AD-A041 563 FEDERAL COBOL COMPILER TESTING SERVICE WASHINGTON D C F/G 9/2
COBOL COMPILER. VALIDATION SUMMARY REPORT.(U)
JUN 77

UNCLASSIFIED CCVS74-VSR225 NL

| OF |
ADA
041563

END

DATE
FILMED
7-77

AD A041563

FEDERAL COBOL COMPILER TESTING SERVICE

VALIDATION SUMMARY REPORT

(10)

AD No.
DDC FILE COPY

Department of the Navy
(ADPESO)

Washington, D.C.
20376

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC
JUL 14 1977
B

(14) CCVS74-VSR225

(11) 30 Jun 77

(12) 29 p.

(6) COBOL COMPILER
VALIDATION SUMMARY REPORT

VALIDATION NUMBER CCVS74-VSR225

Prepared By:

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

DDC
RECEIVED
JUL 14 1977
B

AD A048438

CCVS74-VSR225

COBOL COMPILER VALIDATION

1.	Validation Number	CCVS74-VSR225
2.	Vendor	IBM Corporation
3.	Mainframe	IBM 370/158
4.	Compiler Identification	IBM OS/VS COBOL Release 2.0 MOD 0
5.	Operating System Identification	MVS 3.7 VS2 with SU8
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 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:

Jay Valentine
Federal Support Center
IBM Corporation
10401 Fernwood Road
Bethesda, Maryland 20034

TABLE OF CONTENTS

SECTION 1.		INTRODUCTION.	4
	1.1	Purpose of the Validation Summary Report.	4
	1.2	Preparation of the VSR.	4
	1.3	Organization of the VSR	4
	1.4	Abstract Covering Compliance to ANS COBOL	5
	1.5	Federal Standard COBOL.	10
	1.6	Use of the VSR.	12
	1.7	Sources of Additional Information	12
	1.8	Requests for Interpretation	12
	1.9	Modules and Language Elements Excluded from Testing	13
	1.10	Timeliness of the Validation Summary Report	14
SECTION 2.		DETAILED EVALUATION OF ERRORS	15
	2.1	Nucleus Level 1	18
	2.2	Nucleus Level 2	23
	2.3	Table Handling Level 1.	26
	2.4	Table Handling Level 2.	27
	2.5	Sequential I-O Level 1.	28
	2.6	Sequential I-O Level 2.	30
	2.7	Relative I-O Level 1.	31
	2.8	Relative I-O Level 2.	33
	2.9	Indexed I-O Level 1	34
	2.10	Indexed I-O Level 2	36
	2.11	Sort-Merge Level 1.	37
	2.12	Sort-Merge Level 2.	38
	2.13	Report Writer Level 1	39
	2.14	Segmentation Level 1.	41
	2.15	Segmentation Level 2.	42
	2.16	Library Level 1	43
	2.17	Library Level 2	44
	2.18	Debug Level 1	45
	2.19	Debug Level 2	46
	2.20	Inter-Program Communication Level 1	47
	2.21	Inter-Program Communication Level 2	48
	2.22	Communication Level 1	49
	2.23	Communication Level 2	51
SECTION 3.		COMPILER STATUS	52
	3.1	Federal Standard COBOL.	52
	3.2	American National Standard COBOL.	52
SECTION 4.		SOFTWARE ENVIRONMENT.	53
SECTION 5.		ASCII VALIDATION.	56
	5.1	Purpose of ASCII Validation	56
	5.2	Applicable ANSI Standards	56
	5.3	ASCII Validation Process.	57
	5.4	Results for This Validation	57
APPENDIX A -		VALIDATION SUMMARY WORKING DOCUMENT	58

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 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 discrepan-

cies 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 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	Mass storage

(Sequential I-O only)	
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 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

diagrcsed. 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 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 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 identified by a 5-character program name. The name associates the routine with the functional processing module and level of American National Standard Programming Language COBOL tested within the program.

The five character name has the general format 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.

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 Stancard Programming Language COBOL. The reference at the end of the description of the error is to Appendix A which contains the detailed information collected during the validation. The reference is made up of the routine name followed by an A or B (A for compile time or syntax 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
QUOTE
Special-character words I-80
Literals. I-80
Nonnumeric literals have lengths from 1 through 120 characters
Numeric literals have lengths from 1 through 18 digits
PICTURE character-strings I-82
Comment-entries I-82

Reference format. I-105
Sequence number I-105
Area A. I-105
Division header I-106
Section header. I-106
Paragraph header. I-107
Data Division entries I-107
Area B. I-105
Paragraphs. I-107
Data Division entries I-107
Continuation of lines I-106

2.1.1 ----------

* Continuaton of the section header WORKING-STORAGE was rot
* accepted.
* (NC113.A)

Only nonnumeric literals may be continued . . . II-1
Comment lines I-108
Asterisk (*) comment lines
Stroke (/) comment line

Identification Division I-94
The PROGRAM-ID paragraph. II-3
The AUTHOR paragraph. II-2
The INSTALLATION paragraph. II-2
The DATE-WRITTEN paragraph. II-2
The SECURITY paragraph. II-2

Environment Division. I-95
The SOURCE-COMPUTER paragraph II-5
computer-name
The OBJECT-COMPUTER paragraph II-6
computer-name
MEMORY SIZE clause
PROGRAM COLLATING SEQUENCE clause
The SPECIAL-NAMES paragraph II-8
implementor-name IS mnemonic-name
implementor-name IS mnemonic-name series
ON STATUS

