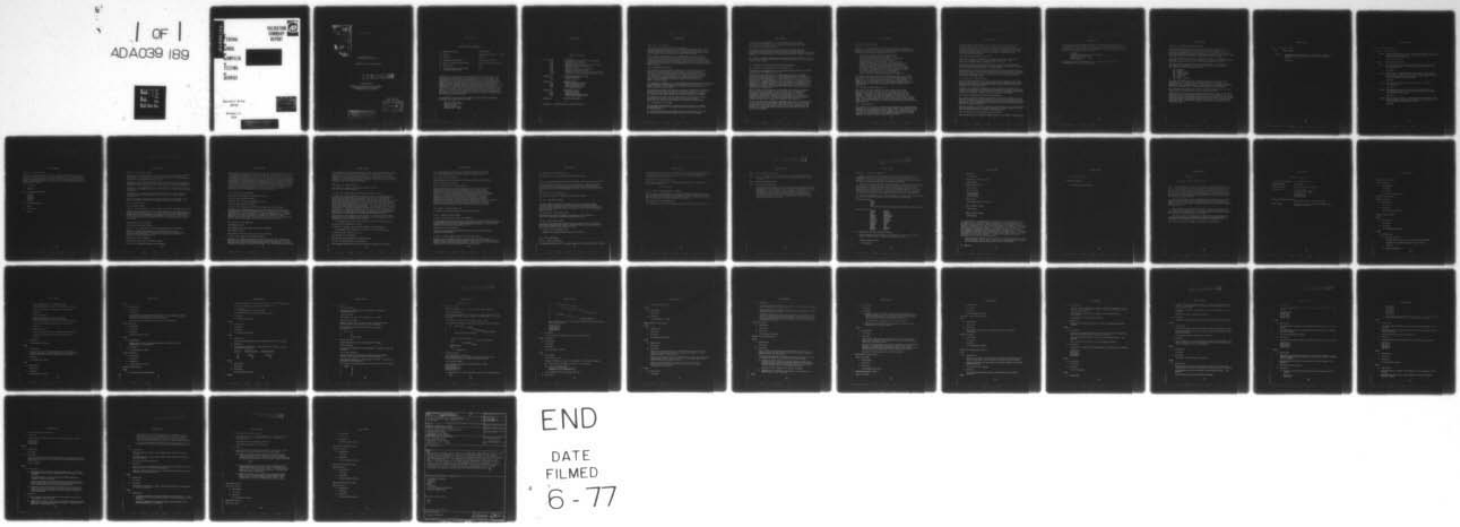
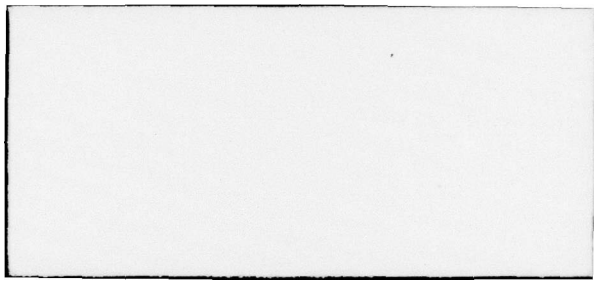
AD-A039 189 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
MAY 77
UNCLASSIFIED CCVS68-VSR220 NL



END

DATE
FILMED
6-77



CCVS68-VSR220

Per: DLL
10 May 77

COBOL COMPILER
VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS68-VSR220

Prepared By:

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

(See form 1473)

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS68-VSR220
2.	Vendor	International Business Machines
3.	Mainframe	360/65
4.	Compiler Identification	COBOL Version 4 R1.4
5.	Operating System Identification	OS/MVT Version 21.8 HASP 3.1
6.	CCVS Version	6.2
7.	Federal Information Processing Standard Publication	21

*PLEASE NOTE. 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 this 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 Federal 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.

For information concerning this compiler you can contact the vendor's designated representative named below:

Mr. J. D. Valentine
Federal Support Center
IBM Corporation
10401 Fernwood Road
Bethesda, MD 20043

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1. Purpose of the VALIDATION Summary Report
The Validation Summary Report (VSR) provides a consolidated summary of the compiler identified in this document. The purpose of the report is to identify individual COBOL Language elements whose usage does not conform to the Federal COBOL Standard as specified in FIPS PUB 21.

