AD-A047 097 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER VALIDATION SUMMARY REPORT.(U)
NOV 77
UNCLASSIFIED CCVS74-VSR245 NL

1 OF 1
AD
A047 097

END
DATE
FILMED
1-78
DDC

CCVS74-VSR245

(1) 8

(6) COBOL COMPILER
VALIDATION SUMMARY REPORT.

(14) VALIDATION NUMBER CCVS74-VSR245 .

(11) 30 Nov 77

(12) 57 p.

Prepared By:

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

408438

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR245 ✓	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report #CCVS74-VSR245 (Assigned by Manager of Testing) Texas Instruments ASC COBOL			5. Report Date November 30, 1977 6.
7. Author(s) Same as organization - see 9.			8. Performing Organization Rept. No.
9. Performing Organization Name and Address Federal COBOL Compiler Testing Service ✓ Department of the Navy Washington, D. C. 20376			10. Project/Task/Work Unit No. 11. Contract/Grant No.
12. Sponsoring Organization Name and Address Automatic Data Processing Equipment Selection Office Department of the Navy (start here) Washington, D. C. 20376			13. Type of Report & Period Covered 14.

15. Supplementary Notes
cont fr p 4 → Scientific Computer (COBOL)

16. Abstracts

This Validation Summary Report (VSR) for the Texas Instruments Advanced Scientific COBOL (computer) Compiler Version 5.22 ± (OS ± Version ± 4.024) 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 Texas Instruments of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1974, 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. Key Words and Document Analysis. 17a. Descriptors

Porgramming Languages
Standards
Compilers
COBOL
Verifying
Proving Program Correctness
Software Engineering

17b. Identifiers/Open-Ended Terms

CCVS
CVS

17c. COSATI Field/Group 09/02

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

CCVS74-VSR245

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR245
2.	Vendor	Texas Instruments Incorporated
3.	Mainframe	Advanced Scientific Computer
4.	Compiler Identification	ASC COBOL Version 5.22
5.	Operating System Identification	System OS4.024
6.	Compiler Validation System Version Number	CCVS74 2.0
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 specific 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. Donald Caraway
Texas Instruments Incorporated
MS 2186
P.O. Box 2909
Austin, Texas 78767

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 Purpose of the Validation Summary Report

The purpose of ~~the~~ (This 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). This VSR for the Texas Instruments Advanced Scientific Computer → (cont on p. A)

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 21-1 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 for the Report Writer Module, which is not a part of Federal Standard COBOL.

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.

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 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 betweer 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 cn 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-[illegible] clause	Device capable of supporting the specified code.
MU[illegible]E 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 amoung 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 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 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.

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

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

SECTION 2. DETAILED EVALUATION OF ERRORS.

This section summarizes the results of the compilation and execution of the programs comprising the COBOL Compiler Validation System (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 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 Valication 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 identifiec 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 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
SM - 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.

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 mace up of the routine name followed by an A or B (A for compile time or syntax error and B for execution time or semantic error) and a number which makes the error unique in 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

- Language Concepts I-75
  - Characters used for words I-76
    - 0, 1, ..., 9
    - A, B, ..., Z
    - - (hyphen or minus)
  - Characters used for punctuation I-65
    - " quotation mark
    - (left parenthesis
    -) right parenthesis
    - . period
    - space
    - = equal sign
  - Characters used in editing. I-58
    - B space
    - 0 zero
    - + plus
    - - minus
    - CR credit
    - DB debit
    - Z zero suppress
    - * check protect
    - $ currency sign
    - , comma
    - . period
    - / stroke
  - Separators. I-75
    - The separators, semicolon and comma, are not allowed II-1
  - Character-strings I-76
  - COBOL words I-76
    - Not more than 30 characters
    - User-defined words. I-76
      - data-name
        - Must begin with an alphabetic character II-1
        - Must be unique; may not be qualified. . II-1
      - level-number
      - mnemonic-name
      - paragraph-name
      - program-name
      - routine-name
      - section-name
    - System-names. I-78
      - computer-name
      - implementor-name
      - language-name
    - Reserved words. I-79
      - Key words
      - Optional words
      - Figurative constants. I-80
        - ZERO
        - SPACE
        - HIGH-VALUE
        - LOW-VALUE