OFF STATUS
alphabet-name clause
CURRENCY SIGN clause
DECIMAL-POINT clause

Data Division I-97
Working-Storage Section II-11
The data description entry. II-12
The BLANK WHEN ZERO clause. II-14
The data-name or FILLER clause. II-15
The JUSTIFIED clause (may be abbreviated JUST). II-16
Level-number. II-17
01 through 10 (level numbers must be 2 digits) II-13
77. . . . . II-11
The PICTURE clause (may be abbreviated PIC) . . II-18
Character-string may contain 30 characters. . II-18
Data characters: A X 9 . . . . . II-18
Operational symbols: S V P II-21
Fixed insertion characters. II-21
0 (may be used only in edited items)
,
B (may be used only in edited items)
.
$ (currency sign)
+ and -
DB and CR
/
Replacement or floating characters. II-21
$ (currency sign)
+ and -
Z
*
Currency sign substitution. II-21
Decimal point substitution. II-21
The REDEFINES clause (may not be nested). . . . II-27
The SIGN clause II-31
The SYNCHRONIZED clause (may be abbreviated SYNC) II-33
The USAGE clause. II-35
COMPUTATIONAL (may be abbreviated COMP)
DISPLAY
The VALUE clause. II-36
literal

2.1.2 ----------

* The VALUE clause containing an alphanumeric literal for an
* alphanumeric-edited data item was not accepted.
* (ASVP9.A)

Procedure Division I-99
Conditional expressions II-41
Simple condition. II-41
Relation condition. II-41

Relational operators
[NOT] GREATER THAN
[NOT] LESS THAN
[NOT] EQUAL TO
Comparison of numeric operands. II-42
Comparison of nonnumeric operands (operands must be of equal size) II-42
Class condition II-43
NOT option
Switch-status condition II-44
The arithmetic statements II-51
Arithmetic operands limited to 18 digits
Overlapping operands. II-51
The ACCEPT statement (only one transfer of data) II-53
The ADD statement II-55
identifier/literal series
TO identifier

2.1.3 ----------

* The move of a group item to an alphanumeric-edited data item
* did not produce the expected result.
* (NC105 B.1)

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
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
identifier series
The MULTIPLY statement II-77
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
identifier/literal series
FROM identifier
GIVING identifier
ROUNDED phrase
SIZE ERROR phrase

2.2 NUCLEUS LEVEL 2

All elements of 1 NUC 1,2 are a part of 2 NUC 1,2

Language Concepts. I-75
  Characters used for punctuation. I-65
    , comma
    ; semicolon
  Characters used for arithmetic operations. I-52
    + addition
    - subtraction
    * multiplication
    / division
    ** exponentiation
  Characters used in relations I-66
    = equal to
    > greater than
    < less than
  Separators I-75
    The separators, semicolon and comma, are allowed . . II-1
  Character-strings. I-76
    COBOL words. I-76
      User-defined words I-76
        condition-name
        data-name
          Need not begin with an alphabetic character. . II-1
          May be qualified if necessary for uniqueness . II-1
      Reserved words I-79
        Figurative constants I-80
          ZEROS; ZEROES
          SPACES
          HIGH-VALUES
          LOW-VALUES
          QUOTES
          ALL literal
        Connectives I-79
          Qualifier connectives: OF, IN
          Series connectives: , (separator comma)
              and ; (separator semicolon)
          Logical connectives: AND, OR, AND NOT, OR NOT
    Qualification I-87

Reference format I-105
  Continuation of lines (continuation of words and
    numeric literals is allowed) II-1

2.2.1 ----------
* The compiler did not accept continuation of a numeric literal
* used as an operand in a MOVE statement.
* (NC205 A.1)

2.2.2 ----------
* The compiler did not accept an IF statement which contained
* words that extended across several source lines.

* (NC205 A.2)

Identification Division. I-94
  The DATE-COMPILED paragraph. II-4

Environment Division
  The SPECIAL-NAMES paragraph. II-8
    alphabet-name clause
      literal

Data Division. I-97
  The data description entry II-12
  Level-number II-17
    01 through 49 (level-numbers may be 1 or 2 digits) .
    66
    88
  The REDEFINES clause (may be nested) II-27
  The RENAMES clause (may be nested) II-29
    data-name
    data-name THRU data-name
  The VALUE clause II-36
    literal-1, literal-2
    literal-1 THRU literal-2
    literal range series

Procedure Division I-99
  Arithmetic expressions II-39
  Conditional expressions. II-41
    Simple condition II-41
      Relational condition II-41
        Relational operators
          [NOT] =
          [NOT] >
          [NOT] <
        Comparison of nonnumeric operands (operands of
            unequal size are allowed). II-42
      Condition-name condition II-44
      Sign condition II-44
        NOT option
    Complex condition. II-45
      Logical operators AND, OR, and NOT
      Negated simple condition II-46
      Combined and negated combined conditions II-46
    Abbreviated combined relation condition. II-47
  Multiple results in arithmetic statements. II-51
  The ACCEPT statement (no restrictions on the number
      of transfers of data). II-53
    FROM phrase
  The ADD statement. II-55
    TO identifier series
    GIVING identifier series

CORRESPONDING phrase
The ALTER statement. II-57
The series option is allowed
The COMPUTE statement. II-58
identifier series
ROUNDED phrase
SIZE ERROR phrase
The DISPLAY statement (no restrictions on the number of transfers of data). II-59
UPON phrase
The DIVIDE statement II-61
INTO identifier series
GIVING identifier series
REMAINDER phrase
The GO TO statement (procedure-name may be omitted). . II-65
The IF statement (nested statements) II-66
The INSPECT statement (multi-character data items) . . II-68
series
The MOVE statement II-74
CORRESPONDING phrase
The MULTIPLY statement II-77
BY identifier series
GIVING identifier series
The PERFORM statement. II-78
UNTIL phrase
VARYING phrase
The STRING statement II-86
DELIMITED series
POINTER phrase
ON OVERFLOW phrase
The SUBTRACT statement II-89
FROM identifier series
GIVING identifier series

