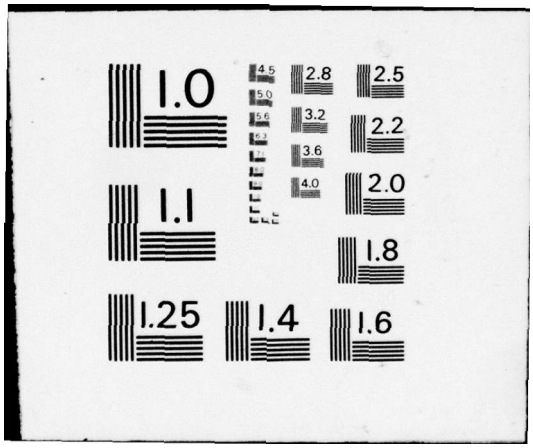
AD-A031 019 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
LOW-INTERMEDIATE PDP-11/70, PDP-11 COBOL COMPILER V 2.0.(U)
OCT 76
UNCLASSIFIED CCVS74-VSR160 NL

1 OF 1
ADA031019

END
DATE FILMED
11-76


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

FEDERAL COBOL COMPILER TESTING SERVICE

VALIDATION SUMMARY REPORT

Department of the Navy
(ADPESO)

Washington, D.C.
20376

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR160	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report #CCVS74-VSR160 Low-Intermediate PDP-11/70, PDP-11 COBOL Compiler V 2.0.			5. Report Date 8 October 1976 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 (ADPESO) 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 CCVS74-VSR160			

16. Abstracts

This Validation Summary Report (VSR) for the Digital Equipment Corporation PDP-11/70 Series, PDP-11 COBOL Compiler Version 2.0 provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 (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 Federal level module; 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

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group 09/02

408 438 AB

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited.	UNCLASSIFIED	35
	20. Security Class (This Page) UNCLASSIFIED	22. Price

INSTRUCTIONS FOR COMPLETING FORM NTIS-35 (Bibliographic Data Sheet based on COSATI Guidelines to Format Standards for Scientific and Technical Reports Prepared by or for the Federal Government, PB-180 600).

1. **Report Number.** Each individually bound report shall carry a unique alphanumeric designation selected by the performing organization or provided by the sponsoring organization. Use uppercase letters and Arabic numerals only. Examples FASEB-NS-73-87 and FAA-RD-73-09.

2. Leave blank.

3. **Recipient's Accession Number.** Reserved for use by each report recipient.

4. **Title and Subtitle.** Title should indicate clearly and briefly the subject coverage of the report, subordinate subtitle to the main title. When a report is prepared in more than one volume, repeat the primary title, add volume number and include subtitle for the specific volume.

5. **Report Date.** Each report shall carry a date indicating at least month and year. Indicate the basis on which it was selected (e.g., date of issue, date of approval, date of preparation, date published).

6. **Performing Organization Code.** Leave blank.

7. **Author(s).** Give name(s) in conventional order (e.g., John R. Doe, or J.Robert Doe). List author's affiliation if it differs from the performing organization.

8. **Performing Organization Report Number.** Insert if performing organization wishes to assign this number.

9. **Performing Organization Name and Mailing Address.** Give name, street, city, state, and zip code. List no more than two levels of an organizational hierarchy. Display the name of the organization exactly as it should appear in Government indexes such as Government Reports Index (GRI).

10. **Project/Task/Work Unit Number.** Use the project, task and work unit numbers under which the report was prepared.

11. **Contract/Grant Number.** Insert contract or grant number under which report was prepared.

12. **Sponsoring Agency Name and Mailing Address.** Include zip code. Cite main sponsors.

13. **Type of Report and Period Covered.** State interim, final, etc., and, if applicable, inclusive dates.

14. **Sponsoring Agency Code.** Leave blank.

15. **Supplementary Notes.** Enter information not included elsewhere but useful, such as: Prepared in cooperation with . . . Translation of . . . Presented at conference of . . . To be published in . . . Supersedes . . . Supplements . . . Cite availability of related parts, volumes, phases, etc. with report number.

16. **Abstract.** Include a brief (200 words or less) factual summary of the most significant information contained in the report. If the report contains a significant bibliography or literature survey, mention it here.

