

PB93-163178



CCVS 85

USER GUIDE

Version 4.2

Copyright 1993
The National Computing Centre Limited
All rights reserved.

Prepared by: The National Computing Centre Limited
Oxford Road
Manchester
M1 7ED
United Kingdom

01 March 1993

The National Computing Centre acknowledge the valuable assistance of the National Institute of Standards and Technology (NIST) in the production of the C CVS and its associated documentation.

REPRODUCED BY

U.S. DEPARTMENT OF COMMERCE
NATIONAL TECHNICAL INFORMATION SERVICE
SPRINGFIELD, VA. 22161

CONTENTS

	Page
1 INTRODUCTION	1
1.1 BACKGROUND	1
1.2 PURPOSE AND NATURE OF COBOL COMPILER VALIDATION	2
1.3 DIFFERENCES IN USING THE C CVS FOR FIPS AND ISO/ANSI TESTING	2
1.4 X/OPEN XPG4 COBOL TESTING	2
2 OVERVIEW OF THE COBOL 85 COMPILER VALIDATION SYSTEM	3
2.1 NAMING CONVENTIONS FOR THE AUDIT ROUTINES	3
2.2 CHOICE OF AUDIT ROUTINES	4
2.3 REQUIRED RESOURCES	5
3 RUNNING THE C CVS	5
3.1 LOADING THE C CVS	5
3.2 EXTRACTING THE EXECUTIVE ROUTINE	6
3.3 EDITING AND COMPIILING THE EXECUTIVE ROUTINE	6
3.4 PREPARING THE EXECUTIVE INPUT DATA	6
3.4.1 ** COMMANDS	7
3.4.2 OPTION SWITCHES	9
3.4.3 X-CARDS	10
3.4.4 FINAL ** COMMANDS AND PROGRAM FIXES	11
3.5 RUNNING THE EXECUTIVE ROUTINE	11
3.5.1 PRINT-FILE	11
3.5.2 NEW POPULATION FILE	12
3.5.3 SOURCE-COBOL-PROGRAMS FILE	12
3.6 RUNNING THE AUDIT ROUTINES	12
3.7 FLAGGING TESTS	13

3.7.1	USE OF THE FLAGGING TESTS	13
3.7.2	METHOD OF OPERATION	14
3.7.3	TEST OVERVIEW	15
3.7.3.1	INTERMEDIATE	15
3.7.3.2	HIGH	15
3.7.3.3	OBSOLETE	15
4	LIST OF AUDIT ROUTINES	16
4.1	COMMUNICATIONS MODULE	CM-1
	CM101M	CM-1
	CM102M	CM-1
	CM103M	CM-1
	CM104M	CM-2
	CM105M	CM-2
	CM201M	CM-2
	CM202M	CM-2
4.2	DEBUG MODULE	DB-1
	DB101A	DB-1
	DB102A	DB-1
	DB103M	DB-2
	DB104A	DB-2
	DB105A	DB-3
	DB201A	DB-4
	DB202A	DB-4
	DB203A	DB-4
	DB204A	DB-5
	DB205A	DB-5
4.3	INTER-PROGRAM COMMUNICATIONS MODULE	IC-1
	IC101A	IC-1
	IC102A	IC-1
	IC103A	IC-1
	IC104A	IC-2
	IC105A	IC-2
	IC106A	IC-2
	IC107A	IC-2
	IC108A	IC-3
	IC109A	IC-3
	IC110A	IC-3

IC111A	IC-3
IC112A	IC-3
IC113A	IC-4
IC114A	IC-4
IC115A	IC-4
IC116M	IC-4
IC117M	IC-5
IC118M	IC-5
IC201A	IC-5
IC202A	IC-5
IC203A	IC-5
IC204A	IC-6
IC205A	IC-6
IC206A	IC-6
IC207A	IC-6
IC208A	IC-6
IC209A	IC-7
IC210A	IC-7
IC211A	IC-7
IC212A	IC-7
IC213A	IC-7
IC214A	IC-7
IC215A	IC-8
IC216A	IC-8
IC217A	IC-8
IC222A	IC-8
IC223A	IC-9
IC224A	IC-9
IC225A	IC-9
IC226A	IC-9
IC227A	IC-9
IC228A	IC-10
IC233A	IC-10
IC234A	IC-10
IC235A	IC-10
IC237A	IC-10
OBIC1A	IC-11
OBIC2A	IC-11
OBIC3A	IC-11