The results of running the COBOL Compiler Validation System recorded in the VSR do not suggest the degree to which the compiler is usable (i.e., capable of data processing applications), but the degree to which individual language elements are usable. This will give an indication of possible conversion procedures which may be necessary in order to utilize a source program from another system supporting the Federal Standard for the COBOL language.

1.2. Preparation of the VSR

The Validation Summary Report is prepared by analyzing the results of running the COBOL Compiler Validation System. The COBOL Compiler Validation System consists of audit routines, 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 many tests and supporting procedures indicating the result of the tests. The audit routines collectively contain the features of Federal Standard COBOL.

The testing of a compiler in a particular hardware/operating system environment is accomplished by compiling and executing each audit routine. The output report produced by each routine tells whether the compiler passed or failed the tests 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 showing the discrepancies found. The following briefly describes the content of these sections.

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 the COBOL Standard for the Federal Community. Section 3 of the VSR lists the discrepancies described in Section 2 by the level in which the problem occurs. This readily permits easy determination of the status of the compiler against a particular level.

c. Section 4 contains information discovered during the validation process which does not impact the status of the compiler with respect to the defined Federal COBOL levels.

d. Appendix A is the Validation Summary Working Paper.

1.4. Abstract covering compliance to FIPS PUB 21

Definition of an Implementation of USA Standard COBOL (excerpts from ANSI 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 implmentation 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 COBOL Compiler Validation results enclosed in this document reflect the degree to which the subject COBOL compiler implements the Federal COBOL Standard. The Federal COBOL Standard is essentially the same as the ANSI Standard 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 temporarity dropped from the Federal Standard, however, use of the module is encouraged whenever it is available.

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 the ANSI Standard COBOL X3.23-1968. Not all computers are large enough to accommodate a COBOL compiler containing the full ANSI Standard. Therefore, the Federal Government requires that all compilers acquired by its agencies contain as a minimum one of the four Federal levels, depending on machine size, configuration and user needs. The knowledge that all computers will support at least one of these four subsets simplifies the task of developing machine-independent COBOL programs.

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.

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 Interpretation

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.

1.10 Timeliness of the Validation Summary Reports

The timeliness of the Validation Summary Report is important. Compilers and

their related operating system software are modified several times a year. The Compiler Validation System used to validate compilers is also updated during the life of the system. Therefore to ensure that the latest version of both the vendor's compiler and the Validation System are the latest officially released versions, check with the:

Director
Federal COBOL Compiler Testing Service
Department of the Navy
Washington, D. C. 20376
(202) 697-1247

Please use the Validation Summary Report number of this report when corresponding with the Testing Service.

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 NC210 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. Syntactic Errors

2.1.1. Nucleus module.

2.1.1.1 Character-string continuation. The compiler rejected the continuation of character-strings across more than one line.

See NC205.

2.2 Semantic Errors.

2.2.1 Nucleus Module.

2.2.1.1 The compiler failed to treat a group level move as an alphanumeric to alphanumeric move. Editing was done even though both operands were not elementary.

See NC105 and SQ214

2.2.2 Sequential Access Module.

2.2.2.1 The contents of the first character position of each record of a file destined for the printer, was always blanked out on the hard copy.

See SQ101

2.2.2.2 The compiler failed to successfully execute a subscripted READ...INTO... statement. The subscript was evaluated before the record was read instead of after the record was read, but before the implied move takes place.

See SQ215

2.2.2.3 The compiler failed to reference correctly a qualified 66-level entry.

See SQ215

2.2.2.4 The compiler rejected the figurative constant ZERO in the advancing phrase of a write statement referencing a record of a printer destined file.

See SQ218

2.2.2.5 WRITE mnemonic-name. A write statement referencing top-of-page followed subsequently by a second write referencing top-of-page caused incorrect page ejection.

See SQ218

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
2.2.2.1

3.2. Low-Intermediate Level

2.1.1.1
2.2.2.2
2.2.2.3
2.2.2.4
2.2.2.5

3.3. High-Intermediate

None

3.4. High Level

None

SECTION 4. INFORMATION ITEMS

The COBOL Compiler Validation System contains tests which are not considered as being PASS or FAIL situations. These tests cover the cases where the results of a given statement are not defined in the language specifications. These tests are documented in Section 4.1.

