AD-A068 919 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT. DATA GENERAL C350 - A--ETC(U)
APR 79

UNCLASSIFIED FCCTS-CCVS74-VSR360 NL

| OF |
AD
A068919

END
DATE
FILMED
7-79
DDC

LEVEL II

(1)

(11) 30 Apr 79

(6)

COBOL COMPILER
VALIDATION SUMMARY REPORT

DATA GENERAL C350 - AOS COBOL

VALIDATION NUMBER CCVS74-VSR360

(14) FCCTS-

(12) 42p.

Prepared By:

FEDERAL COMPILER TESTING CENTER
AUTOMATED DATA AND TELECOMMUNICATIONS SERVICE
GENERAL SERVICES ADMINISTRATION
WASHINGTON, D.C. 20406

408 438

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. CCVS74-VSR360	2.	3. Recipient's Accession No.

4. Title and Subtitle
Validation Summary Report # CCVS74-VSR360 (Assigned by Manager Data General C350 AOS COBOL of Testing)

5. Report Date
April 30, 1979

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 (ADPSO)
Washington, D.C. 20376

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C)
(G)

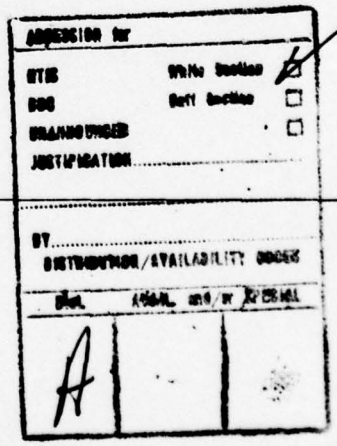
12. Sponsoring Organization Name and Address
Automatic Data Processing Selection Office
Department of the Navy
Washington, D.C. 20376

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)
This Validation Summary Report (VSR) for the Data General AOS COBOL Compiler Version 2.00 (AOS Version 2.03) provides a consolidated summary of the results obtained from the validation of the subject compiler against the 1974 COBOL Standard (X3.23-1974/FIPS PUB 21-1). The compiler was validated at the Low 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 be level/module within X3.23-1974, 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. Document Analysis a. Descriptors

Programming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

b. Identifiers/Open-Ended Terms

CCVS
CVS

c. COSATI Field/Group 09/02

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited UNCLASSIFIED	UNCLAS	42
	20. Security Class (This Page) UNCLAS.	22. Price PC E02 MF E02

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

TABLE OF CONTENTS

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR360
2.	Vendor	Data General Corporation
3.	Mainframe	Eclipse C350
4.	Compiler Identification	Data General Corporation AOS COBOL Release 2.00
5.	Operating System Identification	AOS Revision 2.03
6.	Other Hardware/Software Environments*	Data General M600 system
7.	Compiler Validation System Version Number	CCVS74 3.0
8.	Federal Information Processing Standard Publication	21-1
9.	FIPS Validation Level	Low
10.	Report Date	15 April 1979

* List of the additional hardware/software environments on which the COBOL Compiler Validation System was not run but for which the vendor certifies that the identified compiler will produce the same results as noted in this report.

PLEASE NOTE. The Federal Compiler Testing Center 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. Robert M. Kiburz
Data General Corporation
Route 9
Westboro, Massachusetts 01581

Phone: 617-366-8911

1. 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 American National Standard Programming Language COBOL, X3.23-1974, and to Federal Standard COBOL as adopted from the American National Standard 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 Compiler Testing Center 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 21-1 defines four Federal levels of the COBOL Standard. Section 3.1 of the VSR lists the discrepancies described in Section 2

by the Federal level in which the problem occurs. Section 3.2 lists discrepancies with regard to the Flagging requirement defined in Federal Standard COBOL (the FIPS FLAGGER). Section 3.3 lists discrepancies for the Report Writer Module, which is not a part of Federal Standard COBOL but is contained in X3.23-1974.

c. Section 4 contains information which describes 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. Section 5 contains the results of the ASCII validation. The purpose of these tests is to ascertain whether magnetic tapes written in ASCII code and with ANSI standard labels, and card decks with ASCII code, can be transported between the system being validated and a foreign computer system.