4.4	INTRINSIC FUNCTION MODULE	IF-1
	IF101A	IF-1
	IF102A	IF-1
	IF103A	IF-2
	IF104A	IF-2
	IF105A	IF-2
	IF106A	IF-2
	IF107A	IF-3
	IF108A	IF-3
	IF109A	IF-3
	IF110A	IF-4
	IF111A	IF-4
	IF112A	IF-4
	IF113A	IF-5
	IF114A	IF-5
	IF115A	IF-5
	IF116A	IF-6
	IF117A	IF-6
	IF118A	IF-6
	IF119A	IF-7
	IF120A	IF-7
	IF121A	IF-7
	IF122A	IF-8
	IF123A	IF-8
	IF124A	IF-8
	IF125A	IF-9
	IF126A	IF-9
	IF127A	IF-9
	IF128A	IF-10
	IF129A	IF-10
	IF130A	IF-10
	IF131A	IF-11
	IF132A	IF-11
	IF133A	IF-11
	IF134A	IF-12
	IF135A	IF-12
	IF136A	IF-12
	IF137A	IF-13
	IF138A	IF-13
	IF139A	IF-13
	IF140A	IF-14
	IF141A	IF-14
	IF142A	IF-14

4.5	INDEXED I-O MODULE	IX-1
	IX101A	IX-1
	IX102A	IX-1
	IX103A	IX-1
	IX104A	IX-1
	IX105A	IX-2
	IX106A	IX-2
	IX107A	IX-2
	IX108A	IX-2
	IX109A	IX-2
	IX110A	IX-3
	IX111A	IX-3
	IX112A	IX-3
	IX113A	IX-3
	IX114A	IX-3
	IX115A	IX-4
	IX116A	IX-4
	IX117A	IX-4
	IX118A	IX-4
	IX119A	IX-4
	IX120A	IX-5
	IX121A	IX-5
	IX201A	IX-5
	IX202A	IX-5
	IX203A	IX-5
	IX204A	IX-6
	IX205A	IX-6
	IX206A	IX-6
	IX207A	IX-6
	IX208A	IX-7
	IX209A	IX-7
	IX210A	IX-7
	IX211A	IX-8
	IX212A	IX-8
	IX213A	IX-8
	IX214A	IX-9
	IX215A	IX-9
	IX216A	IX-9
	IX217A	IX-9
	IX218A	IX-9

4.6	NUCLEUS MODULE	NC-1
	NC101A	NC-1
	NC102A	NC-1
	NC103A	NC-1
	NC104A	NC-1
	NC105A	NC-2
	NC106A	NC-2
	NC107A	NC-2
	NC108M	NC-2
	NC109M	NC-3
	NC110M	NC-3
	NC111A	NC-3
	NC112A	NC-4
	NC113M	NC-4
	NC114M	NC-4
	NC115A	NC-4
	NC116A	NC-4
	NC117A	NC-4
	NC118A	NC-5
	NC119A	NC-5
	NC120A	NC-5
	NC121M	NC-5
	NC122A	NC-5
	NC123A	NC-6
	NC124A	NC-6
	NC125A	NC-6
	NC126A	NC-6
	NC127A	NC-6
	NC131A	NC-7
	NC132A	NC-7
	NC133A	NC-7
	NC134A	NC-7
	NC135A	NC-7
	NC136A	NC-8
	NC137A	NC-8
	NC138A	NC-8
	NC139A	NC-8
	NC140A	NC-8
	NC141A	NC-9
	NC170A	NC-9
	NC171A	NC-9
	NC172A	NC-9
	NC173A	NC-9
	NC174A	NC-10
	NC175A	NC-10
	NC176A	NC-10
	NC177A	NC-10
	OBNC1M	NC-10
	NC201A	NC-10
	NC202A	NC-11
	NC203A	NC-11
	NC204M	NC-11