17. **Key Words and Document Analysis. (a). Descriptors.** Select from the Thesaurus of Engineering and Scientific Terms the proper authorized terms that identify the major concept of the research and are sufficiently specific and precise to be used as index entries for cataloging.
(b). Identifiers and Open-Ended Terms. Use identifiers for project names, code names, equipment designators, etc. Use open-ended terms written in descriptor form for those subjects for which no descriptor exists.
(c). COSATI Field/Group. Field and Group assignments are to be taken from the 1964 COSATI Subject Category List. Since the majority of documents are multidisciplinary in nature, the primary Field/Group assignment(s) will be the specific discipline, area of human endeavor, or type of physical object. The application(s) will be cross-referenced with secondary Field/Group assignments that will follow the primary posting(s).

18. **Distribution Statement.** Denote public releasability, for example "Release unlimited", or limitation for reasons other than security. Cite any availability to the public, other than NTIS, with address, order number and price, if known.

19 & 20. **Security Classification.** Do not submit classified reports to the National Technical Information Service.

21. **Number of Pages.** Insert the total number of pages, including introductory pages, but excluding distribution list, if any.

22. **NTIS Price.** Leave blank.

(5)

COBOL COMPILER
VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS74-VSR160

Prepared By:

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

CCVS74-VSR16D

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR16D
2.	Vendor	DIGITAL EQUIPMENT CORPORATION
3.	Mainframe	PDP-11/70
4.	Compiler Identification	PDP-11 COBOL Compiler Version 2.0
5.	Operating System Identification	RSTS/E
6.	Compiler Validation System Version Number	CCVS74 1.0
7.	Federal Information Processing Standard Publication	21-1

*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. Frank Infante
Digital Equipment Corporation
146 Main Street
Maynard, Massachusetts
01754

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 Purpose of the VALIDATION Summary Report

The purpose of the Validation Summary Report (VSR) is to identify individual COBOL language elements whose implementation does not conform to Federal Standard COBOL as adopted from American National Standard Programming Language COBOL, X3.23-1974, by Federal Information Processing Standard 21-1 (FIPS PUB 21-1).

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 many tests and supporting procedures indicating the result 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 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 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 a subsection representing each level of each module defined in American National Standard Programming Language COBOL, X3.23-1974. Each subsection contains a list of all of the language elements which must be implemented in order to claim support of that level/module. The list of language elements will be annotated to include a description of both syntax and semantic errors detected during the validation.

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

c. Section 4 contains information which described the software environment in which the compiler was tested. This includes 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.

d. 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 Abstract Covering Compliance to American National Standard Programming Language COBOL

Definition of an Implementation of American National Standard Programming Language COBOL (excerpts from X3.23-1974, Chapter 1, Section 1.5).

An implementation is defined to meet the requirements of the American National Standard COBOL specification if that implementation includes a fully implemented specified level of each of the functional processing modules and of the nucleus as defined in this standard. It follows from this that, in order to meet the requirements of this standard, an implementation must:

a. Not require the inclusion of substitute or additional language elements in the source program, in order to accomplish any part of the function of any of the standard language elements.

b. Accept all standard language elements contained in a given level of a module which is specified as being included in the implementation, except as specifically exempted as pertaining to specific hardware components for which support is claimed.

These points are of particular pertinence in two areas:

(1) There are throughout the American National Standard COBOL specification certain language elements whose syntax, or semantics are specified to be, in part, implementor-defined. While the implementor specifies the constraints on that portion of each element's syntax or rules that is indicated in this standard to be implementor-defined, such constraints may not include any requirement for the inclusion in the source program of substitute or additional language elements.

(2) When a function is provided outside the source program that accomplishes a function specified by any particular standard COBOL element, then the implementation must not require, except for Environment Division elements, the specification of that external function in place of or in addition to that standard language element:

The following qualifications apply to the American National Standard COBOL specification:

a. There are certain language elements which pertain to specific types of hardware components. In order for an implementation to meet the requirements of this standard, the implementor must specify the minimum hardware configuration required for that implementation and the hardware components that it supports. Further, when support is thus claimed for a specific hardware component, all standard language elements that pertain to that component must be implemented if the module in which they appear is included in the implementation. Language elements that pertain to specific hardware components for which support is not claimed, need not be implemented. However, the absence of such elements from an implementation of American National Standard COBOL must be specified.