2.1.1 ---

*The addition of signed numeric data items with and without the *SEPARATE clause did not give the correct results. (NC118 B)

---

identifier/literal series
TO identifier
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase
The ALTER statement (only one procedure-name). . II-57
The DISPLAY statement (only one transfer of data) II-59
The DIVIDE statement II-61

2.1.2 ----------

*The division of two signed numeric data items did not
*give the correct result to a signed numeric data item
*containing a SEPARATE clause. (NC117 B)

INTO identifier
BY identifier/literal
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase
The ENTER statement II-63
The EXIT statement II-64
The GO TO statement (procedure-name is required) II-65
DEPENDING ON phrase
The IF statement (statements must be imperative) II-66
ELSE phrase
The INSPECT statement (only single character
data item) II-68
TALLYING phrase
ALL
LEADING
CHARACTERS
REPLACING phrase
ALL
LEADING
FIRST
CHARACTERS
TALLYING and REPLACING phrases
The MOVE statement II-74
TO identifier

2.1.3 ----------

*A group level MOVE of alphanumeric data items containing the
*figurative constants ZERO, QUOTE, and SPACE did not execute
*correctly. (NC103 B)

2.1.4 ----------

*The MOVE of a signed numeric data item with a SEPARATE clause
*to an alphanumeric edited data item containing the B and /
*editing symbols did not execute correctly. (NC114 B)

2.1.5 ----------

*The MOVE of a signed numeric data item with a SEPARATE clause
*to a signed numeric data item without a SEPARATE clause did
*not execute correctly. (NC116 B1)

2.1.6 ----------

*The MOVE of a signed numeric data item without a SEPARATE clause
*to a signed numeric data item with a SEPARATE clause did not
*execute correctly. (NC116 B2)

2.1.7 ----------

*The MOVE of a signed numeric data item with a SEPARATE clause to
*an alphanumeric data item caused the sign to be moved. (NC116 B3)

---

identifier series
The MULTIPLY statement II-77

2.1.8

---

*The multiplication of a signed numeric data item and a signed
*numeric edited data item did not give the correct result tc a
*signed numeric data item. (NC120 B)

BY identifier
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase
The PERFORM statement. II-78
procedure-name
THRU phrase
TIMES phrase
The STOP statement II-85
literal
RUN
The SUBTRACT statement II-89

2.1.9

---

*The subtraction of signed numeric data items from the figurative
*constant ZERO did not give the correct result to a signed
*numeric data item with a SEPARATE clause. (NC119 B)

---

identifier/literal series
FROM identifier
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase

2.2 TABLE HANDLING LEVEL 1

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

2.2.1 ----------

*All subscripted references to a two dimensional table containing
*numeric COMPUTATIONAL items and to a three dimensional table
*containing signed numeric items did not give the correct results.
* (TH109 B)

Indexing - 3 levels I-89

Data Division
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
UP BY identifier/integer
DOWN BY identifier/integer

2.2.2 ----------

*The use of positive signed numeric literals to initialize the
*increment value of the SET UP BY identifier and SET DOWN BY
*identifier statement failed to give the correct results. (TH111 B)

---

index-name series

2.3 SEQUENTIAL I-O LEVEL 1

Language Concepts
User-defined words I-76
file-name
record-name
I-O status IV-1

Environment Division
The FILE-CONTROL paragraph IV-4
The file control entry IV-4
SELECT clause
ASSIGN TO implementor-name clause
ORGANIZATION IS SEQUENTIAL clause
ACCESS MODE IS SEQUENTIAL clause
FILE STATUS clause
The I-O-CONTROL paragraph. IV-6
RERUN clause

2.3.1 ----------
*Unable to force RERUN due to lack of system checkpoint
*capability. (SQ113 B)

SAME AREA clause
SAME AREA series

Data Division
File Section IV-9
The file description entry IV-10
The record description entry IV-9
The BLOCK CONTAINS clause. IV-11
integer CHARACTERS
integer RECORDS
The CODE-SET clause. IV-12

