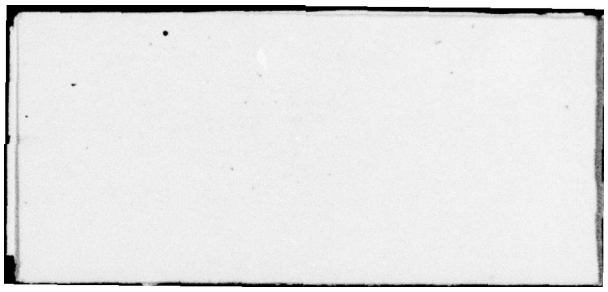
AD-A050 282 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
FEB 78

UNCLASSIFIED CCVS68-VSR280 NL

| OF |
AD A050282

END
DATE FILMED
3-78
DDC



(10)

(6) COBOL COMPILER
VALIDATION SUMMARY REPORT.

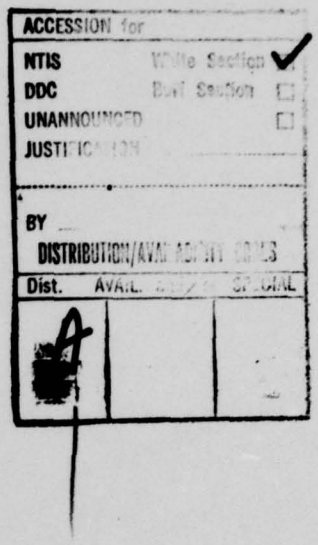
(14) VALIDATION NUMBER CCVS68-VSR280

(11) 15 Feb 78 (12) 34p.

Prepared By:

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

408 438



COBOL COMPILER VALIDATION

1.	Validation Number	CCVS68-VSR280
2.	Vendor	HISI
3.	Mainframe	H2051A
4.	Compiler Identification	OS/2000 ANS COBOL Version 4.02
5.	Operating System Identification	OS/2000
6.	CCVS Version	6.2
7.	Federal Information Processing Standard Publication	21

8. For additional information on this compiler contact:

Dr. Clair Miller
HISI, Honeywell Center
7900 Westpark Drive
McLean, Virginia 22101

TABLE OF CONTENTS

SECTION 1.		INTRODUCTION	1
	1.1	Purpose of the Validation Summary Report	1
	1.2	Preparation of the VSR	1
	1.3	Organization of the VSR	1
	1.4	Abstract Covering Compliance to FIPS PUB 21	2
	1.5	Federal Standard COBOL Levels	4
	1.6	Use of the VSR	4
	1.7	Sources of Additional Information	4
	1.8	Requests for Interpretation	5
	1.9	Federal Standard COBOL Approved Interpretation	5
	1.10	Timeliness of the Validation Summary Report	5
SECTION 2.		DETAILED EVALUATION OF ERRORS	6
	2.1	Syntactical Errors	7
	2.2	Semantic Errors	8
SECTION 3.		COMPILER STATUS	9
	3.1.	Low Level (Minimum COBOL)	9
	3.2	Low-Intermediate Level	9
	3.3	High-Intermediate Level	9
	3.4	High Level (Full Standard Level)	9
SECTION 4.		INFORMATION ITEMS	10
	4.1	Information Tests	10
	4.2	Other Information	14
	4.3	Hardware Dependent Features	15
	4.4	Transparent Implementation	15
SECTION 5.		SOFTWARE ENVIRONMENT	16
APPENDIX A - VALIDATION SUMMARY WORKING DOCUMENT			18

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 additonal 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 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 Library Module.

2.1.1.1 The compiler rejected identifiers containing either qualifiers or subscripts in the REPLACING phrase of a COPY statement.

See LB202.

2.1.2 Nucleus Module.

2.1.2.1 The compiler rejected a DISPLAY statement containing twenty-one data-names as operands.

See NC109.

2.1.3 Random Access Module.

2.1.3.1 The compiler required that a LABEL RECORD OMITTED clause be changed to LABEL RECORD STANDARD in a file description entry for a file whose access mode is random.

See RC102.