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. These items do not render a compiler nonstandard.

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):

CLOSE-TEST-2 of LB103 closed then reopened the printer file. The result of closing 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 logically repositioned to its beginning and all previous results overwritten or whether the file is partitioned and all output is preserved.

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 being PIC S9(18) VALUE 111111111111111111 and the second having a PIC X(18) VALUE "111111111111111111". The items are compared by the form IF identifer-1 EQUAL TO identifier-2...

The results of this compiler:

The two elementary items did 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 alphanumeric 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 implementors of COBOL compilers as to its representation.

The results for this compiler:

Negative data redefined as alphanumeric data:

The data item tested NOT NUMERIC.

Positive data redefined as alphanumeric 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 tests 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 < 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 = 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 of this compiler:

The word NOT was treated as a logical 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 of this compiler:

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

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 of this compiler are:

The compiler appears to support multiple length records.

4.1.3. Sequential Access Module

4.1.3.1. CLOSE REEL for Output Files (SQ101):

The language specifications are not clear on what is the first record written to a 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 attempts to create a tape file with multiple length records. The file is then read and the record area examined to determine whether fixed length records are written. For a discussion of the operation refer to 4.1.2.2 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):

Audit routine SQ104 creates a sequential mass-storage file with multiple length records. The file is then read and the records are checked for multiple length by evaluating the data in them. Refer to 4.1.2.1 under the 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 multiple length records.

4.1.3.4. FILE-LIMITS Clause

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

The results for this compiler are:

The FILE-LIMITS clause is treated as comments by this compiler in that file allocation is handled through control cards.

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. For this validation the following RERUN Clause option was used.

The results for this compiler are:

RERUN ON UT-S-RERUN EVERY 10 RECORDS OF file-name.

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 a subsequent COPY statement. This information is provided in order to understand the technique used by this system.

The results for this compiler are:

This system requires that the following program entries be duplicated in the library text:

01 record-name

4.2.2. Normal Print Positioning (SQ101):

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

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

4.4 Transparent Implementation

4.4.1 Declarative Procedures.

Labeling of files on this system is handled through external control statements and consequently the COBOL constructs intended for labeling procedures are not utilized by this compiler. However there were cases in which statements either had to be removed for successful execution or produced compiler messages indicating the statement was syntactically incorrect. This included the presence of USE procedures which made reference to uninitialized user label fields defined by the VALUE OF clause.

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:

LOAD
STATE
TRUNC
LIB
OPTIMIZE/NOOPTIMIZE (Both options were used at different times)

Options defaulted:

SPACE1	NONUM
FLAGW	NOBATCH
SEQ	NONAME
SOURCE	COMPILE=01
NODMAP	NORESIDENT
NOPMAP	NODYNAM
NOCLIST	NOSYNTAX
NOSUPMAP	NOSYMDMP
NOXREF	NOTEST
NOSXREF	VERB
NODECK	ZWB
APOST	SYST
NOFLOW	NOENDJOB
NOTERM	NOADV

2. Environment Division implementor-names.

For the purposes of this discussion the suffix 'ddname' is the external name by which the file is known to the operating system.

Printer destined files

UT-S-ddname

Tape files

UT-S-ddname

Sequential Mass-storage files

UT-S-ddname

Random Access files

UT-S-ddname

Sort files (SD)

UT-S-SORTWK01
UT-S-SORTWK02
UT-S-SORTWK03

Switch names

Not supported by the compiler.

Source Computer names

IBM-360-I65

Object Computer names

IBM-360-I65

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 call statement. See 1. above. There was no differnce in the execution of the programs when the optimization feature was invoked.

4. Compiler.

COBOL Version 4 P1.4

5. Operating system.

OS/MVT Version 21.8 HASP 3.1

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 descrepancy and as such is reflected only once in the body of the VSR (The VSR will in turn only reference the first occurance of the problem in the appendix.)