b. An implementation of American National Standard COBOL may include the ENTER statement or not, at the option of the implementor.

c. An implementation that includes, in addition to a specified level of each of the functional processing modules and of the Nucleus, elements or functions that either are not defined in the American National Standard COBOL specification or are defined in a given level of a standard module not otherwise included in the implementation, meets the requirements of this standard. 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 that meet the requirements of this standard. The implementor must specify any optional language (language not defined in a specified level but defined elsewhere in the standard) or extensions (language elements or functions not defined in this standard) that are included in the implementation.

d. In general, the American National Standard COBOL specification specifies no upper limit on such things as the number of statements in a program, the number of operands permitted in certain statements, etc. It is recognized that these limits will vary from one implementation of American National Standard COBOL to another and may prevent the proper compilation of some programs that meet the requirements of this standard.

IMPLEMENTOR-DEFINED LANGUAGE SPECIFICATIONS

The language elements in the following lists depend on implementor definitions to complete the specificaton of the syntax or rules for the elements.

The elements whose syntax is partly implementor-defined are:

Element	Implementor-Defined Aspect
SOURCE-COMPUTER paragraph	computer-name
OBJECT-COMPUTER paragraph	computer-name
MEMORY SIZE clause	integer
alphabet-name	implementor-name; whether implementor-names are provided.

SPECIAL-NAMES paragraph	implementor-name
ASSIGN clause	implementor-name
VALUE OF clause	implementor-name; whether implementor-names are provided.
RERUN clause	implementor-name and the form; the implementor provides at least one of seven specified forms.
CALL and CANCEL statements	relationship between operand and the referenced program.
COPY statement	relationship between library-name text-name, and the library.
ENTER statement	language-name
Margin R	The location.
Area B	The number of character positions.
Qualification	The number of qualifiers; at least five must be supported.

The elements whose effect is partly implementor-defined are:

Element	Implementor-Defined Aspect
alphabet-name	The correspondence between native and foreign character sets.
implementor-name switches	Whether setting can change during execution.
USAGE IS COMPUTATIONAL clause	Representation and whether automatic alignment occurs.
USAGE IS INDEX clause	Representation and whether automatic alignment occurs.
SYNCHRONIZED clause	Whether implicit FILLER positions are generated; their effect on the size of group items and redefining items.
ACCEPT statement	Maximum size of one transfer of data in Level 1 Nucleus.
DISPLAY statement	Maximum size of one transfer of data

	in Level 1 Nucleus.
Numeric test	Representation of valid sign in the absence of the SIGN IS SEPARATE clause.
Comparison of nonnumeric items	Collating sequence, where NATIVE or implementor-name collating sequence is implicitly or explicitly specified.
Arithmetic expressions	Number of places carried for intermediate results.

Elements That Pertain to Specific Hardware Components

The standard language elements in the list that follows pertain to specific types of hardware components. These language elements must be implemented in an implementation of American National Standard COBOL when support is claimed, by the implementor, for the specific types of hardware components to which they pertain, and the module in which they are defined is included in that implementation.

Element	Hardware Component
CODE-SET clause	Device capable of supporting the specified code.
MULTIPLE FILE TAPE clause	Reel
CLOSE...REEL/UNIT statement	Reel or mass storage
CLOSE...NO REWIND statement	Reel or mass storage
OPEN...REVERSED statement	Reel with the capability of making records available in the reversed order; mass-storage with the capability of making records available in the reversed order.
OPEN...NO REWIND statement	Reel or mass storage
OPEN...I-O statement (Sequential I-O only)	Mass storage
OPEN EXTEND statement	Reel or mass storage
REWRITE statement (Sequential I-O only)	Mass storage
SEND...BEFORE/AFTER	Devices capable of vertical posi-

ADVANCING statement	tioning; devices capable of action based on mnemonic-names.
USE...I-O (Sequential I-O only)	Mass storage
WRITE...BEFORE/AFTER ADVANCING	Devices capable of vertical positioning; devices capable of action based on mnemonic-name.

1.5 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 American National Standard Programming Language COBOL, X3.23-1974, 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 1.5.1) 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 is not contained in the Federal Standard COBOL.

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.