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. Not all VSR's will contain this appendix.

1.4 Abstract Covering Compliance to ANS 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 not claimed). See "Elements that Pertain to Specific Hardware Components" below.

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 effect, is specified to be, in part, implementor-defined. While the implementor specifies the con-

straints on that portion of each element's syntax or rules that is indicated cated 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 textname, 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 ADVANCING statement	Devices capable of vertical positioning; 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 COBOL 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 Federal COBOL Standard states that the Report Writer Module is not mandatory in any Federal level, but that the specifications contained in X3.23-1974 should be used to the extent practical, consistent with requirements.

The Federal COBOL 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	Low Inter-mediate	High Inter-mediate	High
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 Compiler Testing Center 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 21-1 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 Compiler Testing Center.)

1.8 Requests for Interpretation

Questions regarding this VSR or the CCVS in general should be forwarded to the FCTC. If any problem cannot be adequately resolved through the FCTC, 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 Federal Compiler Testing Center, Automated Data and Telecommunications Service (CFT), General Services Administration, Washington, DC 20406.

1.9 Modules and Language Elements Excluded from Testing

During an official validation, certain CCVS tests may not be used, and certain facilities provided by the subject compiler may not be tested.

1.9.1 Federal Standard COBOL Approved Interpretations

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.

1.9.2 Report Writer Module

FIPS PUB 21-1 excludes the Report Writer Module from the Federal COBOL Standard. However, the Report Writer Module is still tested during a validation if support for that module is claimed by the compiler vendor.

1.9.3 Communication Module

Although it is part of Federal Standard COBOL as defined by FIPS PUB 21-1, the Communication Module is not currently tested in the course of an official validation for two specific reasons. First, a large volume of requests for interpretation on this module have been submitted to the cognizant ANSI committee (X3J4) for resolution. Secondly, facilities for testing were insufficient to determine the validity of the Communication Module test programs during the development of CCVS74.

1.9.4 Vendor Omissions or Extensions

Language elements are not tested which have been legitimately omitted from the implementation by the implementor (refer to 1.4). Additionally, no implementor extensions to the standard COBOL language are tested in any way.

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 Compiler Testing Center
Automated Data and Telecommunications Service (CFT)
General Services Administration
Washington, DC 20406
(703) 557-7806

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

2. 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 Standard COBOL would result in three subsections being present. One for Nucleus level 1, one for Sequential I-O level 1, and one for Table Handling level 1.

Each error or deviation noted in this section makes reference to a program or functional COBOL module. If additional information is required or is available it will be contained in the optional Appendix A. If no further information is necessary then Appendix A will not be present.

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.

For the audit routines which are not flagging routines, a five character name is used which has the general format XXNMM. The first two characters are alphabetic and identify the functional module tested by the program. The permissable values are:

NC - Nucleus
TH - Table Handling
SQ - Sequential I-O
RL - Relative I-O
IX - Indexed I-O
ST - Sort-Merge
RW - Report Writer
SG - Segmentation
LB - Library
DB - Debug
IC - Inter-Program Communication
CM - Communication

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.

For the audit routines which are used to validate the flagging requirement of Federal Standard COBOL (1.5.2.c), a five character program name is used which has the format XXYZN. The first two characters XX identify the functional COBOL module being tested by the program and the permissible values are the same as noted above. The third character Y is the character 4 and identifies the program as one of a series which is used for testing compiler flagging. The fourth character Z is the character 2, 3 or 4 and identifies the highest Federal level of the COBOL language needed to support all language elements used in the program. The characters 2, 3 and 4 represent the Low-Intermediate, High-Intermediate and High Levels respectively of Federal Standard COBOL. The fifth character N is a number 1 through 9 which gives a unique name to each program containing features within the same functional processing module and Federal COBOL level.