2.3.2 ----------
*The ASCII (STANDARD-1) character code set was not processed
*correctly for a punched card file. (SQ119 B, SQ120 B)
----
*A punched card file created using the ASCII (STANDARD-1) character
*code set could not be processed correctly as ASCII code on a
*foreign computer system. (SQ118 B)

The DATA RECORDS clause. IV-13
data-name
data-name series
The LABEL RECORDS clause IV-14
STANDARD
OMITTED
The RECORD CONTAINS clause IV-18
integer-1 TO integer-2 CHARACTERS
The VALUE OF clause. IV-19
implementor-name IS literal
implementor-name IS literal series

Procedure Division
The CLOSE statement (only a single file-name may appear

2.4 RELATIVE I-O LEVEL 1

2.5 SEGMENTATION LEVEL 1

2.5.1 ----------

*Level 1 of the Segmentation module is not implemented on this system.

Language Concepts
User-defined words. I-76
segment-number

Procedure Division
Segment-numbers IX-4
Fixed segment-number range 0 through 49
Non-fixed segment-number range 50 through 99
All sections with the same segment-number must
be together in the source program

2.6 LIBRARY LEVEL 1

2.7 DEBUG LEVEL 1

2.7.1 ---

*Level 1 of the Debug module is not implemented on this system.

2.8 INTER-PROGRAM COMMUNICATIONS LEVEL 1

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.

3.1.1 Low Level

2.1.1 Addition of signed numeric data.
2.1.2 Division of signed numeric data.
2.1.3 Group level alphanumeric MOVE with figurative constants.
2.1.4 MOVE of signed numeric data with a SEPARATE clause to alphanumeric edited field.
2.1.5 MOVE signed numeric data with SEPARATE clause to signed numeric field.
2.1.6 MOVE signed numeric data to signed numeric field with SEPARATE clause.
2.1.7 MOVE signed numeric data with a SEPARATE clause to an alphanumeric field.
2.1.8 Multiplication of signed numeric data.
2.1.9 Subtraction of signed numeric data.
2.2.1 Subscripted reference to 2 and 3 dimensional tables containing COMPUTATIONAL and signed numeric data.
2.2.2 Initialization of SET UP BY and SET DOWN BY increment value with positive signed numeric literals.
2.3.1 No RERUN capability for sequential files.
2.3.2 ASCII (STANDARD-1) character code set not processed correctly for input and output punched card files.

3.1.2 Low-Intermediate Level

2.4.1 No RERUN capability for relative files.
2.4.2 Multiple record blocks for relative files with variable length records can not be processed.
2.5.1 Segmentation level 1 is not implemented.
2.7.1 Debug module level 1 is not implemented.

3.2 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.

Report Writer module is not implemented.

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.)

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

Options specified: APOST

Options defaulted: LIST, NOMAP, DECK, NOSEQ, NOXREF, NODECK, SUPPRESS

2. Environment Division implementor-names.

Printer destined files

```
ACCESS-NAME PRINTER
```

Tape files

```
ACCESS-NAME 'FILE$nnn'
```

Sequential Mass-storage files

```
ACCESS-NAME 'FILE$nnn'
```

Random Access files

```
ACCESS-NAME 'FILE$nnn'
```

Switch names

```
SWITCH A
SWITCH $
```

Source Computer names

```
ASC
```

Object Computer names

ASC

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.

There is no specific optimization feature for this compiler.

4. Compiler.

Texas Instrument Incorporated, ASC COBOL, Version 5.22

5. Operating system.

Texas Instrument Incorporated, System OS4.024

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) X3.4-1968. (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, X3.27-1969.

5. American National Standard Recorded Magnetic Tape for Information Interchange (800 CPI, NZRI), X3.22-1967.

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

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 Inter-

change" 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 ECF1 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. The Texas Instruments ASC system processed the ANSI unlabeled magnetic tape correctly, but was unable to properly read the card deck. Only 9 of the 50 cards processed as ASCII data. However, when the CODE-SET clause was removed from the file description all data was processed correctly.