1.5.1 Federal Standard COBOL Levels

a. Federal Standard COBOL specifications are the language specifications contained in American National Standard Programming Language COBOL, X3.23-1974. For purposes of the Federal Standard, the modules defined in X3.23-1974 are combined into four levels. 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.

b. The four levels of Federal Standard COBOL are identified as: Low, Low-Intermediate, High-Intermediate, and High. Each Federal Standard COBOL level is composed of either the high or low levels of the nucleus and ten of the eleven Functional Processing Modules (FPMs) defined in X3.23-1974. The four Federal Standard COBOL levels are reflected in the following table. The numbers in the table refer to the level within the FPM or nucleus as designated in X3.23-1974, and a dash in the table denotes that the corresponding FPM is omitted.

	Low Level	Low Inter-mediate Level	High Inter-mediate Level	High Level
NUCLEUS	1	1	2	2
FPMs				
TABLE HANDLING	1	1	2	2
SEQUENTIAL I-O	1	1	2	2
RELATIVE I-O	-	1	2	2
INDEXED I-O	-	-	-	2
SORT-MERGE	-	-	1	2
REPORT WRITER	-	-	-	-
SEGMENTATION	-	1	1	2
LIBRARY	-	1	1	2
DEBUG	-	1	2	2
INTER-PROGRAM COMMUNICATION	-	1	2	2
COMMUNICATION	-	-	2	2

1.5.2 Conformance to Federal Standard COBOL

A compiler implemented in conformance to Federal Standard COBOL must meet at least the following requirements.

a. The implementation must include all of the language elements of at least one of the levels of Federal Standard COBOL.

b. The implementation must meet all of the requirements defined in American National Standard COBOL, X3.23-1974, Section I, paragraph 1.5, Definition of An Implementation of American National Standard COBOL which is provided in Section 1.4 of this VSR.

c. The implementation must provide a facility for the user to optionally specify a level of Federal Standard COBOL for monitoring his source program at compile time. The monitoring will be an analysis of the syntax used in a source program against the syntax included in the specified level of Federal Standard COBOL. Any syntax used in the source program that does not conform to that allowed by the user selected level of Federal Standard COBOL will be diagnosed. The syntax diagnosed as not conforming to the specified level will be identified to the user through a diagnostic message on the source program listing. The diagnostic message will contain, at least: (1) The identification of the source program line number in which the nonconforming syntax occurs, (2) the identification of the level of Federal Standard COBOL that supports

the syntax or that the syntax is nonstandard COBOL.

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 [illegible] 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 21-1 defines the Federal COBOL Language Standard. This publication is available from the Office of ADP Standards Management, National Bureau of Standards, Washington, D.C., 20234.

The detailed COBOL language specifications are given in the publication "American National Standard Programming Language COBOL X3.23-1974", available from the 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 containing a copy of the CCVS programs are available from the National Technical Information Service, Springfield, Virginia, 22151. (Ordering information can be obtained from the Federal COBOL Compiler Testing Service.)

1.8. Requests for Interpretation

Questions regarding this VSR, or the CCVS in general 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.

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 (CCVS). The version of the CCVS used during this validation is shown inside the front cover of the VSR.

Section 2 is made up of a variable number of subsections. The number of subsections is dependent on the Level of Federal COBOL being validated. There will be a subsection for each level of each module which is validated. If the high level of a module is validated then there will be two subsections for that module; one for the low level and one for the high level.

A validation of the low level of Federal COBOL would result in three subsections being present. One for Nucleus level 1, one for Sequential I-O level 1 and one for level 1 Table Handling.

Each error or deviation noted in this section makes reference to a program or functional COBOL module contained in Appendix A (Validation Summary Working Document). This reference provides the documented results of an occurrence of errors/deviataions detected during the running of the CCVS using the compiler within the environment identified within this document. The Validation Summary Working Document is presented in sequence by functional module, functional module level and program number as defined below.

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 American National Standard Programming Language COBOL tested within the program.

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

CM - Communications
DB - Debugging
IC - Inter-Program Communications
IX - Indexed I-O
LB - Library
NC - Nucleus
RL - Relative I-O
RP - Report Writer
SG - Segmentation
SQ - Sequential
ST - Sort
TH - Table Handling

The third character of the audit routine name is either a 1 or 2, 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.