As an example, the program name LB421 is the first program of a set and the COBOL elements used in the program are available at the Low-Intermediate and higher levels of Federal Standard COBOL. Also, the program is used in testing compiler flagging for the Library 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 language element is a page reference to X3.23-1974, American National Standard Programming Language COBOL. The reference enclosed in parenthesis at the end of the description is to the appropriate program contained in the Compiler Validation System which gives an example of the of the error/deviation detected. If the suffix to the program name is zeros (e.g., NC000) then the error was detected in the supporting documentation provided with the compiler.

If the optional Appendix A is present, then the program name associated with each error may be suffixed with additional information. For example an A or B following the name represents a syntax or semantic error respectively. A number following the A or B represent the error number which can be found in appendix A.

Example:

2.1 Nucleus Level 1

.
.
.

Operational symbols: S V P II-21

2.1.9 ----------

* The Picture character 'P' is not supported
* in this implementaion.
* (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	(If Appendix A is present)
second error	-	2	(If Appendix A is present)

2.1 Nucleus Level 1

2.1.1 ERROR: Accept Statement

```
**************************************************************************
*   The ACCEPT statement failed to place the proper data in a field      *
*   with a numeric PICTURE when the character-string being ACCEPTed     *
*   was longer than the receiving field and truncation was required     *
*                                                          (NC109.B.1)  *
**************************************************************************
```

2.2 Table Handling Level 1

2.3 Sequential I-O Level 1

2.3.1 ERROR: File Description Entry

****
* A non-standard clause had to be added to the file description *
* of a program reading a card deck as part of the ASCII *
* validation. The record length had to be increased by one *
* in order to accept the card images as 80 character records. *
* (SQ119.B) *
****

3. SECTION 3. COMPILER STATUS

3.1 Federal Standard COBOL

Section 1.5 explains the four levels of Federal Standard COBOL and their relation to American National 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. When the subject compiler fails to support an entire functional module, that error indication appears in upper case to highlight its seriousness in comparison with other errors which may involve only a particular language element or facility.

3.1.1 Low Level

2.1.1 ACCEPT statement involving truncation.
2.3.1 FD - modifications required to read ASCII card deck.

3.1.2 Low-Intermediate Level

Validation was not performed at this level.

3.1.3 High-Intermediate Level

Validation was not performed at this level.

3.1.4 High Level

Validation was not performed at this level.

3.2 Federal Standard COBOL Flagging

A requirement of Federal Standard COBOL is that the COBOL implementation include a facility for flagging COBOL elements which do not conform to a specified level of Federal Standard COBOL. This section lists the flagging discrepancies described in Section 2 by Federal COBOL level.

This compiler does not include the required flagging facility.

3.3 American National Standard COBOL

Full American National Standard COBOL consists of the entire set of language elements defined in the ANSI COBOL standard (refer to 1.7). It is also the equivalent of high level Federal Standard COBOL plus the Report Writer module. Therefore, this section lists only those discrepancies found while validating the Report Writer Module.

This implementation does not include a Report Writer module.

4. 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 21-1 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.)

4.1 Compiler options used

Options specified:

```
L  (list source file)
C  (source is in card format)
```

Option defaulted:

```
Omit address map (no A option)
Omit debugger (no D)
Do not compile extensions (no E)
Omit machine code listing (no G)
Omit data and procedure storage maps (no M)
Generate object file (no P)
Give complete compilation (no Q)
Omit statistics (no S)
Omit virtual segmentation (no V)
List warning messages (no W)
Omit cross reference listing (no X)
```

4.2 Environment Division implementor-names

Printer destined files

```
PRINTER "@LIST"
```

Tape Files

```
"linkname:file-id"
```

Sequential I-O Mass-storage files

"disk-file-id"

Switch names

SWITCH "A"
SWITCH "B"

Source Computer name

ECLIPSE

Object Computer name

ECLIPSE

4.3 Optimization

The compiler may or may not have optimization features. If optimization is available by option, 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.

This compiler has no optimization option.

4.4 Compiler

AOS COBOL Revision 2.00

4.5 Operating System

AOS Revision 2.03

4.6 Other Software Support

Other system software released with this revision of AOS COBOL and the AOS operating system includes AOS INFOS Revision 2.02 and AOS SORT Revision 2.02.