2.2.3 ----------

* A data item containing an edited picture character-string which
* was the subject of the GIVING phrase did not produce the expected
* result.
* (NC213.B)

CORRESPONDING phrase
The UNSTRING statement II-91
DELIMITED BY phrase
POINTER phrase
TALLYING phrase
ON OVERFLOW phrase

2.3 TABLE HANDLING LEVEL 1

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

Data Division
- 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
  - index-name series

2.4 TABLE HANDLING LEVEL 2

All elements of 1 TBL 1,2 are a part of 2 TBL 1,2

Data Division
- The OCCURS clause III-2
  - integer-1 TO integer-2 DEPENDING ON data-name
  - ASCENDING/DESCENDING data-name
    - data-name series
  - ASCENDING/DESCENDING series

Procedure Division
- The SEARCH statement. III-7
  - VARYING phrase
  - AT END phrase
  - WHEN phrase
- The SEARCH ALL statement. III-7
  - AT END phrase
  - WHEN phrase

2.5 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
  - SAME AREA clause
  - SAME APEA 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
- 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 in a CLOSE statement). IV-20
  - REEL
  - UNIT
- The OPEN statement (only a single file-name may appear in an OPEN statement). IV-24
  - INPUT
  - OUTPUT
  - I-O
- The READ statement IV-28

INTO identifier
AT END phrase
The REWRITE statement. IV-31
FROM identifier
The USE statement. IV-32
EXCEPTION/ERROR PROCEDURE
ON file-name
ON INPUT
ON OUTPUT
ON I-O
The WRITE statement IV-34
FROM identifier
BEFORE/AFTER integer LINES
BEFORE/AFTER PAGE

2.6 SEQUENTIAL I-O LEVEL 2

All elements of 1 SEQ 1,2 are a part of 2 SEQ 1,2

Language Concepts
- Special register I-80
  - LINAGE-COUNTER. IV-3

Environment Division
- The FILE-CONTROL paragraph IV-4
- The file control entry IV-4
  - SELECT clause
    - OPTIONAL phrase
  - RESERVE integer AREA(S) clause
- The I-O-CONTROL paragraph. IV-6
  - SAME RECORD AREA clause
  - SAME RECORD AREA series
  - MULTIPLE FILE TAPE clause

Data Division
- The file description entry. IV-10
- The BLOCK CONTAINS clause IV-11
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The LINAGE clause IV-15
  - FOOTING phrase
  - TOP phrase
  - BOTTOM phrase
- The VALUE OF clause IV-19
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The CLOSE statement IV-20
  - NO REWIND, REMOVAL, or LOCK
  - file-name series
- The OPEN statement. IV-24
  - INPUT
    - REVERSED
    - NO REWIND
  - OUTPUT
    - NO REWIND
  - EXTEND
  - file-name series
  - INPUT, OUTPUT, I-O, and EXTEND series
- The USE statement IV-32
  - EXCEPTION/ERROR PROCEDURE ON file-name series
  - EXCEPTION/ERROR PROCEDURE ON EXTEND
- The WRITE statement IV-34
  - BEFORE/AFTER identifier LINES
  - BEFORE/AFTER mnemonic-name
  - AT END-OF-PAGE imperative-statement

2.7 RELATIVE I-O LEVEL 1

Language Concepts
- User-defined words. I-76
  - file-name
  - record-name
- I-O status. V-2

Environment Division
- The FILE-CONTROL paragraph. V-5
- The file control entry. V-5
  - SELECT clause
  - ASSIGN TO implementor-name clause
  - ORGANIZATION IS RELATIVE clause
  - ACCESS MODE clause
    - SEQUENTIAL
    - RANDOM
  - FILE STATUS clause
- The I-O-CONTROL paragraph V-7
  - RERUN clause
  - SAME AREA clause
  - SAME AREA series

Data Division
- File Section. V-10
- The file description entry. V-11
- The record description entry. V-10
- The BLOCK CONTAINS clause V-12
  - integer CHARACTERS
  - integer RECORDS
- The DATA RECORDS clause V-13
  - data-name
  - data-name series
- The LABEL RECORDS clause. V-14
  - STANDARD
  - OMITTED
- The RECORD CONTAINS clause. V-15
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause V-16
  - implementor-name IS literal
  - implementor-name IS literal series

Procedure Division
- The CLOSE statement V-17
  - WITH LOCK
  - file-name series
- The DELETE statement. V-19
  - INVALID KEY phrase
- The OPEN statement. V-20
  - INPUT
  - OUTPUT
  - I-O

file-name series
INPUT, OUTPUT, and I-O series
The READ statement. V-23
INTO identifier
AT END phrase
INVALID KEY phrase
The REWRITE statement V-26
FROM identifier
INVALID KEY phrase
The USE statement V-30
EXCEPTION/ERROR PROCEDURE
ON file-name
ON INPUT
ON OUTPUT
ON I-O
The WRITE statement. V-32
FROM identifier
INVALID KEY phrase

2.8 RELATIVE I-O LEVEL 2

All elements of 1 REL 0,2 are a part of 2 REL 0,2

Environment Division
- The FILE-CONTROL paragraph. V-5
- The file control entry. V-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
- The I-O-CONTROL paragraph V-7
  - SAME RECORD AREA
  - SAME RECORD AREA entries