Description of Section 2.

Each error/deviation is noted by number in the left hand margin opposite the language element in question. This number is used in section 3 to categorize errors by Federal level (See 1.5.1). Inserted directly below the language element is a brief description of the error. To the right of the langauge element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference at the end of the description of the error is to appendix A which contains the detailed information collected during the validation. The reference is made up of the routine name followed by an A or B (A for compile time or syntax errors and B for execution time or semantic errors) and a number which makes the error unique in the Appendix A.

Example:

```
  2.1 Nucleus Level 1

        .
        .
        .
        Operational symbols: S V P                    II-21
2.1.9   ------------------------------------------------------
        * The scaling character 'P' is not permitted in a
        * PICTURE character-string.
        *                                        (NC101.A.2)
        ------------------------------------------------------
        .
        .
        .
  2.2  Sequential I-O Level 1
```

2.1.9 represents the ninth error for Nucleus Level 1

II-21 represents the page in X3.23-1974 where the language element is defined

* Boxes the description of the error/deviation

NC101.A.2 represents:

Program name - NC101
Syntax error - A
second error - 2

2.1 NUCLEUS LEVEL 1

2.1.1 ----------

* The compiler does not recognize the WORKING-STORAGE
* SECTION header continued across source lines. (Refer
* to Reference Format 5.8.2.2 Continuation of Lines
* on page I-106.)
* (NC113.A.1)

---

implementor-name IS mnemonic-name series
ON STATUS
OFF STATUS
alphabet-name clause
CURRENCY SIGN clause
DECIMAL-POINT clause

2.1.2

```
*     The compiler does not accept the syntax for
*     mnemonic-name for a switch in the SPECIAL-
*     NAMES paragraph.  (Refer to Nucleus 3.1.3.4
*     on page II-9).
*                          (NC103.A)
```


2.1.3

```
*     The compiler does not perform certain storage
*     allocations when the unsigned data items sub-
*     ordinate to a group item containing SIGN IS
*     SEPARATE clause in the data description.
*     (Refer to Nucleus 4.10 on page II-31.)
*                          (NC120.B.1)
```

2.1.4

---

* The compiler limits the intermediate results
* of any arithmetic operation to 18 digits, which
* may cause errors when the composite of operands
* in the ADD statement has 18 digits. The SIZE ERROR
* code is also being performed even though no SIZE
* ERROR should occur in the ADD operation. (Refer to
* Nucleus 5.5 on page II-55.)
* (NC106.B.1)

2.1.5

---

* The compiler has a limit of sixteen (16) sending
* operands in a DISPLAY statement. (Refer to Nucleus
* 5.8 on page II-59.)
* (NC109.A)

BY identifier/literal
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase

2.1.6 ----------

* The compiler has a problem with DIVIDE when the
* data items have a SIGN SEPARATE clause as part of
* the data description. (Refer to Nucleus 5.9 on
* page II-51.)
* (NC117.3.1 and NC117.3.2)

2.1.7 ----------

* The compiler has a problem with MULTIPLY when the
* data items have a SIGN SEPARATE clause as part of
* the data description. (Refer to Nucleus 5.16 on
* page II-77.)
* (NC120.3.2)

2.1.8 ----------

* The compiler has a limit of 16 active levels in
* a sequence of nested PERFORM statements. (Refer to
* Nucleus 5.17 on page II-78.)

```
*                                   (NC102.3)
-----------------------------------------------------------------
   The STOP statement . . . . . . . . . . . . . . . .  II-85
     literal
     RUN
   The SUBTRACT statement . . . . . . . . . . . . . .  II-89
     identifier/literal series
     FROM identifier
     GIVING identifier
     ROUNDED phrase
     SIZE ERROR phrase
2.1.9 -----------------------------------------------------------
*     The compiler has a problem with SUBTRACT when the
*     SIZE ERROR option is used.  The SIZE ERROR code is
*     executed even though no SIZE ERROR should have occurred
*     during the SUBTRACT operation.  (Refer to Nucleus 5.20
*     on page II-89.)
*                                (NC106.3.2)
-----------------------------------------------------------------
```