5. SECTION 5. ASCII VALIDATION

5.1 Purpose of ASCII Validation

The ASCII Validation is performed by running a sequence of three CCVS74 programs (SQ118, SQ119, SQ120) using special procedures. The purpose of this special run is to validate that the compiler/operating system being tested is capable of processing ASCII code represented on magnetic tape and punched cards that were produced (in accordance with the appropriate American National Standard) by another system. There is also a magnetic tape and a card file created during the validation which will be taken to another system for further processing. The purpose is to determine whether the compiler/operating system being tested can also produce ASCII representation on magnetic tape and punched cards which can be processed by a another computer system.

5.2 Applicable ANSI Standards

The ASCII Validation is based on several American National Standards and presumes their support by the compiler/operating system being validated. These are:

1. American National Standard Programming Language COBOL X3.23-1974.

   - The CODE-SET clause is used to read and write the ASCII files.

   - The PROGRAM COLLATING SEQUENCE clause is used to process the data in ASCII mode as well as native mode.

   - The SIGN...SEPARATE clause is used for signed data and all data is in the DISPLAY (character) mode.

2. American National Standard Code for Information Interchange (ASCII) X3.4-1977. (Note that this describes the code, not the labeling and tape recording formats.)

3. American National Standard Hollerith Punched Card Code, X3.26-1970.

4. American National Standard Magnetic Tape Labels for Information Interchange, X3.27-1977.

5. American National Standard Recorded Magnetic Tape

for Information Interchange (800 CPI, NZRI), X3.22-1973.

6. American National Standard Recorded Magnetic Tape for Information Interchange (1600 CPI, PE), X3.39-1973.

7. American National Standard Recorded Magnetic Tape for Information Interchange (6250 CPI, Group Coded Recording), X3.54-1976.

The language of the 1974 COBOL Standard provides the capability to accept, process, and produce ASCII code. The ASCII Standard describes the code insofar as the bit arrangement and configuration, but does not address recording techniques, record formats or any labeling scheme. The 800 CPI, NZRI magnetic tape recording standard was used to establish the recording density and techniques. (1600 CPI, PE based on X3.39-1973 "Recorded Magnetic Tape for Information Interchange" could be used under special arrangements.) The tape labeling scheme used in these tests is based on X3.27-1969 but is also compatible with the revision to that tape label standard. Only the VOL1, HDR1, and EOF1 labels are used. The records are fixed length and unblocked.

5.3 ASCII Validation Process

During the validation, the Validation Manager for the Federal COBOL Compiler Testing Service uses the ASCII-encoded magnetic tape and card files in addition to the normal tape files associated with a validation. For the ASCII portion of the validation the following steps are performed:

1. The tape file and card deck (produced on another computer system) are used as input to several programs designed to validate whether the system being validated can accept and process the data as defined by the respective standards. Any changes made during this validation to the source programs reading the data are noted below in 5.4.1.

2. A tape file and card file are produced during the validation which should prove to be identical to the files described in 1 above. These two files are then processed on a different computer system to determine the degree to which the system being validated supports the ASCII standard. Any changes made during this validation to the source program producing the data are noted below in 5.4.2.

5.4 Results for This Validation

1. In order for the program to be able to read the supplied card deck, it was necessary to modify the source programs.

   (a) A non-standard clause:

   RECORDING MODE IS DATA-SENSITIVE RECORD LENGTH IS 81

   was added to the File Description (FD) for the card file.

   (b) The record description had to be modified by adding:

   02 FILLER PICTURE X.

   to the end of the record, thus increasing the record size to 81 characters.

   The system requires this additional character position in the record description entry in order to handle a delimiter character that it uses at the end of each card image record. This delimiter should have been transparent to the user.

   The subject system was able to handle the unlabeled tape but "could not find" the ANSI labeled tape. It was presumed that the problem resulted from the use of embedded space characters in the file-identifier field contained in the HDR1 label on the foreign tape.