2. The Texas Instruments ASC system was unable to produce an ANSI punched card deck which could be verified later on another system as being correct in ANSI format and character set. The card deck contained only unintelligible data. However, on a second run when the CODE-SET clause was removed from the file description the card deck was verified to contain correct data.

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.)

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

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	Texas Instruments, Incorporated, ASC COBOL Version 5.22
COMPUTER SYSTEM:	Advanced Scientific Computer
OPERATING SYSTEM:	System OS4.024

DEBUG MODULE Level 1

DB101 thru DB103, DB105

The test set for this module was not run since the standard DEBUG MODULE has not been implemented in this compiler.

INTER-PROGRAM COMMUNICATION MODULE Level 1

IC101 thru IC115

A. Compilation:

No errors.

B. Execution:

No failures.

IC151 thru IC152

A. Compilation:

No errors.

B. Execution:

No failures.

LIBRARY MODULE Level 1

LB101 thru LB107

A. Compilation:

No errors.

B. Execution:

No failures.

NUCLEUS MODULE Level 1

NC101 thru NC102

A. Compilation:

No errors.

B. Execution:

No failures.

NC103

A. Compilation:

No errors.

B. Execution

IF--TEST-96 failed when the following two alphanumeric group level data items did not compare equal.

```
02 GROUP-X1000.
   03 FILLER PIC X (500) VALUE ZERO.
   03 XNAME PICTURE X(100) VALUE QUOTE.
   03 FILLER PICTURE X(399) VALUE SPACE.
   03 FILLER PICTURE X VALUE  '.' .
02 GROUP-X500-2.
   03 GROUP-X500 PICTURE X(500) VALUE ZERO.
   03 GROUP-X500-1.
      04 FILLER PICTURE X(50) VALUE QUOTE.
      04 FILLER PICTURE X(50) VALUE QUOTE.
      04 FILLER PICTURE X(398) VALUE SPACE.
      04 FILLER PICTURE XX VALUE ' .' .
```

NC104 thru NC113

A. Compilation:

No errors.

B. Execution:

No failures.

NC114

A. Compilation:

No errors.

B. Execution:

MOVE-TEST-17 failed to properly move a signed numeric data item with a SEPARATE clause to an alphanumeric edited data item containing the B and / editing symbols. Refer to 5.15 (4)a, page II-75.

NC115

A. Compilation:

No errors.

B. Execution:

No failures.

NC116

A. Compilation:

No errors.

B. Execution:

1) MOVE-TEST-04.02 and MOVE-TEST-04.04 failed when the move of a signed numeric data item with a SEPARATE clause to a signed numeric data item without a SEPARATE clause caused the sign to overlay the rightmost digit.

	Computed Data	Correct Data
MOVE-TEST-04.02	000000000000009129	+91275
MOVE-TEST-04.04	000000000000080368	+80361

2) MOVE-TEST-05.02 and MOVE-TEST-05.04 failed when the move of a signed numeric data item without the SEPARATE clause to a signed numeric data item with a SEPARATE clause caused the sign to overlay the rightmost digit in addition to the correct placement of the respective LEADING and TRAILING signs.

	Computed Data	Correct Data
MOVE-TEST-05.02	-362J-	-03621
MOVE-TEST-05.04	0362J-	03621-

3) MOVE-TEST-06.02 failed when the move of a signed numeric data item with a SEPARATE clause to an alphanumeric data item caused the sign to be moved. Refer to 5.15.4(4)a., page II-75.

	Computed Data	Correct Data

MOVE-TEST-06.02	8036-	8036

NC117

A. Compilation:

No errors.

B. Execution:

DIV-TEST-30 and DIV-TEST-40 failed when the division of two signed numeric data items containing the "P" and "V" symbols failed to give the correct result to a signed numeric data item containing a SEPARATE clause.

	Computed Data	Correct Data
DIV-TEST-30	000000000000000001	000000000000000010
DIV-TEST-40	000000000000000001	000000000000000010

NC118

A. Compilation:

No errors.

B. Execution:

ADD-TEST-12, 13, 15, 17, 34, 43, and 45 failed to produce the correct results. All these tests involve the use of sigred numeric data items defined with and without the SIGN and SEPARATE clauses.