This appendix is divided into two parts. The first part describes the Validation Environment which includes configuration, operating system, and compiler information as well as the input parameter and options used by the VP-Routine to accomplish th e validation.

The second part of th document is divided into two categories of information: compilation and execution results. Information items are referenced only in section 4 of the VSR and are not repeated here.

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

CCVS68-VSR220

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	COBOL Version 4 R1.4
COMPUTER SYSTEM:	IBM 360/65
OPERATING SYSTEM:	OS/MVT Version 21.8 HASP 3.1
CONFIGURATION:	28 DISK DRIVES 01 7 TRACK TAPE DRIVES 07 9 TRACK TAPE DRIVES 02 PRINTERS 01 CARD READER 01 CARD PUNCH
COMPILER AND SYSTEM OPTIONS:	LOAD, STATE, NOMAP, TRUNC, OPT, NOOPT
COBOL MANUAL:	IBM OS Full American National Standard COBOL, Revision, GC28-6396-3

LIBRARY MODULE LEVEL 1

LB101 thru LB107

A. Compilation

No errors.

B. Execution

No discrepancies noted.

LIBRARY MODULE LEVEL 2

LB201 thru LB205

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NUCLEUS MODULE LEVEL 1

NC101

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC102

A. Compilation

1. Warning messages:

"Period preceded by space. Assume end of sentence."

flagged the following construct in NOTE-TEST-4:

"NOTE $."

2. Warning messages:

"Comma should not be preceded by space."
"Comma not followed by space. Assumed."
"Period preceded by space. Assume end of sentence."

flagged the following construct in NOTE-TEST-4:

"NOTE ,."

3. Warning messages:

"Comma should not be preceded by space."
"Comma not followed by space. Assumed."
"Period preceded by space. Assume end of sentence."

flagged the following construct in NOTE-TEST-4:

"NOTE ;."

4. Warning message:

"Period preceded by space. Assume end of sentence."

flagged the following construct in NOTE-TEST-4:

"NOTE ."

B. Execution

No discrepancies noted.

NC103

A. Compilation

All references to hardware switches had to be deleted to avoid fatal errors, as hardware switches were not implemented. SWH-TEST-1 through SWH-TEST-2 were deleted.

B. Execution

No discrepancies noted.

NC104

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC105

A. Compilation

No errors.

B. Execution

MOVE-TEST-76 and MOVE-TEST-92 failed when a group level move was not treated as an alphanumeric to alphanumeric move. Editing was done even though both operands were not elementary.

See page 2-38, paragraph 6.13.4 (5), the MOVE statement.

NC106 and NC107

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC108

A. Compilation

ABBREV-TEST-9 had to be deleted as hardware switches were not implemented in this compiler.

B. Execution

No discrepancies noted.

NC109 thru NC113

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NUCLEUS MODULE LEVEL 2

NC201

A. The following E-level message:

ILLEGAL COMPARISON OF TWO NUMERIC LITERALS. STATEMENT DISCARDED.

flagged the following construct in IF-TEST-119:

```
IF 2 + 2 = 4
    PERFORM PASS GO TO IF--WRITE-119.
```

IF-TEST-119 had to be deleted for that reason.

B. Execution

No discrepancies noted.

NC202

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC203

A. Compilation

Several warning messages of an informative nature were flagged.

B. Execution

COMP-TEST-40, COMP-TEST-41, and COMP-TEST-42 failed in testing the COMPUTE statement.

Test No.	Computed Results	Expected Results
40	99320	108140
41	0	1
42	0	1

NC204

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC205

A. Compilation

1. C-level message:

TO MISSING OR MISPLACED IN MOVE STATEMENT. ASSUMED IN REQUIRED POSITION.

W-level message:

SUPERFLUOUS TO FOUND IN MOVE STATEMENT. IGNORED.

and E-level message:

NUMERIC LITERAL (ID) MAY NOT BE TARGET FIELD FOR NUMERIC LITERAL (ID) IN MOVE STATEMENT, AND IS DISCARDED.

all flagged the following broken-line construct in CONTIN-TEST-1:

```
    "MOVE    4
-            5
-            6
-            7
-            8 TO CONT-B."
```

2. W-level message:

PERIOD PRECEDED BY SPACE. ASSUME END OF SENTENCE.