2. Since the subject system does not support a card punch, no card deck could be punched and checked. However, the system was able to produce both an ANSI labeled tape and an unlabeled tape. Each was subsequently checked and found to be correct in format and content. No modifications to the source programs were required.

A. APPENDIX A. VALIDATION SUMMARY REPORT WORKING DOCUMENT

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.)

The reference documents for COBOL are American National Standard Programming Language COBOL (X3.23-1974), and Federal Standard COBOL (FIPS PUB 21-1).

A.1 Validation Environment

COMPILER IDENTIFICATION: AOS COBOL Revision 2.00

COMPUTER SYSTEM: Data General ECLIPSE C350

OPERATING SYSTEM: AOS Revision 2.03

APPENDIX A

A.2 Run Summaries

NUCLEUS MODULE

Nucleus Level 1

NC101 through NC108

A. Compilation.

No errors

B. Execution.

No errors

NC109

A. Compilation.

No errors

B. Execution.

(1) ACCEPT-TEST-7 failed. This test ACCEPTs the 10-character string "0123456789" into a field PICTUREd "9(9)". The resultant value in the field is expected to be 012345678 (right-end truncation). The Data General implementation treats the operation numerically, forces a decimal point alignment, and stores the result as 123456789. This treatment of the data is documented in the Data General COBOL manual as the desired procedure, and is acknowledged there as an extension to ANSI standard COBOL.

(2) NC109 outputs a number of messages via DISPLAY statements. It also requests early operator termination of the program via "STOP literal" statement. It was found that when the program was aborted, the last DISPLAY message was lost because it still remained in a system buffer and was not sent on to the printer. However, if the execution was permitted to proceed, all DISPLAY messages appeared. The standard does not prescribe the action to be taken regarding the contents of system buffers if a program is aborted by the operator.

NC110 through NC120

A. Compilation.

No errors

B. Execution.

No errors

NC151 through NC165

A. Compilation.

No errors

B. Execution.

No errors

APPENDIX A

SEQUENTIAL I-O MODULE

Sequential I-O Level 1

SQ101

A. Compilation.

No errors.

B. Execution.

SQ101 checks vertical spacing effected by the WRITE ... ADVANCING ... statement. Before the printed output from the program could be produced in the correct format, the system Forms Control Utility (FCU) had to be invoked to reprogram the printer's vertical forms unit. The top of the page had to be set at line 1 (nominally it is line 4), and the bottom of the page had to be set at line 66 (its default value), thus indicating that each of the 66 lines on the printer form was subject to use.

SQ102 through SQ118

A. Compilation.

No errors

B. Execution.

No errors

SQ119 and SQ120

A. Compilation.

No Errors

B. Execution.

Two modifications were made to SQ119 in order to correctly process the card deck associated with the ASCII validation. No changes should have been necessary.

The non-standard clause:

RECORDING MODE IS DATA-SENSITIVE RECORD LENGTH IS 81

was added to the FD and the Record length had to be increased by one character (02 FILLER PIC X(1) added to the record description) in order to process the data correctly.

SQ121

A. Compilation.

No errors

B. Execution.

No errors

SQ151 through SQ153

A. Compilation.

No errors

B. Execution.

No errors

APPENDIX A

TABLE HANDLING MODULE

Table Handling Level 1

TH101 through TH111

A. Compilation.

No errors

B. Execution.

No errors

TH151 and TH152

A. Compilation.

No errors

B. Execution.

No errors

APPENDIX A

GENERAL OBSERVATIONS

This compiler can only handle binary (COMPUTATIONAL) fields of 16 digits or less. COMPUTATIONAL items of longer lengths are handled internally as "COMPUTATIONAL-3", a packed decimal format.

A warning message:

WARNING: BINARY (COMP) ITEMS GREATER THAN 16 DIGITS HANDLED AS COMP-3

flagged fields beyond that limitation in the following programs:

NC101, NC103, NC105, NC106, NC117, NC118, NC119, NC120, NC153, NC154, NC155, NC156, NC163, NC164, NC165, TH102, TH104, TH151

No Errors resulted.