2.2 TABLE HANDLING LEVEL 1

Language Concepts
- User-defined words. I-76
  - index-name
- Subscripting - 3 levels I-89
- Indexing - 3 levels I-89

Data Division

2.2.1

```
*     The compiler has a limit of 4095 characters for
*     any COBOL data item (group item, elementary item,
*     or table).  (Refer to Table Handling  2. on page III-2.)
*                              (TH107.A  and  TH103.A)
```

- The OCCURS clause III-2
  - integer TIMES
  - INDEXED BY index-name series
- The USAGE IS INDEX clause III-5

Procedure Division
- Relation conditions III-6
  - Comparisons involving index-names and/or index data items
- Overlapping operands. III-6
- The SET statement III-11
  - index-name/identifier series
  - index-name

2.2.2

```
*     The compiler cannot handle leading zeroes in a
*     numeric literal in the SET statement.  (Refer to
*     Table Handling  3.4 on page III-11.)
*                                              (TH104.A)
```

  - UP BY identifier/integer
  - DOWN BY identifier/integer
  - index-name series

2.3 SEQUENTIAL I-O, LEVEL 1

The OPEN statement (only a single file-name may appear in an OPEN statement). IV-24
INPUT
OUTPUT
I-O
The READ statement IV-28
INTO identifier
AT END phrase
The REWRITE statement. IV-31
FROM identifier
The USE statement. IV-32
EXCEPTION/ERROR PROCEDURE
ON file-name
ON INPUT
ON OUTPUT
ON I-O
The WRITE statement IV-34
FROM identifier
BEFORE/AFTER integer LINES

2.3.2

```
*     The compiler has a series of problems with WRITE
*     BEFORE/AFTER ADVANCING integer LINES.  Numeric
*     literals cannot have leading zeroes.  The figurative
*     constant ZERO and the numeric literal 0 cannot be
*     used.  WRITE BEFORE ADVANCING 1 LINES advances
*     before the WRITE is performed.  (Refer to Sequential
*     I-O 4.6 on page IV-34.)
*          -- (SQ101.A.1, SQ101.A.2, SQ101.B.1, and SQ101.B.2)
```

BEFORE/AFTER PAGE

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

3.1.1 Nucleus Level 1

2.1.1 Continuation of WORKING-STORAGE SECTION header.
2.1.2 Syntax for switch mnemonic-name in SPECIAL-NAMES paragraph.
2.1.3 Storage allocation for SIGN IS SEPARATE clause.
2.1.4 ADD SIZE ERROR problem, also a compiler limit of 18 digits for any arithmetic intermediate result.
2.1.5 Limit of 16 sending operands in a DISPLAY statement.
2.1.6 DIVIDE problem when SIGN IS SEPARATE.
2.1.7 MULTIPLY problem when SIGN IS SEPARATE.
2.1.8 Limit of 16 active levels in a nested PERFORM sequence.
2.1.9 SUBTRACT problem when SIZE ERROR option is used.

3.1.2 Table Handling Level 1

2.2.1 Limit of 4095 characters for any COBOL data item.
2.2.2 Leading zeroes in numeric literals in the SET statement.

3.1.3 Sequential I-O Level 1

2.3.1 The RERUN statement is not implemented.
2.3.2 Problems with WRITE BEFORE/AFTER ADVANCING integer LINES.

3.2 Low Intermediate Level

Validation at this level not attempted.

3.3 High-Intermediate Level

Validation at this level not attempted.

3.4 High Level

Validation at this level not attempted.

SECTION 4. 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:

The largest possible task size was requested for all executions.
SWITCH 1 was set ON.

2. Environment Division implementor-names.

Printer destined files

"SY:name.extension".

Tape files

"MTn:name.extension".

Sequential Mass-storage files

"SY:name.extension".

Random Access files

N/A

Sort files (SD)

N/A

Switch names

SWITCH 1
SWITCH 2

Source Computer names

PDP-11.

Object Computer names

PDP-11

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 cal 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.

N/A

4. Compiler.

PDP-11 COBOL Compiler Version 2.0

5. Operating system.

RSTS/E

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 21-1 (X3.23-1974).

CCVS74-VSR16D

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION: PDP-11 COBOL Compiler Version 2.0

COMPUTER SYSTEM: PDP-11/70

OPERATING SYSTEM: RSTS/E

NUCLEUS MODULE LEVEL 1

NC101

A. Compilation

No errors.