NC205A	NC-12
NC206A	NC-12
NC207A	NC-12
NC208A	NC-12
NC209A	NC-12
NC210A	NC-12
NC211A	NC-13
NC214M	NC-13
NC215A	NC-13
NC216A	NC-13
NC217A	NC-13
NC218A	NC-14
NC219A	NC-14
NC220M	NC-14
NC221A	NC-14
NC222A	NC-14
NC223A	NC-15
NC224A	NC-15
NC225A	NC-15
NC231A	NC-15
NC232A	NC-15
NC233A	NC-16
NC234A	NC-16
NC235A	NC-16
NC236A	NC-16
NC237A	NC-17
NC238A	NC-17
NC239A	NC-17
NC240A	NC-17
NC241A	NC-17
NC242A	NC-18
NC243A	NC-18
NC244A	NC-18
NC245A	NC-18
NC246A	NC-18
NC247A	NC-19
NC248A	NC-19
NC250A	NC-19
NC251A	NC-19
NC252A	NC-19
NC253A	NC-20
NC254A	NC-20
OBNC2M	NC-20

4.7	RELATIVE I-O MODULE	RL-1
	RL101A	RL-1
	RL102A	RL-1
	RL103A	RL-1
	RL104A	RL-2
	RL105A	RL-2
	RL106A	RL-2
	RL107A	RL-2
	RL108A	RL-3
	RL109A	RL-3
	RL110A	RL-3
	RL111A	RL-4
	RL112A	RL-4
	RL113A	RL-4
	RL114A	RL-4
	RL115A	RL-5
	RL116A	RL-5
	RL117A	RL-5
	RL118A	RL-6
	RL119A	RL-6
	RL201A	RL-6
	RL202A	RL-7
	RL203A	RL-7
	RL204A	RL-7
	RL205A	RL-7
	RL206A	RL-8
	RL207A	RL-8
	RL208A	RL-8
	RL209A	RL-8
	RL210A	RL-8
	RL211A	RL-9
	RL212A	RL-9
	RL213A	RL-9
4.8	REPORT WRITER MODULE	RW-1
	RW101A	RW-1
	RW102A	RW-1
	RW103A	RW-2
	RW104A	RW-2

4.9	SEGMENTATION MODULE	SG-1
	SG101A	SG-1
	SG102A	SG-1
	SG103A	SG-1
	SG104A	SG-2
	SG105A	SG-2
	SG106A	SG-2
	SG201A	SG-2
	SG202A	SG-2
	SG203A	SG-3
	SG204A	SG-3
4.10	SOURCE TEXT MANIPULATION MODULE	SM-1
	SM101A	SM-1
	SM102A	SM-1
	SM103A	SM-1
	SM104A	SM-2
	SM105A	SM-2
	SM106A	SM-2
	SM107A	SM-2
	SM201A	SM-2
	SM202A	SM-3
	SM203A	SM-3
	SM204A	SM-3
	SM205A	SM-3
	SM206A	SM-3
	SM207A	SM-4
	SM208A	SM-4