2.1.3.2 The compiler rejected a LABEL RECORD data-name clause in a file description entry for a file whose access mode was random. This compiler does not support user defined labels for the random access module.

See RC203.

2.1.4 Table Handling Module.

2.1.4.1 The compiler rejected an index-data-item (USAGE INDEX) as the subject of a VARYING phrase of a SEARCH statement for an indexed table.

See TH309.

2.2 Semantic Errors

2.2.1 Nucleus Module.

2.2.1.1 The system truncates strings of characters DISPLAYed on the system console to 64 characters instead of providing additional lines as necessary to exhibit all of the data transferred in a DISPLAY statement.

See NC204.

2.2.2 Random Access Module.

This module was not executed. The semantic errors described below were found in testing Version 2.01 of the OS/2000 ANS COBOL compiler. References are to validation summary report number 73-C011.

2.2.2.1 The system does not support the concept of user defined labels (LABEL RECORD is data-name) on mass-storage files.

See 73-C011 RC203.

2.2.3 Sequential Access Module.

2.2.3.1 For a mass-storage sequential file which had been OPENed for I-O, the ending declaratives were executed when the file was CLOSEd, even though the end of the file had not been reached.

See SQ209.

2.2.3.2 For a sequential tape file which had been OPENed for INPUT, the ending file declaratives were executed when the end of the file was reached, but without a CLOSE command having been issued.

See SQ211.

2.2.3.3 For the READ/RETURN with the INTO phrase, the move from the record area into the identifier was always treated as a group move regardless of the data items involved.

See SQ214/ST207.

2.2.3.4 The system reserves the first five characters of user-defined file labels for its own use.

See SQ215.

2.2.4 Sort Module.

2.2.4.1 RETURN INTO - Refer to 2.2.3.3

See ST207.

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

2.1.2.1 DISPLAY statements with 21 operands was rejected.

3.2. Low-Intermediate Level

2.1.3.1 LABEL RECORDS STANDARD are required for random access files. (See also 2.2.2.1)

2.1.3.2 The LABEL RECORD data-name clause is not permitted for mass-storage files.

2.2.1.1 DISPLAY has a maximum of 64 characters on the system console.

2.2.3.1 There is a problem with the Ending USE procedures for sequential mass-storage files.

2.2.3.2 There is a problem with the Ending USE procedures for sequential tape files.

2.2.3.3 READ INTO functions like a group MOVE.

2.2.3.4 User labels on tape are modified by the system.

3.3. High-Intermediate

None

3.4. High Level

2.1.1.1 Identifiers with subscripts and qualifiers are not replaced by the COPY statement.

2.1.4.1 Indexed data items are not permitted in the SEARCH statement.

2.2.4.1 RETURN INTO functions like a group MOVE.

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 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 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 rightmost character(s) are truncated when the number of characters ACCEPTed from the input device exceeds the size of the receiving field.

4.1.2. Random Access Module (determined from a previous validation)

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.

4.1.2.2 CLOSE . . . WITH LOCK (RC201):

The language specification indicates that the results of closing a file with lock is that the file cannot be accessed again for the duration of the current run unit. The results of any attempt to access the file are left to the implementor.

The results for this compiler are:

The run unit terminated at the time an access to the file was attempted.

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 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.