Data Division
- The file description entry. V-11
- The BLOCK CONTAINS clause V-12
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause V-16
  - implementor-name IS data-name
  - implementor-name IS data-name entries

Procedure Division
- The READ statement. V-23
  - NEXT RECORD
- The START statement V-28
  - KEY IS phrase
  - INVALID KEY phrase
- The USE statement V-30
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.9 INDEXED I-O LEVEL 1

Language Concepts
- User-defined words. I-76
  - file-name
  - record-name
- I-O status. VI-2

Environment Division
- The FILE-CONTROL paragraph. VI-5
- The file control entry. VI-5
  - SELECT clause
  - ASSIGN TO implementor-name clause
  - ORGANIZATION IS INDEXED clause
  - ACCESS MODE clause
    - SEQUENTIAL
    - RANDOM
  - RECORD KEY clause
  - FILE STATUS clause
- The I-O-CONTROL paragraph VI-8
  - RERUN clause
  - SAME AREA clause
  - SAME AREA series

Data Division
- File Section. VI-11
- The file description entry. VI-12
- The record description entry. VI-11
- The BLOCK CONTAINS clause VI-13
  - integer CHARACTERS
  - integer RECORDS
- The DATA RECORDS clause VI-14
  - data-name
  - data-name series
- The LABEL RECORDS clause. VI-15
  - STANDARD
  - OMITTED
- The RECORD CONTAINS clause. VI-16
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause VI-17
  - implementor-name IS literal
  - implementor-name IS literal series

Procedure Division
- The CLOSE statement VI-18
  - WITH LOCK
  - file-name series
- The DELETE statement. VI-20
  - INVALID KEY phrase
- The OPEN statement. VI-21

2.9.1 ----------

* It appears that improper execution of the OPEN statement

```
*     may have invalidated all accesses made to the file by the
*     READ and WRITE statements.
*                                                        (IX205.8)
-----------------------------------------------------------------------
```


INPUT
OUTPUT
I-O
file-name series
INPUT, OUTPUT, and I-O series
The READ statement. VI-24
INTO identifier
AT END phrase
INVALID KEY phrase
The REWRITE statement VI-28
FROM identifier
INVALID KEY phrase
The USE statement VI-32
EXCEPTION/ERROR PROCEDURE
ON file-name
ON INPUT
ON OUTPUT
ON I-O
The WRITE statement VI-33
FROM identifier
INVALID KEY phrase

2.10 INDEXED I-O LEVEL 2

All elements of 1 INX 0,2 are a part of 2 INX 0,2

Environment Division
- The FILE-CONTROL paragraph VI-5
- The file control entry VI-5
  - SELECT clause
  - RESERVE integer AREA(S) clause
  - ACCESS MODE IS DYNAMIC clause
  - ALTERNATE RECORD KEY clause
    - WITH DUPLICATES phrase
- The I-O-CONTROL paragraph. VI-8
  - SAME RECORD clause
  - SAME RECORD AREA series

Data Division
- The file description entry VI-12
- The BLOCK CONTAINS clause. VI-13
  - integer-1 TO integer-2 RECORDS
  - integer-1 TO integer-2 CHARACTERS
- The VALUE OF clause. VI-17
  - implementor-name IS data-name
  - implementor-name IS data-name series

Procedure Division
- The READ statement VI-24
  - KEY IS phrase
  - NEXT RECORD
- The START statement. VI-30
  - KEY IS phrase
  - INVALID KEY phrase
- The USE statement. VI-32
  - EXCEPTION/ERROR PROCEDURE
    - ON file-name series

2.11 SORT-MERGE LEVEL 1

Language Concepts
- User-defined words. I-76
  - file-name

Environment Division
- The FILE-CONTROL paragraph. VII-2
- The file control entry. VII-2
  - SELECT clause
  - ASSIGN TO implementor-name clause

Data Division
- File Section. VII-5
- The sort-merge file description entry VII-5
- The DATA RECORDS clause VII-6
- The RECORD CONTAINS clause. VII-7

Procedure Division
- The RELEASE statement VII-12
  - FROM phrase
- The RETURN statement. VII-13
  - INTO phrase
  - AT END phrase
- The SORT statement (only one SORT statement, a STOP RUN statement, and any associated input-output procedures allowed in the nondeclarative portion of a program) VII-14
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - INPUT PROCEDURE phrase
    - THRU
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase

2.12 SORT-MERGE LEVEL 2

All elements of 1 SRT 0.2 are a part of 2 SRT 0.2

Environment Division
- The FILE-CONTROL paragraph. VII-2
- The file control entry. VII-2
  - SELECT clause
- The I-O-CONTROL paragraph VII-3
  - SAME RECORD AREA clause
  - SAME SORT/SORT-MERGE AREA clause
  - SAME series

Procedure Division
- The MERGE statement VII-8
  - KEY data-name
    - data-name series
  - ASCENDING series
  - DESCENDING series
  - mixed ASCENDING/DESCENDING
  - COLLATING SEQUENCE phrase
  - USING phrase
  - OUTPUT PROCEDURE phrase
    - THRU
  - GIVING phrase
- The SORT statement (multiple SORT statements are permitted). VII-14
  - COLLATING SEQUENCE phrase

2.13 REPORT WRITER LEVEL 1

2.13.1 ----------

```
*
*     The Report Writer Module was not run during the validation.
*
```

Language Concept
  User-defined words I-76
    file-name
    report-name
  Special registers I-80
    LINE-COUNTER VIII-1
    PAGE-COUNTER VIII-1