4.11	SEQUENTIAL I-O MODULE	SQ-1
	SQ101M	SQ-1
	SQ102A	SQ-1
	SQ103A	SQ-1
	SQ104A	SQ-2
	SQ105A	SQ-2
	SQ106A	SQ-2
	SQ107A	SQ-2
	SQ108A	SQ-2
	SQ109M	SQ-3
	SQ110M	SQ-3
	SQ111A	SQ-4
	SQ112A	SQ-4
	SQ113A	SQ-4
	SQ114A	SQ-4
	SQ115A	SQ-5
	SQ116A	SQ-5
	SQ117A	SQ-5
	SQ121A	SQ-5
	SQ122A	SQ-5
	SQ123A	SQ-6
	SQ124A	SQ-6
	SQ125A	SQ-6
	SQ126A	SQ-6
	SQ127A	SQ-6
	SQ128A	SQ-7
	SQ129A	SQ-7
	SQ130A	SQ-7
	SQ131A	SQ-7
	SQ132A	SQ-7
	SQ133A	SQ-8
	SQ134A	SQ-8
	SQ135A	SQ-8
	SQ136A	SQ-8
	SQ137A	SQ-9
	SQ138A	SQ-9
	SQ139A	SQ-9
	SQ140A	SQ-9
	SQ141A	SQ-9
	SQ142A	SQ-9
	SQ143A	SQ-10
	SQ144A	SQ-10
	SQ146A	SQ-10
	SQ147A	SQ-10
	SQ148A	SQ-10
	SQ149A	SQ-11
	SQ150A	SQ-11
	SQ151A	SQ-11
	SQ152A	SQ-11
	SQ153A	SQ-11
	SQ154A	SQ-12
	SQ155A	SQ-12

SQ156A	SQ-12
OBSQ1A	SQ-12
SQ201M	SQ-13
SQ202A	SQ-13
SQ203A	SQ-13
SQ204A	SQ-14
SQ205A	SQ-14
SQ206A	SQ-14
SQ207M	SQ-14
SQ208M	SQ-14
SQ209M	SQ-15
SQ210M	SQ-15
SQ211A	SQ-15
SQ212A	SQ-15
SQ213A	SQ-16
SQ214A	SQ-16
SQ215A	SQ-16
SQ216A	SQ-16
SQ217A	SQ-17
SQ218A	SQ-17
SQ219A	SQ-17
SQ220A	SQ-17
SQ221A	SQ-18
SQ222A	SQ-18
SQ223A	SQ-18
SQ224A	SQ-18
SQ225A	SQ-18
SQ226A	SQ-19
SQ227A	SQ-19
SQ228A	SQ-19
SQ229A	SQ-19
SQ230A	SQ-20
OBSQ3A	SQ-20
OBSQ4A	SQ-20
OBSQ5A	SQ-21



4.12	SORT-MERGE MODULE	ST-1
	ST101A	ST-1
	ST102A	ST-2
	ST103A	ST-2
	ST104A	ST-2
	ST105A	ST-2
	ST106A	ST-3
	ST107A	ST-3
	ST108A	ST-3
	ST109A	ST-3
	ST110A	ST-4
	ST111A	ST-4
	ST112M	ST-4
	ST113M	ST-5
	ST114M	ST-5
	ST115A	ST-5
	ST116A	ST-5
	ST117A	ST-6
	ST118A	ST-6
	ST119A	ST-7
	ST120A	ST-7
	ST121A	ST-7
	ST122A	ST-7
	ST123A	ST-7
	ST124A	ST-8
	ST125A	ST-8
	ST126A	ST-8
	ST127A	ST-8
	ST131A	ST-9
	ST132A	ST-10
	ST133A	ST-11
	ST134A	ST-11
	ST135A	ST-11
	ST136A	ST-12
	ST137A	ST-12
	ST139A	ST-12
	ST140A	ST-13
	ST144A	ST-13
	ST146A	ST-14
	ST147A	ST-14

4.13	THE FLAGGING TESTS	Flagging-1
	INTERMEDIATE	Flagging-1
	IX301M	Flagging-1
	RL301M	Flagging-1
	SM301M	Flagging-2
	ST301M	Flagging-2
	HIGH	Flagging-2
	IC401M	Flagging-2
	IF401M	Flagging-3
	IF402M	Flagging-3
	IF403M	Flagging-4
	IX401M	Flagging-4
	NC401M	Flagging-5
	RL401M	Flagging-6
	SM401M	Flagging-6
	SQ401M	Flagging-7
	OBSOLETE	Flagging-7
	IX302M	Flagging-7
	NC302M	Flagging-7
	NC303M	Flagging-8
	RL302M	Flagging-8
	SQ302M	Flagging-8
	SQ303M	Flagging-8
	OPTIONAL	Flagging-9
	CM303M	Flagging-9
	CM401M	Flagging-9
	DB301M	Flagging-9
	DB302M	Flagging-9
	DB303M	Flagging-10
	DB304M	Flagging-10
	DB305M	Flagging-10
	RW301M	Flagging-10
	RW302M	Flagging-11
	SG302M	Flagging-11
	SG303M	Flagging-11
	SG401M	Flagging-11