C-level message:

TO MISSING OR MISPLACED IN MOVE STATEMENT. ASSUMED IN REQUIRED POSITION.

and E-level messages:

NUMERIC LITERAL (ID) MAY NOT BE TARGET FIELD FOR NUMERIC LITERAL (ID) IN MOVE STATEMENT, AND IS DISCARDED.

EXPECTING NEW STATEMENT. FOUND NUMERIC LITERAL. DELETING TILL NEXT VERB OR PROCEDURE-NAME.

all flagged the following broken-line construct in CONTIN-TEST-2:

```
     "MOVE    -
-             9
-             9
-             9
-             .
-             7
-             7
```

```
-            7 TO CONT-B."
```

3. C-level message:

```
ILLEGAL CHARACTER.  SCAN RESUMED AT NEXT VALID CHARACTER.
```

and F-level messages:

```
FOUND 'NUMERIC LITERAL' AFTER CONDITION.  EXPECT 'OR', 'AND' OR
VERB TO IMMEDIATELY FOLLOW CONDITION.  DELETING TILL ONE OF THESE
IS FOUND.

I NOT DEFINED.  DELETING TILL LEGAL ELEMENT FOUND.
```

all flagged the following broken-line construct in CONTIN-TEST-4:

```
     IF    CONT-E EQUA
-                  L TO ZERO
-                        S AN
-                           D GREATER
-                                    ZERO AND CONT-B
           EQUAL TO CONT-C OR (((((((D


      - CONT-D EQUAL TO CONT-D O
-                                 R -11 + CONT-F))))))
      AND N
-                  OT NEGATIVE
                              ZERO
      PERFORM PASS
      EL
-       SE
            PERFORM FAIL.
```

4. W-level message:

```
DIVID SHOULD BEGIN A-MARGIN.
ILLEGAL CHARACTER IN COL 7 PRECEDING DIVID PARAGRAPH NAME.
END OF SENTENCE SHOULD PRECEDE E.  ASSUMED PRESENT.
```

and E-level messages:

```
E NOT DEFINED.  DELETING TILL LEGAL ELEMENT FOUND.
GIVIN NOT DEFINED.
G NOT DEFINED.
ROUNDE NOT DEFINED.
D NOT DEFINED.
```

all flagged the following broken-line construct in CONTIN-TEST-5:

```
    "DIV
```

```
-         ID
-            E CONT-E IN
-                        TO CONT-C GIV
-                   IN
-                       G CONT-D ROUN
-                                    DE
-                                        D  O
-                                             N SIZE ERR
-         OR PERFOR
-                       M PASS G
-                                O  T
-                                     O CONTIN-WRITE-5."
```

As a result of the above errors, the following tests were deleted from the final run:

CONTIN-TEST-1
CONTIN-TEST-2
CONTIN-TEST-4
CONTIN-TEST-5

B. Execution

No discrepancies encountered in the non-deleted tests.

NC206 thru NC210

A. Compilation

No errors.

B. Execution

No discrepancies noted.

NC211

A. Compilation

The E-level error:

ILLEGAL COMPARISON OF TWO NUMERIC LITERALS. STATEMENT DISCARDED.

was flagged on the following construct in CC-TEST-34:

```
IF 1 + 2 + 3 IS LESS THAN 7
    OR NOT 6 IS LESS THAN 1 + 2 + +4
    PERFORM PASS ELSE PERFORM FAIL.
```

Test CC-TEST-34 was subsequently deleted.

B. Execution

No discrepancies noted.

NC212

A. Compilation

No errors.

B. Execution

No discrepancies noted.

RANDOM ACCESS MODULE LEVEL 1

RC101

A. Compilation

No errors.

B. Execution

No discrepancies noted.

RC102

A. Compilation

No errors.

B. Execution

(Run 1) Run terminated following MULT-TEST-10 while attempting to open for output the second of two random access files. System completion code 30A was flagged.

(Run 2) The BLOCK CONTAINS ... CHARACTERS clause for the second file was deleted for this run, however, the run terminated after MULT-TEST-12 with an 0C7 completion code.

(Run 3) For the program to execute successfully Information Tests MULT-TEST-13 and MULT-TEST-14 were deleted. No other discrepancies were noted.