B. Execution

No errors.

NC102

A. Compilation

No errors.

B. Execution

The execution of NC102 stopped in PFM-TEST-14 with the message:

F000012 PROC:12/220 TOO MANY ACTIVE PERFORMS.

PERFORM-TEST-14 is a test of the nested PERFORM statement with 20 levels. The compiler can handle a maximum of 16 levels in a nested PERFORM. PFM-TEST-14 was deleted.

NC103

A. Compilation

The compiler does not accept the syntax for mnemonic-name for a switch in the *SPECIAL-NAMES paragraph*.

SPECIAL-NAMES. implementor-name IS mnemonic-name, ON STATUS IS condition-name-1, OFF STATUS IS condition-name-2.

After the mnemonic-name reference was deleted, the switch and condition-

name references functioned according to the Standard.

B. Execution

No errors.

NC104 thru NC105

A. Compilation

No errors.

B. Execution

No errors.

NC106

A. Compilation

A warning message that POSSIBLE HIGH ORDER RECEIVING FIELD TRUNCATION might occur was given by the compiler for ADD-TEST-41.

B. Execution

1. ADD-TEST-39 and ADD-TEST-41 failed. These are ADD data-name series tests with operands like PIC S9(18) and SV9(18). The composite operand does not exceed 18 digits. In ADD-TEST-39 the SIZE ERROR did not occur; however, the result of the ADD was -999999999999999999 instead of the expected value 1. In ADD-TEST-41 the SIZE ERROR did not occur yet the result was -.9 instead of the expected value of .1 .

2. SUB-TESTS-43, 44, 45, and 46 failed. These are tests of SUBTRACT data-name series GIVING result ON SIZE ERROR. Tests 44 and 46 show that a SIZE ERROR was produced during the subtract even though no SIZE ERROR was expected. Like ADD-TESTS-39 and 41 above, the operands include those with PICTUREs like S9(18) and SV9(18). In SUB-TEST-43 the result was -999999999999999999 instead of the expected value of 0. In SUB-TEST-45 the result was -.9 instead of the value .0

The compiler has a limit of 18 digits for any intermediate result.

NC107

A. Compilation

No errors.

B. Execution

CCVS74-VSR160

No errors.

NC108

A. Compilation

The mnemonic switch name had to be deleted. See NC103.

B. Execution

No errors.

NC109

A. Compilation

DISP-11STS-12 and 13 had to be deleted. These are DISPLAY tests with 21 operands. The compiler has a limit on the number of operands that are allowed in a DISPLAY statement.

B. Execution

No errors.

NC110 thru NC112

A. Compilation

No errors.

B. Execution

No errors.

NC113

A. Compilation

Lines 003000 thru 003200 had to be modified. These lines contain the WORKING-STORAGE SECTION header which is split across lines using the continuation character hyphen in column 7.

```
003000     WORKING-
003100-     STORAGE
003200       SECTION.
```

B. Execution

No errors.

NC114 thru NC116

CCVS74-VSR16D

A. Compilation

No errors.

B. Execution

No errors.

VC117

A. Compilation

No errors.

B. Execution

1. DIV-TEST-7 is as follows:

```
DIVIDE D.D97 INTO DIV7 ROUNDED ON SIZE ERROR ...
```

The SIZE ERROR occured as expected; however, the resultant identifier was changed from its initial value of 9.6 to a final value of 99. DIV-TEST-8 failed because is is a check on whether the resultant identifier was changed after a SIZE ERROR ocurred. The resultant identifier was contained in a group item which was described as 'TRAILING SEPARATE' for the signs within the data items subordinate to the group.

2. DIV-TEST-19 failed. The test is as follows:

```
DIVIDE A99-DS-D2VDD INTO WRK-DS-LS-18VDD.

77 A99-DS-D2DVDD PIC S99 VALUE 99.
77 WRK-DS-LS-18VDD PIC S9(18) SIGN IS LEADING SEPARATE CHARACTER
   VALUE 99.
```

A result of 0 was produced instead of the expected value of 1.

NC118

A. Compilation

No errors.

B. Execution

ADD-TESTS-39 and 41 failed. These are the same tests as in NC106.