The RERUN clause used for this compiler was:

```
RERUN EVERY 10 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.

No standard hardware dependent features were restricted or limited by this compiler.

4.4 Transparent Implementation

None were noted.

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: INSERT LN

Options defaulted: None

2. Environment Division implementor-names.

Printer destined files	PRINTER-SPR
Tape files	TAPE-UNIT-MWn
Sequential Mass-storage files	MSD-MRn
Random Access files	MSD-MRn
Sort files (SD)	MSD-MR7
Switch names	SENSE-SWITCH-1 SENSE-SWITCH-2
Source Computer names	H-2000-SPECIAL
Object Computer names	H-2000

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 was not tested.

4. Compiler.

OS/2000 ANS COBOL Version 4.02

5. Operating system.

OS/2000

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: OS/2000 ANS COBOL Version 4.02

COMPUTER SYSTEM: H2051A

OPERATING SYSTEM: OS/2000

LIBRARY MODULE LEVEL 1

LB101 through LB107

A. Compilation:

No errors.

B. Execution:

No errors.

LIBRARY MODULE LEVEL 2

LB201

A. Compilation:

No errors.

B. Execution:

No errors.

LB202

A. Compilation:

In COPY-TEST-17, attempts to replace datanames by qualified datanames or by subscribted datanames elicited fatal syntax errors.

B. Execution:

COPY-TEST-17 was deleted.

LB203 through LB205

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:

In DISP-TEST-12, a DISPLAY statement with 21 subscripted data-name operands elicited the following fatal error message:

TOO MANY DATA NAMES IN THIS STATEMENT.

B. Execution:

DISP-TEST-12 was deleted.

NC110 through NC113

A. Compilation:

No errors.

B. Execution:

No errors.

NUCLEUS MODULE LEVEL 2

NC201

A. Compilation:

In IF--TEST-119, the following construct:

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

was flagged with the following fatal error message:

IT IS ILLEGAL TO COMPARE TWO LITERALS WITH EACH OTHER.

X3.23-1968, 2-2-5.2.1, does indeed prohibit comparison of two literals, but places no restriction on comparison of an arithmetic expression to a literal as above.

B. Execution:

IF--TEST-119 was deleted..

NC202 through NC203

A. Compilation:

No errors.

B. Execution:

No errors.

NC204

A. Compilation:

No errors.

B. Execution:

DISP-TEST-10 and DISP-TEST-14 both fail because the data to be DISPLAYED on the CONSOLE is truncated 64 characters. When the operand or composite of operands exceeds 64 characters, DISPLAY should use the number of lines necessary to exhibit the entire length. Only one line of data was actually DISPLAYed.

NC205 through NC210

A. Compilation:

No errors.

B. Execution:

No errors.

NC211

A. Compilation:

In CC-TEST-34, the following construct:

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

was flagged with the following fatal error message:

IT IS ILLEGAL TO COMPARE TWO LITERALS WITH EACH OTHER.

X3.23-1968, 2-2-5.2.1, does indeed prohibit comparison of two literals, but places no restriction on comparison of an arithmetic expression to a literal and vice versa as above.

B. Execution:

CC--TEST-34 was deleted.

NC212

A. Compilation:

No errors.

B. Execution:

No errors.

RANDOM ACCESS MODULE LEVEL 1 and LEVEL 2

Because the proper form of the ACTUAL KEY clause could not be determined, this module was not run.

SEGMENTATION MODULE LEVEL 1

SG101 through SG103

A. Compilation:

No errors.

B. Execution:

No errors.

SEGMENTATION MODULE LEVEL 2

SG201 through SG203

A. Compilation:

No errors.

B. Execution:

No errors.

SEQUENTIAL ACCESS MODULE LEVEL 1

SQ101 through SQ108

A. Compilation:

No errors.

B. Execution:

No errors.

SEQUENTIAL ACCESS MODULE LEVEL 2

SQ201 through SQ204

A. Compilation:

No errors.

B. Execution:

No errors.

SQ205

A. Compilation:

No errors.

B. Execution:

The following tests failed:

LABEL-TEST-14
LABEL-TEST-15
LABEL-TEST-16
LABEL-TEST-18

These tests employ declarative procedures with the following USE statements:

USE AFTER BEGINNING FILE LABEL PROCEDURE...
USE AFTER STANDARD ENDING LABEL PROCEDURE...

SQ206 through SQ208

A. Compilation:

No errors.

B. Execution:

No errors.

SQ209

A. Compilation:

No errors.

B. Execution:

CLOSE-TEST-2 and CLOSE-TEST-3 failed. Indications were that in both cases, the USE-7 SECTION of the declarative procedures was executed while closing MASS-FILE-B UNIT. MASS-FILE-B had been OPENed for INPUT. X3.23-1968, 2-7-4.1.4(2)F states that ending reel/unit label procedures are engaged for output files, but omits that specification for input and input-output files.

CLOSE-TEST-5 failed. Indications were that USE-10 SECTION of the declarative procedures was executed while closing DUMMY-I-O-FILE which had been OPENed for I-O, but had not been READ from. X3.23-1968, 2-7-4.1.4.(2)C states in part for I-O files:

"...If the file is positioned other than at its end, the closing

operations specified by the implementor are executed, but there is no ending label processing. An input file, or an input-output file, is considered to be at the end of the file if the imperative statement in the AT END phrase of the READ statement has been executed and no CLOSE statement has been executed".

SQ210

A. Compilation:

No errors.

B. Execution:

WRITE-TEST-1 failed. This indicates that 91 records were able to be written on a file where the FILE-LIMITS ARE 1 THRU 90.

SQ211

A. Compilation:

No errors.

B. Execution:

USE-TEST-2 failed. Indications were that ENDING FILE LABEL PROCEDUREs were executed for SAME-FILE-2 when an end-of-file condition was reached, even though no CLOSE statement had been executed for the file.

SQ212 through SQ213

A. Compilation:

No errors.

B. Execution:

No errors.

SQ214

A. Compilation:

No errors.

B. Execution:

Fifteen of the READ-INTO tests failed. These tests all involve reading a file record into an identifier. X3.23-1968, 2-7-4.3.4.(4) and (5) gives the basic rules for this operation. For each test which failed, the file record and INTO-identifier descriptions, and the actual and expected results are given.

(1) 01-READ-INTO-TEST

file-record	9(10)
record-data	1111001111