RC103

A. Compilation

No errors.

B. Execution

TEST-16 failed on INVALID KEY condition on a WRITE statement which was preceeded by a SEEK statement.

This problem was the result of not allowing enough space on a mass storage device for the file.

The FILE-LIMITS clause appears to be treated as comments by the compiler. The function of determining the logical file limits is provided through the job control language. TEST-16 failed because the amount of area allocated was inconsistent with logical space required for the test.

RC104 thru RC105

A. Compilation

No errors.

B. Execution

No discrepancies noted.

RANDOM ACCESS MODULE LEVEL 2

RC201

A. Compilation

No errors.

B. Execution

(Run 1) This run terminated following TEST-01 while attempting to open for output the second of two random access files. System completion code 30A was flagged.

(Run 2) For the program to execute successfully the BLOCK CONTAINS ...CHARACTERS clause for the second file was eliminated.

1. TEST-01, TEST-02, TEST-09, TEST-10, TEST-11, TEST-25, and TEST-26 failed. These tests involve execution of declarative procedures following the opening and closing of two random access files for input, input-output, and output.

   See page 2-179, paragraph 4.5, the USE statement.

2. TEST-16 failed when 200 was considered an invalid key for RANDOM-FILE-1. Refer to RC103.

RC202

A. Compilation

No errors.

B. Execution

1. TEST-01, TEST-02, TEST-09, TEST-10, TEST-11, TEST-27, and TEST-28 failed. These tests involve execution of declarative procedures following the opening and closing of two random access files for input, input-output, and output.

   See page 2-179, paragraph 4.5, the USE statement.

2. TEST-16 failed when 200 was considered an invalid key for RANDOM-FILE-1. Refer to RC103.

RC203

A. Compilation

No errors.

B. Execution

(Run 1) This run terminated following TEST-15 with an IKF111I status code indicating a WRITE, NO REC FOUND and a 00000CE6000D00 BDAM code. All tests failed.

(Run 2) For this program to execute successfully, LABEL-TEST-16 was deleted. All other tests also failed. These tests involve execution of declarative procedures before and after beginning and ending labels are processed for three random access files.

See page 2-179, paragraph 4.5, the USE statement.

SEGMENTATION MODULE LEVEL 1

SG101 thru SG103

A. Compilation

No errors.

B. Execution

No discrepancies noted.

SEGMENTATION MODULE LEVEL 2

SG201 thru SG203

A. Compilation

No errors.

B. Execution

No discrepancies noted.

SEQUENTIAL ACCESS MODULE LEVEL 1

SQ101

A. Compilation

No errors.

B. Execution

The contents of column 1 of the printer report was always suppressed.

SQ102 thru SQ108

A. Compilation

No errors.

B. Execution

No discrepancies noted.

SEQUENTIAL ACCESS MODULE LEVEL 2

SQ201

A. Compilation

Wherever Declarative procedures were specified in conjunction with file or reel labels, the following E-level message was flagged:

USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

Declaratives were deleted.

B. Execution

All tests involving declaratives failed because of the above deletions.

SQ202

A. Compilation

Wherever Declarative procedures were specified in conjunction with "AFTER ... LABEL PROCEDURE ON INPUT or OUTPUT" the following E-level message was flagged:

USE STATEMENTS IMPLY BOTH STANDARD AND NON-STANDARD LABELS.
USE IGNORED.

Declaratives were subsequently deleted.

B. Execution

Because of the above deletions, all tests involving declaratives failed.

SQ203

A. Compilation

Wherever USE BEFORE STANDARD LABEL PROCEDURE ON INPUT or OUTPUT was used, the following E-level message was flagged:

USE STATEMENTS IMPLY BOTH STANDARD AND NON-STANDARD LABELS. USE IGNORED.

USE-2 and USE-4 declaratives were subsequently deleted.

B. Execution

Because of the deletion of declaratives, the following tests failed:

USE-TEST-2
USE-TEST-4
USE-TEST-5
USE-TEST-6
USE-TEST-8

SQ204

A. Compilation

No errors.

B. Execution

No discrepancies.