Data Division
  Report Section VIII-2
  The file description entry VIII-3
  The report description entry VIII-4
  The report group description entry VIII-6
  The BLOCK CONTAINS clause VIII-24
  The CODE clause VIII-25
  The CODE-SET clause VIII-26
  The COLUMN NUMBER clause VIII-27
  The CONTROL clause VIII-28
    data-name
    data-name series
    FINAL
    FINAL data-name series
  The data-name clause VIII-30
  The GROUP INDICATE clause VIII-31
  The LABEL RECORDS clause VIII-32
  The LINE NUMBER clause VIII-33
    integer
      NEXT PAGE
    PLUS integer
  The NEXT GROUP clause VIII-35
    integer
    PLUS integer
    NEXT PAGE
  The PAGE clause VIII-36
    integer LINES
    HEADING
    FIRST DETAIL
    LAST DETAIL
    FOOTING
  The PICTURE clause II-18
  The RECORD CONTAINS clause VIII-39
  The REPORT clause VIII-40
    report-name series
  The SOURCE clause VIII-41

The SUM clause VIII-42
UPON data-name series
RESET phrase
The TYPE clause VIII-45
REPORT HEADING (RH)
PAGE HEADING (PH)
CONTROL HEADING (CH)
DETAIL (DE)
CONTROL FOOTING (CF)
PAGE FOOTING (PF)
REPORT FOOTING (RF)
The VALUE IS clause II-36
The VALUE OF clause VIII-50

Procedure Division
The GENERATE statement VIII-51
report-name
data-name
The INITIATE statement VIII-53
report-name
The SUPPRESS statement VIII-54
report-name
The TERMINATE statement VIII-55
report-name series
The USE statement VIII-56
BEFORE REPORTING

2.14 SEGMENTATION LEVEL 1

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

2.15 SEGMENTATION LEVEL 2

All elements of 1 SEG 0,2 are a part of 2 SEG 0,2

Environment Division
The OBJECT-COMPUTER paragraph
SEGMENT-LIMIT. IX-5

Procedure Division
Segment-numbers IX-4
Sections with the same segment-number need not be physically contiguous in the source program

2.16 LIBRARY LEVEL 1

Language Concepts
User-defined words I-76
text-name

All divisions
The COPY statement X-2

2.17 LIBRARY LEVEL 2

All elements of 1 LIB 0.2 are a part of 2 LIB 0.2

Language Concepts
- User-defined words I-76
  - library-name

All divisions
- The COPY statement X-2
  - OF library-name
  - REPLACING phrase

2.18 DEBUG LEVEL 1

Language Concepts
Special registers. I-80
DEBUG-ITEM. XI-1

Environment Division
The SOURCE-COMPUTER paragraph
WITH DEBUGGING MODE clause. XI-3

Procedure Division
USE FOR DEBUGGING statement. XI-4
procedure-name
procedure-name series
ALL PROCEDURES
Debugging lines. XI-10

2.19 DEBUG LEVEL 2

All elements of 1 DEB 0,2 are a part of 2 DEB 0,2

Procedure Division
USE FOR DEBUGGING statement. XI-4
ALL REFERENCES OF identifier series
file-name series
cd-name series

2.20 INTER-PROGRAM COMMUNICATIONS LEVEL 1

Data Division
Linkage Section. XII-2

Procedure Division
Procedure Division header. XII-4
USING phrase
The CALL statement XII-5
literal
USING data-name series
The EXIT PROGRAM statement XII-8

2.21 INTER-PROGRAM COMMUNICATIONS LEVEL 2

All elements of 1 IPC 0.2 are a part of 2 IPC 0.2

Procedure Division
- The CALL statement. XII-5
  - identifier
  - ON OVERFLOW phrase
- The CANCEL statement. XII-7

2.22 COMMUNICATION LEVEL 1

```
*   The COMMUNICATION Module is not currently evaluated as
*   part of an official validation.  See Section 1.9.3.
```

Language Concepts
- User-defined words. I-76
  - cd-name

Data Division
- Communication Section XIII-2
- The communication description entry XIII-3
  - *FOR INPUT clause*
    - END KEY
    - MESSAGE COUNT
    - MESSAGE DATE
    - MESSAGE TIME
    - *SYMBOLIC QUEUE*
    - SYMBOLIC SOURCE
    - SYMBOLIC SUB-QUEUE-n
    - STATUS KEY
    - TEXT LENGTH
  - *FOR OUTPUT clause*
    - DESTINATION COUNT
    - DESTINATION TABLE
      - INDEXED BY
    - ERROR KEY
    - *SYMBOLIC DESTINATION*
    - STATUS KEY
    - TEXT LENGTH

Procedure Division
- *The ACCEPT MESSAGE COUNT statement.* XIII-12
- The DISABLE statement XIII-13
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- *The ENABLE statement.* XIII-15
  - INPUT
  - OUTPUT
  - KEY identifier/literal
- The RECEIVE statement XIII-17
  - MESSAGE
  - INTO identifier
  - NO DATA phrase
- The SEND statement. XIII-20
  - FROM identifier-1 WITH
  - WITH EMI
  - WITH EGI
  - BEFORE/AFTER ADVANCING

identifier-3 LINES
integer LINES
mnemonic-name
PAGE

2.23 COMMUNICATION LEVEL 2

```
-----------------------------------------------------------------
*    The COMMUNICATION Module is not currently evaluated as
*    part of an official validation.  See Section 1.9.3.
-----------------------------------------------------------------
```

All elements of 1 COM 0,2 are a part of 2 COM 0,2

Communication Section
- The communication description entry. XIII-3
  - FOR INPUT
    - INITIAL

Procedure Division
- The DISABLE statement. XIII-13
  - INPUT
    - TERMINAL
- The ENABLE statement XIII-15
  - INPUT
    - TERMINAL
- The RECEIVE statement. XIII-17
  - SEGMENT
- The SEND statement XIII-20
  - FROM identifier-1
  - WITH identifier-2
  - WITH ESI

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 Line continuation of the word WORKING-STORAGE was not accepted.
2.1.2 VALUE clause not accepted on an alphnumeric-edited cata item.
2.1.3 The move of a group item to an edited item gave incorrect results.