APPENDIX A	- SUBSETS OF FEDERAL STANDARD COBOL Flagging programs for each compiler option switch.	A-1 A-2
APPENDIX B	- TABLE OF PROGRAM DEPENDENCIES	B-1
APPENDIX C	- OPTION SWITCHES	C-1
APPENDIX D	- X-CARDS	D-1
APPENDIX E	- SAMPLE EXECUTIVE INPUT DATA	E-1
APPENDIX F	- SAMPLE PROGRAM FOR EXTRACTING AUDIT ROUTINES FROM THE POPULATION FILE	F-1
APPENDIX G	- CCVS85 ERROR PROCEDURE	G-1
APPENDIX H	- SAMPLE AMENDMENT SHEET	H-1
APPENDIX I	- TEMPORARY PROGRAM FIXES	I-1

APPENDIX I

TEMPORARY PROGRAM FIXES

1. INTRODUCTION

This is a supplement to the 1985 CCVS User Guide Version 4.2, containing the updates for correcting the errors in that version of the CCVS85. The updates, referred to as Temporary Program Fixes (TPFs) are presented in the format required for use by the EXEC85 program, and will be applied in conducting a formal validation. The TPFs are presented in this document in the same order as the programs appear on the CCVS85 Population File, to facilitate preparing the updates for use.

1 INTRODUCTION

1.1 BACKGROUND

Compiler validation was introduced in the United States of America in support of the General Services Administration (GSA) Federal Property Management Regulation 101-36.1305-1 "FIPS PUB 21, Federal Standard COBOL". All COBOL compilers brought into the Federal inventory must be validated, and as the US Government is the largest purchaser of data processing equipment in the world this has had a major impact. Many suppliers of compilers now see validation as an essential means of quality assurance for their product. The 1985 COBOL Compiler Validation System (CCVS) is based on the 1974 CCVS, with enhancements and additional programs designed to test the new functions in COBOL 85. It tests for conformity to the American National Standard (ANSI Document Reference X3.23-1985) and the standard of the International Organisation for Standardisation (ISO Document Reference ISO-1989-1985) as well as FIPS PUB 21. CCVS85 V3.1 onwards tests for the X3.23A-1989 Intrinsic Function Module of which the specifications for the test programs were developed by NIST and the programs were produced by NCC.

The tests for the enhancements to 1985 COBOL were specified by a project team which was funded by the Commission for European Communities and was responsible for technical issues to:

*The Federal Software Management Support Center
Office of Software Development & Information Technology
Two Skyline Place
Suite 1100
5203 Leesburg Pike
Falls Church
VA 22041
U.S.A.

The project team members were:
BIADI (Bureau Inter Administration de Documentation Informatique)
21 Rue Bara
92132 Issy
France

GMD (Gesellschaft fur Mathematik und Datenverarbeitung MBH)
Postfach 1240
Schloss Birlinghoven
D-5205 St Augustin 1
Germany FR

NCC (The National Computing Centre Limited)
Oxford Road
Manchester
M1 7ED
United Kingdom

Version 1.0 of the 1985 CCVS was released in January 1986.