SQ205

A. Compilation

Wherever Declarative procedures were specified in conjunction with "AFTER ... LABEL PROCEDURE ON file-name", the following E-level message was flagged:

USE STATEMENTS IMPLY BOTH STANDARD AND NON-STANDARD LABELS. USE IGNORED.

Declaratives were subsequently deleted.

B. Execution

Because of above deletions all tests involving declaratives failed.

SQ206

A. Compilation

All USE statements were flagged with the following E-level message:

USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

All declaratives were subsequently deleted.

B. Execution

This run terminated before 01-USE-TEST with a S613 completion code while attempting to open for output the third of three random access files. No output report was generated.

SQ207

A. Compilation

No errors.

B. Execution

Suppressed by SQ206 failure to execute.

SQ208

A. Compilation

Two Declarative procedures specifying "USE AFTER ... FILE LABEL PROCEDURE ON I-O" cause the following E-level message to be flagged:

USE STATEMENTS IMPLY BOTH STANDARD AND NON-STANDARD LABELS. USE IGNORED.

These declaratives were subsequently deleted.

B. Execution

Because of deletion of the above declaratives, the following tests failed:

OPEN-I-O-TEST
07-USE-TEST
08-USE-TEST
09-USE-TEST
10-USE-TEST

SQ209

A. Compilation

All declarative procedures were flagged with the following E-level message:

USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

All declaratives were deleted.

B. Execution

Because of the above mentioned deletions the following tests failed.

OPEN-TEST-1
CLOSE-TEST-1
CLOSE-TEST-2
CLOSE-TEST-3

SQ210

A. Compilation

Wherever declarative procedures were specified for "AFTER ... UNIT LABEL PROCEDURE ON INPUT or OUTPUT" were flagged with the following E-level message:

USE STATEMENTS IMPLY BOTH STANDARD AND NON-STANDARD LABELS. USE IGNORED.

Declaratives were deleted.

B. Execution

1. Because of the above mentioned deletions the following tests failed:

USE-TEST-2
USE-TEST-3

USE-TEST-4
USE-TEST-5
USE-TEST-6
USE-TEST-7
USE-TEST-8
USE-TEST-9

2. WRITE-TEST-1 failed when mass-storage file was able to write more records than FILE-LIMITS clause should have permitted.

SQ211

A. Compilation

All declarative procedures were flagged with the following E-level message:

USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

Declaratives were deleted subsequently.

B. Execution

Because of deletion of declaratives the following tests failed:

USE-TEST-1
USE-TEST-2
USE-TEST-3
USE-TEST-4
USE-TEST-5

SQ212

A. Compilation

No errors.

B. Execution

No discrepancies noted.

SQ213

A. Compilation

All declarative procedures were flagged with the following E-level message:

USE SPECIFIED FOR file-name WITH RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

All declaratives were deleted.

B. Execution

As a result of the above deletion, the following tests failed:

01-USE-TEST
07-USE-TEST
09-USE-TEST

SQ214

A. Compilation

No errors.

B. Execution

The WRITE FROM 021-TEST failed when a WRITE alphanumeric edited item FROM a group level item was not treated as an alphanumeric to alphanumeric move.

Refer to NC105.

SQ215

A. Compilation

1. All declarative procedures specifying USE AFTER ... FILE LABEL PROCEDURE ON file-name were flagged with the following E-level messages:

   USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

   These declaratives were subsequently deleted along with DECLAR-TEST-6 and DECLAR-TEST-7 which referenced declarative section and paragraph names from outside the DECLARATIVES.

2. Numerous E-level messages were flagged for line continuation constructs. These constructs were modified for subsequent run. Refer to NC205 for description of general remarks concerning line continuation.

B. Execution

1. As a result of the above mentioned deletions all declarative tests failed or were deleted.

2. LABEL-TEST-1 failed. This test involves the USE BEFORE STANDARD BEGINNING LABEL PROCEDURE ON INPUT and OUTPUT. And tests the VALUE OF ... qualified clause.

3. READ-TEST-1 and READ-TEST-2 failed when they were unable to execute successfully a subscripted READ ... INTO ... statement. The subscript was contained inside the record. The subscript value was evaluated before the record was read and thus the wrong value was used when the record was moved by the INTO phrase.