3.1.2 Low-Intermediate Level

None

3.1.3 High-Intermediate

2.2.1 Line continuation of numeric literals was not accepted.
2.2.2 Line continuation of certain COBOL words was not accepted.
2.2.3 An edited data item used as the receiving operand of a SUBTRACT statement gave incorrect results.

3.1.4 High Level

2.9.1 Improper execution of the OPEN appears to invalidate all accesses by the READ and WRITE statements.

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.

2.13.1 The Report Writer Module programs were not run for this validation.

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:

(a) With no optimization invoked the compiler call statement was

```
//COMP  EXEC  PGM=IKFCBL00,REGION=128K,PARM='ADV,SIZE=160K,
//            BUF=16K,TRUNC,LVL=D,APO,LAG,LIB'
```

(b) With optimization invoked the compiler call statement was

```
//COMP  EXEC  PGM=IKFCBL00,REGION=128K,PARM='ADV,SIZE=160K,
//            BUF=16K,TRUNC,LVL=D,APO,LAG,LIB,OPTIMIZE'
```

Options defaulted:

NOPMAP	LINECNT=57	SPACE1
NOCLIST	FLAGE	SEQ
NOSUPMAP	SOURCE	NODMAP
NOXREF	NOSXREF	LOAD
NODECK	NOENDJOB	NOFLOW
NOTERM	NONUM	NOBATCH
NONAME	COMPILE=01	NOSTATE
NORESIDENT	NODYAM	NOSYNTAX
NOSYSMDMP	NOOPTIMIZE	NOTEST
VERB	ZWB	SYST
LVL=D	NOLIST	NOFDECK
NOCDECK	LCOL2	L120
NOCOUNT	ADV	NOPRINT
DUMP	NOVBSUM	NOVBREF
LANGLVL(2)		

The RES instead of the NORES and the DYNAM instead of the NODYNAM options were specified for all Inter-program Communication audit routines.

2. Environment Division implementor-names.

Printer destined files

S-SYSPRINT

The associated DD statement is

```
//SYSPRINT  DD  SYSOUT=A          (for spooled files)
//SYSPRINT  DD  SYSOUT=010        (for nonspooled files)
```

The latter statement was required for those programs using the LINAGE clause.

Tape files

S-TAPE1

An example of an associated DD statement is

```
//TAPE1 DD UNIT=16BPI,DSN=&T1,DCB=DEN=3
```

Sequential Mass-storage files

SEQMAS1

An example of an associated DD statement is

```
//SEQMAS1 DD UNIT=3330,SPACE=(TRK,(50,20)),DSN=&&SQMAS1
```

Relative I-O files

RELMAS1

An example of an associated DD statement is

```
//RELMAS1 DD DSN=RELMAS1,DISP=SHR
```

Index I-O files

INXMAS1

An example of an associated DD statement is

```
//INXMAS1 DD DSN=SHR,AMP=AMORG
```

Sort files (SD)

SORT1

An example of an associated DD statement is

```
//SORT1 DD UNIT=3330,SPACE=(TRK,(50,20)),DSN=&&SORT1
```

Switch names

UPSI-1
UPSI-0

Source Computer names

IBM-370

Object Computer names

IBM-370

3. Optimization. The compiler may or may not have optimization features. If there was an optimization feature available, it was used during the validation process (during a separate execution of the Compiler Validation System) to determine if its use causes the compiler to produce a program which does not give the expected results. If the optimization is invoked through the compiler call statement then it is mentioned in paragraph 1 above. If it is invoked through the introduction of syntax in other than the Data and Procedure Divisions of the source program it is shown below. Optimization which would require modification to the Data and Procedure Divisions is not considered in this report in that it is beyond the scope of the use of standard COBOL and the validation process.

The optimization feature for this compiler is invoked through the compiler call statement. See 1. above. There was no differnce in the execution of the programs when the optimization feature was invoked.

4. Compiler.

IBM OS/VS COBOL RELEASE 2.0 MOD 0

5. Operating system.

MVS 3.7 VS2 with SU8

6. Computer System Reference Manuals.

(a). IBM VS COBOL for OS/VS, Document No. GC 26-3857-0

(b). IBM OS/VS COBOL Compiler and Library Programmers Guide, Document No. SC 28-6483-1.

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 tech-

niques, 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 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 system did not have a card punch or card reader for processing the ASCII-encoded punched cards and therefore, ASCII code represented on punched cards could not be validated. For this validaton the data intended for these devices had to be placed on magnetic tape in order to execute Audit Routines SQ118, SQ119 and SQ120.

2. Both the ASCII-encoded magnetic tape produced by a foreign system and the tape produced as a result of the validaton were unlabeled for this validation. The recording format of the magnetic tape produced during the test was that corresponding to American National Standard Recorded Magnetic Tape for Information Interchange (1600 CPI, PE) X3.22-1973. A separate job was used to place the tape in 800 CPI for testing the tape on another computer system. The ASCII-encoded magnetic tape which resulted was read by the foreign computer system and proved to be correct in both content and format.