2. CORRECTIONS FOR CCVS85 V4.2

2.1 Program Name: IX110A.

a. Description of Problem

Line 012250 states that STATUS-TEST-10 is PIC P. It should be PIC 9.

b. Update

```
*START,IX110A  
012250 01 STATUS-TEST-10 PIC 9 VALUE ZERO.
```

2.2 Program Name: SM104A.

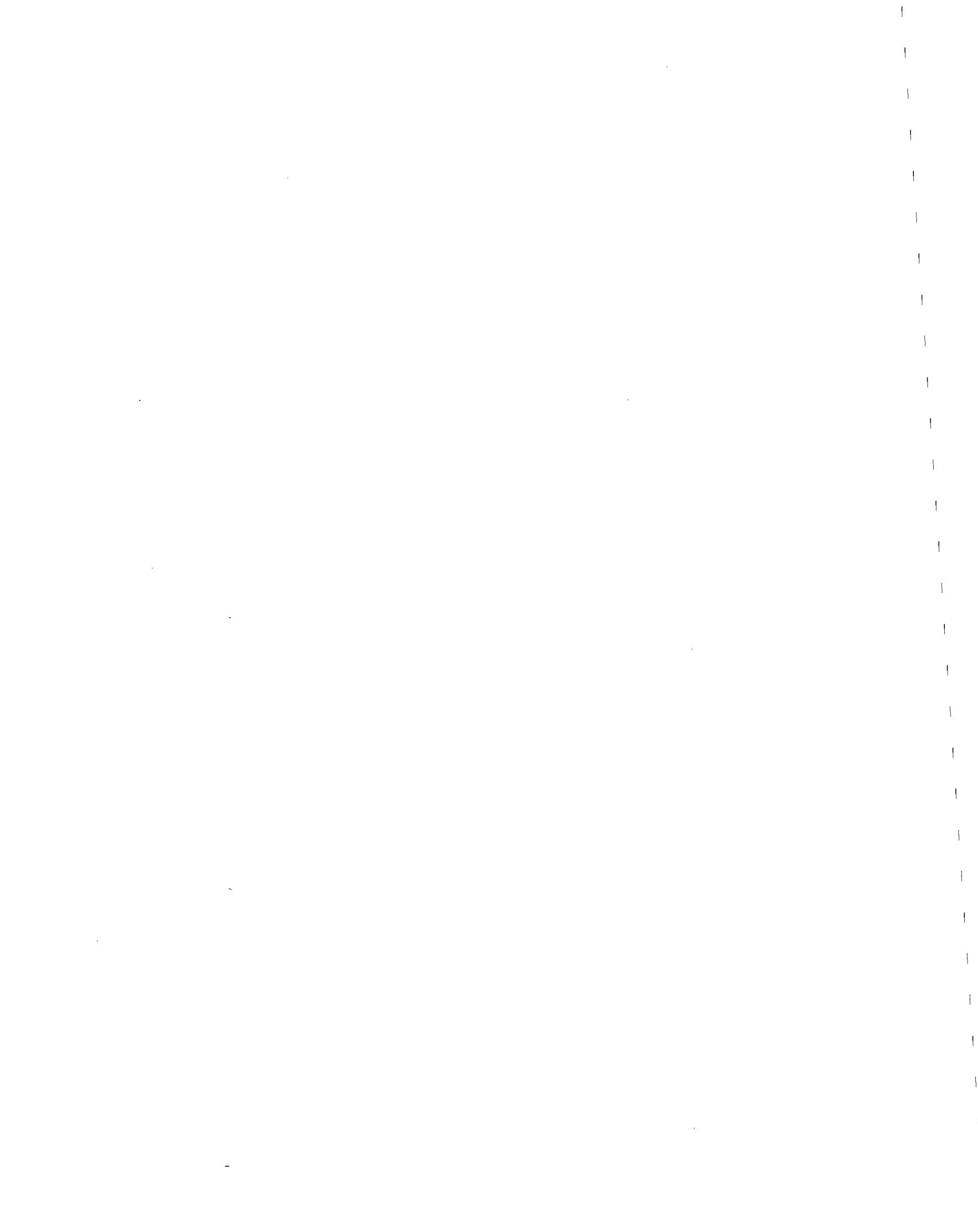
Description of Problem

When the program-id is part of the replacement by the X-card during extraction (See CCVS85 User Guide Version 3.0, Page 11), line 003800 in the extracted program requires modification to correct the expected filename to that created by SM103A. The entry on line 003800 must correspond to the entry for the X-01 card with a manual substitution of "K3FCA" for the program-id.