	Computed Data	Correct Data
ADD-TEST-12	000000000000001122	000000001111122222
ADD-TEST-13	000000000000001123	000000001111122233
ADD-TEST-15	000044777.777443+00	000344777.777443000
ADD-TEST-17	000333333.000000000	000333334.000000000
ADD-TEST-34	- 00000000000000000	222222222222222222
ADD-TEST-43	- 000000000000000000	222222222222222222
ADD-TEST-45	666667444444444446	66666744444444443

NC119

A. Compilation:

No errors.

B. Execution:

SUB-TEST-12 failed when the subtraction of various signed numeric

data items with and without SIGN and SEPARATE clauses did not give the correct result to a signed numeric data item with SIGN and SEPARATE clauses.

	Computed Data	Correct Data
SUB-TEST-12	-000044777.777443-00	-00344777.777443000

NC120

A. Compilation:

No errors.

B. Execution:

MPY-TEST-12 failed when the multiplication of two signed numeric data items did not give the correct result to a signed numeric data item. Only one of the multiplicands contains SIGN IS SEPARATE clauses.

	Computed Data	Correct Data
MPY-TEST-12	0000000000000001555	0000000000000111

NC151 thru NC153

A. Compilation:

No errors.

B. Execution:

No failures.

NC154

A. Compilation:

No errors.

B. Execution:

Same as NC103

NC155 thru NC165

A. Compilation:

No errors.

B. Execution:

No failures.

NUCLEUS MODULE Level 2

NC201

A. Compilation:

No errors.

B. Execution:

No failures.

NC202

A. Compilation:

No errors.

B. Execution:

1) ADD-TEST-11 and SUB-TEST-14 failed to ADD/SUBTRACT CORRESPONDING a series of signed numeric data items TO/FROM a series of unsigned numeric data items. One of the unsigned numeric data items contained the "P" symbol.

	Computed Data	Correct Data
ADD-TEST-11	11223344066677889900	11223344606677889900
SUB-TEST-14	11223344053477889900	11223344606677889900

2) SUB-TEST-4 which subtracts a signed numeric data item and a negative numeric literal from a series of three signed numeric data items failed to produce the correct results in one of the receiving operands.

	Computed Data	Correct Data
SUB-TEST-4	000000000.000000000	000000099.000000000

As a result of the failure in SUB-TEST-4 the ON SIZE ERROR did not OCCUR and therefore SUB-TEST-7 also failed.

NC203

A. Compilation:

1) The compiler correctly inserted the compile date in the DATE-COMPILED paragraph but failed to replace the entire DATE-COMPILED paragraph with the current date.

2) The compiler flagged the data-name assignments of SUB-GRP-FOR-RENAMES-1 and NAME-2 with the following message:

RENAMES REFERENCE SUBORDINATE TO OTHER ITEM.
ASSUME RENAMES SINGLE ITEM:

3) The following DIVIDE statement caused the compiler to abort.

DIVIDE DIV19 INTO DIV20 GIVING DIV21 ROUNDED
REMAINDER DIV22.

B. Execution:

No execution due to compiler abort indicated by A.3 above.

NC204 thru NC208

A. Compilation:

No errors.

B. Execution:

No failures.

NC209

A. Compilation

1) The compiler flagged the data-name assignment of AL with the following message:

RENAMES REFERENCE SUBORDINATE TO OTHER ITEM. ASSUME RENAMES SINGLE ITEM:

2) The following MOVE statement caused the compiler to abort:

MOVE CORRESPONDING C-LEVEL IN A-LEVEL TO C-FLOCK (4).

B. Execution:

No execution due to compiler abort indicated by A.2 above.

NC210 thru NC218

A. Compilation:

No errors.

B. Execution:

No failures.

RELATIVE I-O MODULE Level 1

RL101 thru RL105

A. Compilation:

No errors.

B. Execution:

No failures.

RL106

A. Compilation:

No errors.

B. Execution:

REL-TEST-18 was unable to read the 12th variable length record from a randomly accessed file which is blocked three records to a block. The INVALID KEY condition was taken. When the blocking factor was changed to one, REL-TEST-18 passed.

RL107 thru RL108

A. Compilation:

No errors.