APPENDIX A

VALIDATION SUMMARY WORKING DOCUMENT

A-1 This appendix is a working paper produced during the validation and documents the results of the compilation and execution of each of the programs comprising the CCVS. The results contained herein are based on the use of the compiler within the Validation Environment identified in this appendix. This appendix (Validation Summary Working Document) is not part of the official Validation Summary Report (VSR) and is not intended to reflect in any way the compiler's usefulness or degree of conformance to the language specifications.

The reader of this appendix should keep in mind that the same problem area may appear in more than one program, but is considered only as one single discrepancy and as such is reflected only once in the body of the VSR. (The VSR will in turn only reference the first occurrence of the problem in the appendix.)

This appendix is divided into two parts. The first part describes the Validation Environment. The second part of the document is divided into categories of information: compilation and execution results.

The reference document for COBOL is FIPS PUB 21-1 (X3.23-1974).

VALIDATION ENVIRONMENT

COMPILER IDENTIFICATION:	IBM OS/VS COBOL Release 2.0 Mod 0
COMPUTER SYSTEM:	IBM 370/158
OPERATING SYSTEM:	MVS 3.7 VS2 with SU8

ASVP9

The compiler flagged the VALUE clause on a Data Division entry, whose category was alphanumeric edited, with the following messages:

```
IKF2129I-C VALUE CLAUSE LITERAL DOES NOT CONFORM TO PICTURE.
           CHANCE TO ZERO.
IKF2130I-E ITEM CANNOT HAVE VALUE CLAUSE.  CLAUSE IGNORED.
```

The Data Division entry was

```
02  D-DATE PIC XXBXXBXXBXXBXX VALUE SPACE.
```

DEBUG MODULE LEVEL 1

DB101 through DB105

A. Compilation

No errors.

B. Execution

No errors.

DEBUG MODULE LEVEL 2

DB201 through DB204

A. Compilation

No errors.

B. Execution

No errors.

INDEX I-C MODULE LEVEL 1

IX101 through IX107

A. Compilation

No errors.

B. Execution

No errors.

INDEX I-C MODULE LEVEL 2

IX201 through IX204

A. Compilation

No errors.

B. Execution

No errors.

IX205

A. Compilation

No errors.

B. Execution

1. In test INX-TEST-001 a READ IX-FD1 .. INVALID KEY ... statement (random read) is used to access a file in the DYNAMIC ACCESS mode. The INVALID KEY path was taken on each execution of the READ statement.

2. INX-TEST-002 is to create an INDEX I-O file of 200 records under Dynamic Access mode and then read the file both randomly and sequentially. The INVALID KEY path appeared to be taken on the first execution of the WRITE statement.

3. INX-TEST-003.02 and INX-TEST-003.03 failed; however, these two tests depend on the file produced in INX-TEST-002. The failure of these two tests appears to be a result of B.2 above.

The above three errors appear to be caused by the improper execution of the OPEN statement for the data files by the VSAM file facility thus invalidating all accesses to the file.

IX206

A. Compilation

No errors.

B. Execution

This program is similar to IX205 and produced the same type of failures as described in B.1, B.2, and B.3 above.

IX207 through IX208

A. Compilation

No errors.

B. Execution

No errors.

INTER-PROGRAM COMMUNICATION MODULE LEVEL 1

IC101 through IC115

A. Compilation

No errors.

B. Execution

No errors.

IC151 through IC152

A. Compilation

No errors.

B. Execution

No errors.

INTER-PROGRAM COMMUNICATION MODULE LEVEL 2

IC201 through IC208

A. Compilation

No errors.

B. Execution

No errors.

LIBRARY MODULE LEVEL 1

LB101 through LB107

A. Compilation

No errors.

B. Execution

No errors.

LIBRARY MODULE LEVEL 2

LB201 through LB207

A. Compilation

No errors.

B. Execution

No errors.

NUCLEUS MODULE LEVEL 1

NC101 through NC104

A. Compilation

No errors.

B. Execution

No errors.

NC105

A. Compilation

No errors.

B. Execution

1. MOVE-TEST-76 tests the move of a group data item to the alphanumeric edited data item with the picture character-string 0XXXXX0.

   Computed result 0123AB0
   Expected result 123AB

2. In MOVE-TEST-92 a MOVE of a group data item to an alphanumeric edited data item did not produce the expected results.

   Computed result: AB...XYZ +- =$, .()/0012345678
   Expected result: AB...XYZ+- =$, .()/ 0123456789

NC106 through NC112

A. Compilation

No errors.

B. Execution

No errors.

NC113

A. Compilation

The compiler produced the message

IKF1083I-C ILLEGAL CHARACTER. SCAN RESUMED AT NEXT VALID CHARACTER.

on the following COBOL code

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

B. Execution

No errors.

NC114 through NC120

A. Compilation

No errors.

B. Execution

No errors.

NC151 through NC156

A. Compilation

No errors.

B. Execution

No errors.

NC157

A. Compilation

The compiler flagged a SYNCHRONIZED clause in an 01 level Data Division entry, which was an elementary data item, as an extension to all levels of Federal Standard COBOL.

B. Execution

No errors.

NC158 through NC162

A. Compilation

No errors.

B. Execution

No errors.

NC163

A. Compilation

The compiler flagged a SYNCHRONIZED clause in a 01 level Data Division entry, which was an elementary data item, as an extension to all levels of Federal Standard COBOL.

B. Execution

1. In MOVE-TEST-76 a MOVE statement which moved a group data item to an alphanumeric edited data item did not produce the expected results. The results produced by this test were the same as NC105 B.1 above.