2.3 Program Name: SM204A.

Description of Problem

When the program-id is part of the replacement by the X-card during extraction (See CCVS85 User Guide Version 3.0, Page 11), line 003800 in the extracted program requires modification to correct the expected filename to that created by SM203A. The entry on line 003800 must correspond to the entry for the X-01 card with a manual substitution of "K3FCB" for the program-id.



*Following a reorganisation of the General Services Administration in the United States, the former role of the FSTC was passed over to the National Institute of Standards and Technology (NIST) formerly known as the National Bureau of Standards.

1.2 PURPOSE AND NATURE OF COBOL COMPILER VALIDATION

The validation of a compiler determines the degree to which it conforms to the technical specifications on which it was based. The use of compilers that have attained a high degree of conformity to their respective language standards (technical specifications) enhances source program interchangeability within all DP installations which use that particular programming language.

The results of running a Compiler Validation System do not suggest the degree to which the compiler is usable, (ie capable of data processing applications), but the degree to which individual language elements are usable. This gives an indication of conversion areas which must be considered in order to implement a source program from another computer system supporting a compiler based on the same set of language specifications.

The COBOL Compiler Validation System (CCVS) can thus be used to test a COBOL compiler's adherence to the standard language syntax and, where unambiguous, language semantics of the technical specifications upon which the compiler is based. The latter, of course, is a more difficult area because of the lack of appropriate mechanisms for precise semantic specifications. The Validation System does not evaluate the implementation of a compiler or its quantitative performance characteristics.

1.3 DIFFERENCES IN USING THE CCVS FOR FIPS AND ISO/ANSI TESTING

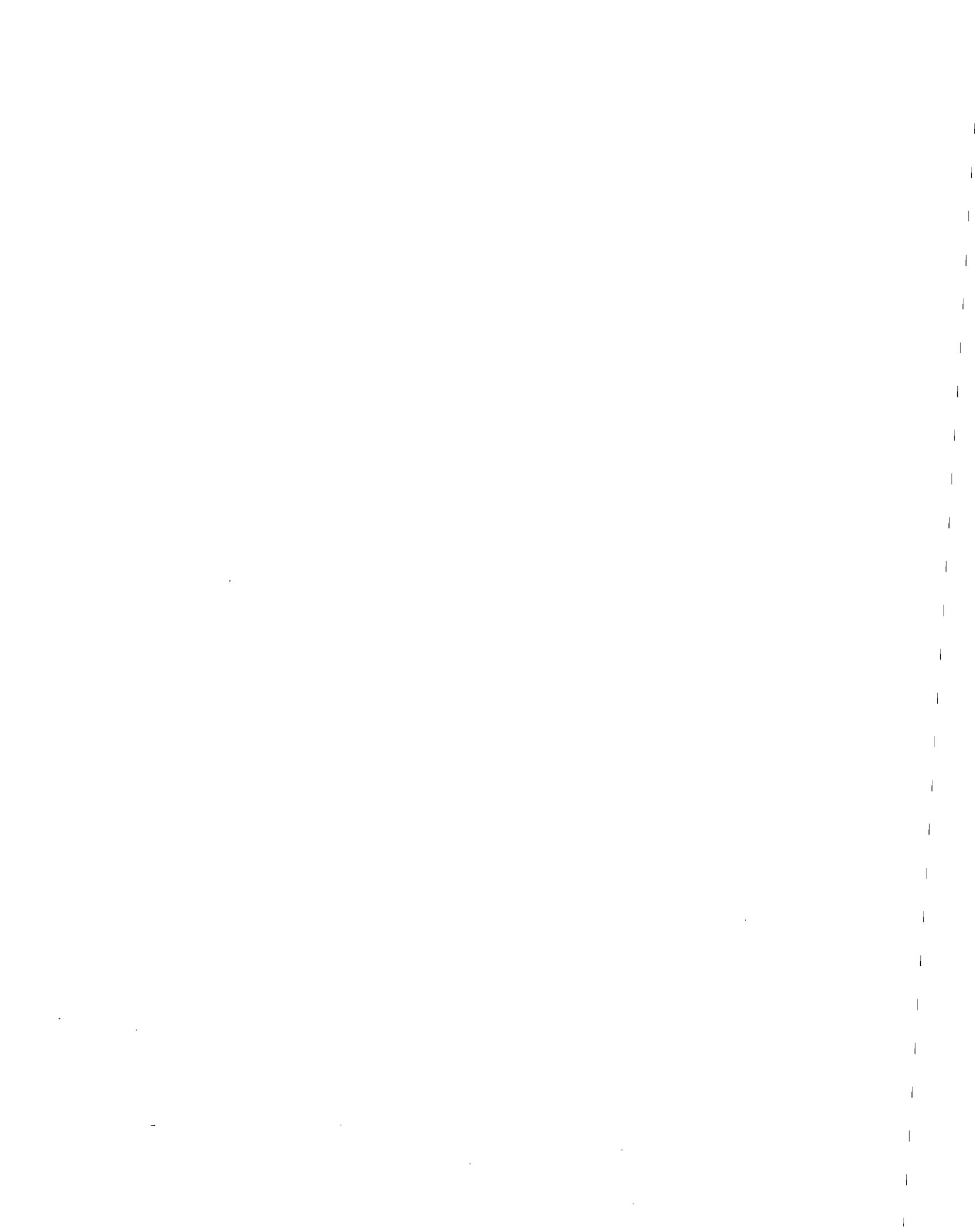
The CCVS may be used for both FIPS and ISO/ANSI testing.