NC119

A. Compilation

CCVS74-VSR160

No errors.

B. Execution

SUB-TESTS-44 and 46 failed. The SIZE ERROR code was executed even though no SIZE ERROR should have occurred for the SUBTRACT operation.

NC120

A. Compilation

No errors.

B. Execution

1. MPY-TEST-2 failed. The group sign is LEADING SEPARATE. This same test passed in NC101 with the sign clause not specified for the data items.

2. MPY-TEST-5 failed. The compiler produced the warning messages:

```
I 00433 0371  POSSIBLE HIGH ORDER RECEIVING FIELD TRUNCATION
I 00433 0372  POSSIBLE LOW ORDER RECEIVING FIELD TRUNCATION.
```

The SIZE ERROR should have been executed during the multiplication in MPY-TEST-5. MPY-TEST-6 also failed. The receiving field should not have been changed from the initial value of 4. The resultant field was changed to 10.

The compiler stored the data 10 in the data item MULT5 which is PIC 9 and also stored the data 100 in MULT6 which has a PIC 99. The 01 level group data name which includes MULT5 and MULT6 has LEADING SEPARATE for the sign; however neither MULT5 nor MULT6 have the sign S as part of the PICTURE character-string.

This problem caused MPY-TESTS-9, 10, 11, and 12 to fail by not performing the SIZE ERROR and changing the resultant field from the initial values.

TABLE HANDLING MODULE LEVEL 1

TH101 thru TH103

A. Compilation

No errors.

B. Execution

No errors.

CCVS74-VSR160

TH104

A. Compilation

The compiler can not handle leading zeroes on a numeric literal; e.g.

```
034700     SET IDX-1 TO 0000000000000000004.
```

All occurrences had the leading zeroes removed.

B. Execution

No errors.

TH105 thru TH106

A. Compilation

No errors.

B. Execution

No errors.

TH107

A. Compilation

The compiler issued the following fatal error message because the number of characters in the 01 group level exceeded 4095:

```
F 00115 0576  DATAITEM LENGTH EXCEEDS 4095 CHARACTERS.
```

An option was set to ignore the fatal error and no other changes had to be made to successfully run the routine.

B. Execution

No errors.

TH108

A. Compilation

The fatal message that a data item exceeds 4095 characters was produced by the compiler (see TH107).

B. Execution

No errors.

CCVS74-VSR160

TH109 thru TH111

A. Compilation

No errors.

B. Execution

No errors.

SEQUENTIAL I-O MODULE LEVEL 1

SQ101

A. Compilation

1. The compiler issues a warning message and does not accept 17 leading zeroes on a numeric literal such as 00000000000000001.

W 00490 0043 INTEGER VALUE TOO BIG. LARGEST VALUE USED.
W 00439 0340 .ADVANCING. INTEGER TOO BIG. TRUNCATED TO 63.

All such leading zeroes occurred in the WRITE ADVANCING numeric literal LINES statements. The leading zeroes were omitted.

2. The compiler does not accept the figurative constant ZERO in the WRITE ADVANCING statement.

3. The compiler also does not handle a numeric literal with the value of zero in the WRITE ADVANCING 0 LINES tests for overprinting. All such occurrences were changed to ADVANCING 1 LINES. This caused overprint tests to fail.

B. Execution

1. The following tests failed because the compiler could not WRITE BEFORE (or AFTER) ADVANCING 0 LINES.

WRT-TEST-7/8
WRT-TEST-15/16

2. The following tests failed because the compiler does not correctly handle a WRITE BEFORE ADVANCING integer LINES statement.

WRT-TEST-01 thru 06, WRT-TEST-09 thru 12, WRT-TEST-15
WRT-TEST-17 thru 13, WRT-TEST-21 thru 22, WRT-TEST-27 thru 28
WRT-TEST-31 thru 32.

SQ102 thru SQ112

CCVS74-VSR160

A. Compilation

No errors.

B. Execution

No errors.

SQ113

A. Compilation

1. Syntax for switch mnemonic-name (see NC103).

2. The RERUN statement is not implemented for this compiler. The entire I-O CONTROL paragraph with the RERUN statement was deleted.

B. Execution

No errors.

SQ114 thru SQ117

A. Compilation

No errors.

B. Execution

No errors.