B. Execution:

No failures.

RL109

A. Compilation:

No errors.

B. Execution:

1) No execution failures.

2) Unable to force RERUN of RL109 due to lack of checkpoint capability within operating system.

RL151 thru RL153

A. Compilation:

No errors:

B. Execution:

No failures.

SEGMENTATION MODULE Level 1

SG101 thru SG103

The test for this module was not run since the standard SEGMENTATION MODULE has not been implemented in this compiler.

SEQUENTIAL I-O MODULE Level 1

SQ101 thru SQ112

A. Compilation:

No errors.

B. Execution:

No failures.

SQ113

A. Compilation:

No errors.

B. Execution:

1) No execution failures.

2) Unable to force RERUN of SQ113 due to lack of checkpoint capability within operating system.

SQ114 thru SQ117

A. Compilation:

No errors.

B. Execution:

No failures.

SQ118

A. Compilation:

No errors.

B. Execution:

The card file created in SQ118 failed to punch the required ASCII character data in the card. When the card file was verified by FCCTS on another computer system all data printed as meaningless information. However, when the program was rerun without the CODE-SET clause in the FD of the card file, all data was created properly.

SQ119

A. Compilation:

No errors.

B. Execution:

The punched card file provided by FCCTS failed to be processed as ASCII character data. 41 of the 50 record comparisons failed. However, when the program was rerun without the CODE-SET clause in the FD of the card file, all data was processed correctly.

Computed Data	Correct Data
@@@@@@	
LLLLLL	$$$$$$
MMMMMM	((((((
	))))))
	******
NNNNNN	++++++
kkkkkk	,,,,,,

KKKKKK	......
aaaaaa	//////
pppppp	000000
qqqqqq	111111
rrrrrr	222222
ssssss	333333
tttttt	444444
uuuuuu	555555
vvvvvv	666666
wwwwww	777777
xxxxxx	888888
yyyyyy	999999
	;;;;;;
LLLLLL	<<<<<<
	======
nnnnnn	>>>>>>
AAAAAA	AAAAAA
BBBBBB	BBBBBB
CCCCCC	CCCCCC
DDDDDD	DDDDDD
EEEEEE	EEEEEE
FFFFFF	FFFFFF
GGGGGG	GGGGGG
HHHHHH	HHHHHH
IIIIII	IIIIII
QQQQQQ	JJJJJJ
RRRRRR	KKKKKK
SSSSSS	LLLLLL
TTTTTT	MMMMMM
UUUUUU	NNNNNN
VVVVVV	000000
WWWWWW	PPPPPP

XXXXXX	QQQQQQ
YYYYYY	RRRRRR
bbbbbb	SSSSSS
cccccc	TTTTTT
dddddd	UUUUUU
eeeeee	VVVVVV
ffffff	WWWWWW
gggggg	XXXXXX
hhhhhh	YYYYYY
iiiiii	ZZZZZZ

SQ120

A. Compilation:

No errors.

B. Execution:

Same as SQ119

SQ121

A. Compilation:

No errors.

B. Execution:

No failures.

SQ151 thru SQ153

A. Compilation:

No errors.

B. Execution:

No failures.

TABLE HANDLING MODULE Level 1

TH101 thru TH108

A. Compilation:

No errors.

B. Execution:

No failures.

TH109

A. Compilation:

No errors.

B. Execution:

SEP-TEST-002.50, 3.50, 5.01 through 5.10, 6.01 through 6.07, and 7.02 through 7.07 all failed to correctly reference data from signed two and three dimensional tables.

TH110

A. Compilation:

No errors.

B. Execution:

No failures.

TH111

A. Compilation:

No errors.

B. Execution:

SET-TEST-003.01 through 003.04 and SET-TEST-004.01 through 004.04 all failed to correctly SET an index-name UP BY or DOWN BY a signed numeric data item. In each case a positive signed numeric literal is used to initialize the increment value.

TH151 thru TH152

A. Compilation:

No errors.

B. Execution:

No failures.

TABLE HANDLING MODULE Level 2

TH201 thru TH220

A. Compilation:

No errors.

B. Execution:

No failures.