```
     INTO identifier          9(18)
          Actual results      1111001111bbbbbbbb
          Expected results    000000001111001111

(2)  02-READ-INTO-TEST

     file-record              9(10)
          record-data         2222002222
     INTO-identifier          9(5)
          Actual results      22220
          Expected results    02222

(3)  07-READ-INTO-TEST

     file-record              9(10)
          record-data         7777007777
     INTO-identifier          X(18)B0
          Actual results      7777007777bbbbbbbbbb
          Expected results    7777007777bbbbbbbbb0

(4)  11-READ-INTO-TEST

     file-record              9(10)
          record-data         0000000000
     INTO-identifier          $99,999,999.99CR BLANK WHEN ZERO.
          Actual results      0000000000bbbbbb
          Expected results    bbbbbbbbbbbbbbb

(5)  24-READ-INTO-TEST

     file-record              X(18) JUST
          record-data         000000001234567890
     INTO-identifier          9(10)
          Actual results      0000000012
          Expected results    1234567890

(6)  27-READ-INTO-TEST

     file-record              S9(9)V9 COMP SYNC
          record-data         111111111v9
     INTO-identifier          9(10)
          Actual results      1111111119
          Expected results    0111111111

(7)  28-READ-INTO-TEST

     file-record              S9(9)V9 COMP SYNC
          record-data         222222222v8
     INTO-identifier          9(10) COMPUTATIONAL SYNCHRONIZED
          Actual results      2222222228
          Expected results    2222222222

(8)  29-READ-INTO-TEST

     file-record              S9(9)V9 COMP SYNC
          record-data         333333333v7
     INTO-identifier          $$$$$$$$$$$$$$$$.$$
```

```
          Actual results        333333333Gbbbbbbbbbb
          Expected results      bbbbbbb$333333333.70

(9)  30-READ-INTO-TEST

     file-record                S9(9)V9 COMP SYNC
          record-data           444444444v6
     INTO-identifier            $99,999,999.99CR BLANK WHEN ZERO
          Actual results        444444440bbbbbb
          Expected results      $44,444,444.60CR

(10) 31-READ-INTO-TEST

     file-record                S9(9)V9 COMP SYNC
          record-data           555555555v5
     INTO-identifier            $99,999,999.99CR BLANK WHEN ZERO
          Actual results        5555555550bbbbbb
          Expected results      $55,555,555.50bb

(11) 32-READ-INTO-TEST

     file-record                S9(9)V9 COMP SYNC
          record-data           000000000v0
     INTO-identifier            $99,999,999.99CR BLANK WHEN ZERO
          Actual results        000000000+bbbbbb
          Expected results      bbbbbbbbbbbbbbb

