIVE procedures are not properly executed.

See Page 2-150 4.4 The USE Statement.

SQ214

A. Compilation:
No errors.

B. Execution:
No errors.

SQ215

A. Compilation:

The following statements produced the compile time message:

```
WARNING :  CONSTRUCT NOT IMPLEMENTED***

VALUE OF ELEMENT OF LABEL-GROUP-1
         ELEMENT IN WORK-GROUP-1,
         ELEMENT IN LABEL-GROUP-2
         ELEMENT OF WORK-GROUP-2.
```

B. Execution:

The test VALUE OF ... QUAL, LABEL-TEST-1 fails because the constructs shown in the Compilation section above were not implemented as indicated by the WARNING compile time message.

See Page 2-142 3.7 The VALUE OF Clause.

SQ216

A. Compilation:

The following DECLARATIVE procedures produced the compile time messages:

WARNING : FILE NOT ASSIGNED TO TAPE***
WARNING : ROUTINE NOT APPLICABLE TO ANY FILE***

USE AFTER ENDING FILE LABEL PROCEDURE ON MASS-FILE
USE AFTER ENDING UNIT LABEL PROCEDURE ON MULTI-MASS-FILE.

B. Execution:

CLOSE WITH LOCK LOCK-TEST-3 fails because the DECLARATIVE procedures associated with the files do not execute properly.

See Page 2-144 4.1.4 (2)C Standard Close File.

SQ217

A. Compilation:
No errors.

B. Execution:

LOCK-TEST-1D fails because the ending label procedure was executed even though the end-of-file condition was not reached when the file was CLOSEd.

See Page 2-144 4.1.4 (2)C Standard Close File.

SQ218

A. Compilation:
No errors.

B. Execution:
No errors.

SORT MODULE LEVEL 1

ST101 through ST114

A. Compilation:
No errors.

B. Execution:
No errors.

SORT MODULE LEVEL 2

ST201 through ST207

A. Compilation:
No errors.

B. Execution:
No errors.

TABLE HANDLING MODULE LEVEL 1

TH101 through TH104

A. Compilation:
No errors.

B. Execution:
No errors.

TABLE HANDLING MODULE LEVEL 2

TH201 through TH211

A. Compilation:
No errors.

B. Execution:
No errors.

TABLE HANDLING MODULE LEVEL 3

TH301 through TH311

A. Compilation:
No errors.

B. Execution:
No errors.