4. The run terminated during 66-TEST-1 with an 0C7 completion code. An attempt was being made to reference a qualified 66-level entry.

SQ216

A. Compilation

All declarative procedures were flagged by the following E-level message:

USE SPECIFIED FOR file-name WITH LABEL RECORDS OMITTED OR STANDARD. SENTENCE IGNORED.

All declaratives were deleted.

B. Execution

(Run 1) The run terminated abnormally following LOCK-TEST-3 with a B37 completion code. LOCK-TEST-4 was deleted.

(Run 2) LOCK-TEST-1, LOCK-TEST-2, and LOCK-TEST-3 failed due to the deletion of all declaratives.

SQ217

A. Compilation

No errors.

B. Execution

LOCK-TEST-7 failed when an input file was specified by a USE procedure as incorrectly opened.

SQ218

A. Compilation

1. All WRITE statements for a file assigned to the system printer and which did not have the ADVANCING option, drew the following E-level message:

ALL WRITE STATEMENTS FOR DNM=3-52 SHOULD HAVE ADVANCING OPTION. BEFORE ADVANCING 1 LINE ASSUMED.

2. The following C-level message:

'ADVANCING' OPTION MUST BE FOLLOWED BY MNEMONIC NAME, NUMERIC INTEGRAL DATA-NAME OR INTEGER LESS THAN 100. FOUND ZERO. SUBSTITUTING 1.

was flagged for the following construct:

WRITE PRINT-REC BEFORE ADVANCING ZERO.

B. Execution

1. Many test failures could be attributed to the compiler action noted above. Those which would not are shown below:

(a) WRITE-TEST-7 which checks consecutive AFTER ADVANCING 0 LINES statements and BEFORE ADVANCING 0 LINES statements failed due to A(2) above. The word "PASS" which should have appeared with all letters on the same line instead appeared:

ASS
P

(b) WRITE-TEST-8 failed. This test issues commands to "WRITE BEFORE ADVANCING mnemonic-name" followed immediately by "WRITE AFTER ADVANCING mnemonic-name". The implementor-name specified was for top-of-page positioning. No blank page appeared between printed page.

(c) WRT-TEST-9 failed. This test issues commands to "WRITE FROM ... BEFORE mnemonic-name" followed immediately by "WRITE FROM ... AFTER ADVANCING mnemonic-name". Test results were the same as in WRITE-TEST-8 which failed.

SORT MODULE LEVEL 1

ST101 thru ST114

A. Compilation.

No errors.

B. Execution

No discrepancies noted.

SORT MODULE LEVEL 2

ST201 thru ST207

A. Compilation

No errors.

B. Execution

No discrepancies noted.

TABLE HANDLING MODULE LEVEL 1

TH101 thru TH104

A. Compilation

No errors.

B. Execution

No discrepancies noted.

TABLE HANDLING MODULE LEVEL 2

TH201 thru TH211

A. Compilation

No errors.

B. Execution

No discrepancies noted.

TABLE HANDLING MODULE LEVEL 3

TH301 thru TH311

A. Compilation

No errors.

B. Execution

No discrepancies noted.

BIBLIOGRAPHIC DATA SHEET	1. Report No. (14) CCVS68-VSR220	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report #CCVS68-VSR120 IBM OS COBOL (6) COBOL Compiler Validation Summary Report.			5. Report Date (11) 4 May 1977 6.
7. Author(s) Same as organization - see 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal COBOL Compiler Testing Service ADP Selection Office Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address ADP Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered (12) 45p 14.

15. Supplementary Notes

16. Abstracts

This Validation Summary Report (VSR) for the IBM OS/VS COBOL Compiler Version 4 R1.4 (OS/MVT Version 21.8/HASP 3.1) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1968 COBOL Standard (X3.23-1968/FIPS PUB 21). The compiler was validated at the High level of FIPS PUB 21. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1968; a section relating the categories of discrepancies to each of the Federal levels of the language; and a detailed listing of discrepancies together with the tests which were failed.

17. Key Words and Document Analysis. 17a. Descriptors

Programming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS

17c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	43
	20. Security Class (This Page) UNCLASSIFIED	22. Price