2. In MOVE-TEST-92 a MOVE statement which moved a group cata item to an edited data item did not produce the expected results. The results produced by this test were the same as NC105 B.2 above.

NC164 through NC165

A. Compilation

No errors.

B. Execution

No errors.

NUCLEUS MODULE LEVEL 2

NC201

A. Compilation

The compiler incorrectly flagged the statement PERFORM ... VARYING ... FROM -5.5 BY ... UNTIL with the following message:

IKF80021-W USE OF NEGATIVE INDEXES IN PERFORM STATEMENT IS AN EXTENSION TO ALL LEVELS.

There were no indices associated with the statement.

B. Execution

No errors.

NC202 through NC204

A. Compilation

No errors.

B. Execution

No errors.

NC205

A. Compilation

1. The compiler issued three E-level and one C-level messages on statements which used line continuation for numeric literals. The statement constructs which caused the messages to be produced were:

```
035900  CONTIN-TEST-1.
036000       MOVE     4
036100-               5
036200-               6
036300-               7
036400-               8 TO CONT-B.
   .
   .
   .
037900  CONTIN-TEST-2.
038000       MOVE     -
038100-               9
038200-               9
038300-               9
038400-               .
038500-               7
038600-               7
038700-               7  TO CONT-B.
```

2. The compiler did not accept the following statement construct in CONT-TEST-4.

```
      IF  CONT EQUA
-                 L TO ZERO
-                       S AN
-                           D GREATER
-                                        ZERO AND CONT-B
          EQUAL TO CONT-C OR (((((((0
                .
                .
                .

                (several blank source lines)

                .
                .
                .
```

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

3. The compiler did not accept the following statement ccnstruct in CONTIN-TEST-5.

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

B. Execution

Because of A. above, 4 tests were deleted.

NC206 through NC212

A. Compilation

No errors.

B. Execution

No errors.

NC213

A. Compilation

The compiler flagged Data Division entries which included a VALUE clause with numeric-edited picture character-strings as an extension to Federal Standard COBOL. The messages produced were

IKF8002I-W NUMERIC LITERAL IN VALUE CLAUSE FOR NUMERIC

EDITED ITEMS IS AN EXTENSION TO ALL LEVELS.

B. Execution

A numeric edited data item whose picture character-string was $.** and was a receiving operand in a SUBTRACT statement did not give the expected results. The initial value of the data item was zero and the value resulting from execution of the statement was 9.99. Because the ON SIZE ERROR phrase was specified, the contents of the data-item was expected to be unaltered and contain its initial value of zero.

NC214 through NC218

A. Compilation

No errors.

B. Execution

No errors.

RELATIVE I-O MODULE LEVEL 1

RL101 through RL109

A. Compilation

No errors.

B. Execution

No errors.

RELATIVE I-O MODULE LEVEL 2

RL201 through RL205

A. Compilation

No errors.

B. Execution

No errors.

REPORT WRITER MODULE LEVEL 1

RW101 through RW104

The Report Writer Module programs were not run for this validation.

SEGMENTATION MODULE LEVEL 1

SG101 through SG106

A. Compilation

No errors.

B. Execution

No errors.

SEGMENTATION MODULE LEVEL 2

SG201 through SG204

A. Compilation

No errors.

B. Execution

No errors.

SEQUENTIAL ACCESS MODULE LEVEL 1

SQ101 through SQ117

A. Compilation

No errors.

B. Execution

No errors.

SQ118 through SQ120

A. Compilation

No errors.

B. Execution

The system does not support card punch or card reader devices for reading ASCII encoded punched cards. For these programs the files intended for use with the card reader and punch were assigned to magnetic tape instead.

SQ121

A. Compilation

No errors.

B. Execution

No errors.

SQ151 through SQ153

A. Compilation

No errors.

B. Execution

No errors.

SEQUENTIAL ACCESS MODULE LEVEL 2

SQ201 through SQ218

A. Compilation

No errors.

B. Execution

No errors.

SORT MODULE LEVEL 1

ST101 through ST117

A. Compilation

No errors.

B. Execution

No errors.

SORT MODULE LEVEL 2

ST201 through ST215

A. Compilation

No errors.

B. Execution

No errors.

BIBLIOGRAPHIC DATA SHEET	1. Report No. CCVS74-VSR225	2.	3. Recipient's Accession No.
4. Title and Subtitle Validation Summary Report # CCVS74-VSR225 IBM VS COBOL Release 2 Mod 0			5. Report Date JUN 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 Washington, D. C. 20376			13. Type of Report & Period Covered 14.

15. Supplementary Notes

16. Abstracts

This Validation Summary Report (VSR) for the IBM 370 COBOL Compiler Version 2 Mod 0 (MVS Version 3.7 VS2) 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 High level of FIPS PUB 21-1. The VSR is made up of several sections showing the discrepancies found. These include an overview of the validation which lists all categories of discrepancies by level/module within X3.23-1974, 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.	UNCLASSIFIED	78
	20. Security Class (This Page) UNCLASSIFIED	22. Price

FORM NTIS-35 (REV. 3-72) THIS FORM MAY BE REPRODUCED USCOMM-DC 14952-P72

TABLE HANDLING MODULE LEVEL 1

TH101 through TH111

A. Compilation

No errors.

B. Execution

No errors.

TH151 through TH152

A. Compilation

No errors.

B. Execution

No errors.

TABLE HANDLING MODULE LEVEL 2

TH201 through TH220

A. Compilation

No errors.

B. Execution

No errors.