(12) 33-READ-INTO-TEST

     file-record                X(10)
          record-data           1020304050
     INTO-identifier            9(18)
          Actual results        1020304050bbbbbbbb
          Expected results      000000001020304050

(13) 34-READ-INTO-TEST

     file-record                X(10)
          record-data           6070809000
     INTO-identifier            9(5)
          Actual results        60708
          Expected results      09000

(14) 35-READ-INTO-TEST

     file-record                X(10)
          record-data           ABCKEFGHIJ
     INTO-identifier            X(18)B0
          Actual results        ABCKEFGHIJbbbbbbbbbb
          Expected results      ABCKEFGHIJbbbbbbbbb0

(15) 36-READ-INTO-TEST

     file-record                X(10)
          record-data           K/L/M/N/O
     INTO-identifier            A(5)0
          Actual results        K/L/M/
```

Expected results K/L/MO

SQ215

A. Compilation:

Lines 0188000 and 018900 in the File Description for CONTIN-FILE elicited the following fatal error message:

"LABEL RECORDS ARE STANDARD" REQUIRED ON MASS-STORAGE FILES.

This error was circumvented by modifying the File Description to specify standard labels.

B. Execution:

LABEL-TEST-1 failed. This test involves OPENing a file with user defined label for INPUT. A declarative procedure is used to check the label.

Test results: 1HDRbbbbbbbbbbbbbbb
Expected results: CAMELbbbbbHORSEbbbbb

SQ216 through 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 ST206

A. Compilation:

No errors.

B. Execution:

No errors.

ST207

A. Compilation:

No errors.

B. Execution:

(1) SORT-TEST-13 failed. A 15-character numeric sort record was returned to an 18-character numeric identifier.

Test results: 000000000091817bbb
Expected results: 000000000000091817

(2) SORT-TEST-14 failed. A 15-character numeric sort record was returned to a 20-character numeric-edited identifier. (PIC $*,*99,999,999,999DB).

Test results: 000968676665646bbbbb
Expected results: $**968,676,665,646bb

(3) SORT-TEST-16 failed. A 15-character alphanumeric sort record was returned to an 18-character alphanumeric edited identifier (PIC AB9X0X(13)).

Test results: 958575655545352bbb
Expected results: 9b580575655545352b

(4) SORT-TEST-17 failed. A 15-character alphanumeric sort record was returned to an 18-character numeric edited identifier (PIC +9(17) BLANK WHEN ZERO).

Test results: 9282726252423222bbb
Expected results: +009282726252423222

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 LEVEL 3

TH301 through TH308

A. Compilation:

No errors.

B. Execution:

No errors.

TH309

A. Compilation:

The construct:

```
SEARCH TBL-A VARYING W AT END GO TO paragraph-name.
```

elicited the following fatal message:

```
ONE OR MORE NAMES IN THIS STATEMENT HAS NOT BEEN DESCRIBED AS DATA.
```

TBL-A had been described as follows:

```
02 TBL-A OCCURS 10 TIMES INDEXED BY A.
   03 ELMT-A PIC 99.
```

W had been described as follows:

```
01 W USAGE INDEX.
```

The error was circumvented by changing the description of W to:

```
01 W PICTURE 99.
```

B. Execution:

No errors.

TH310 through TH311

A. Compilation:

No errors.

B. Execution:

No errors.

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS68-VSR280	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report #CCVS68-VSR280(Assigned by Manager of Testing) HIS OS/2000 COBOL 4.02			5. Report Date 15 February 1978 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 Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Automatic Data Processing Equipment Selection Office Department of the Navy Washington, D. C. 20376			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This Validation Summary Report (VSR) for the Honeywell 2000 Series COBOL Compiler Version 4.02 (OS/2000 Version 4.02) 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 21-1). The compiler was validated at the High level of FIPS PUB 21-1. 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 X3.23-19 , 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

Porgramming 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.	DISTRIBUTION STATEMENT A Approved for public release; Distribution Unlimited	UNCLASSIFIED	
		20. Security Class (This Page) UNCLASSIFIED	22. Price