The only differences in using the suite for ISO/ANSI testing are:

- a) The Intrinsic Function module is optional.
- b) Only Flagging Tests that cover obsolete features are required. For guidance on which tests these are, see section 4.13 (Flagging) and Appendix A.

1.4 X/OPEN XPG4 COBOL TESTING

The CCVS85 includes a separate module of X/Open extension tests. These tests when run with the main CCVS85 tests form the basis of the XPG4 COBOL Language Testing Requirements (These are contained in the document "X/Open CAE Specification for COBOL"). The tests are contained within the audit routines whose names start with the character 'X'. THESE TESTS ARE NOT REQUIRED FOR FIPS TESTING OR ISO/ANSI TESTING. THEY ARE ONLY REQUIRED FOR X/OPEN TESTING. Details of the X/Open tests are given in the CCVS85 X/OPEN EXTENSION USER GUIDE. There is no further reference to these tests in this user guide.



The following is a composite of the source code represented by the TPFs contained in this document.
Card Columns:

```
0.....1.....2.....3.....4.....5.....6..  
*BEGIN-UPDATE  
*START,IX110A  
012250 01 STATUS-TEST-10 PIC 9 VALUE ZERO.  
*END-UPDATE
```


2 OVERVIEW OF THE COBOL 85 COMPILER VALIDATION SYSTEM

The programs comprising the CCVS are a series of audit routines maintained in a system independent source program library on magnetic media. The programs are designed in such a way that all implementation specific names are presented in encoded mnemonics, made up of a series of Xs followed by a three digit integer. The executive routine provided with the CCVS replaces these mnemonics with information provided by the user at the time the source programs are extracted from the program library, resulting in syntactically correct programs for a given compiler.

Once the audit routines have been processed by the executive they may be compiled. The rejection of any of the source code by a compiler is a deviation from the standard. When a program has compiled, the object code produced is executed and the output checked for 'pass' or 'fail' of the features tested. In most instances this is done automatically by the program but some programs require visual checking of the output.

Each program contains entries and statements that thoroughly test features of a portion of the language. Generally, each program is restricted to testing the features of a specific level of one functional module of COBOL, so the user can select the level at which a particular compiler is to be tested. Some basic features of COBOL, such as MOVE, GO TO, PICTURE, IF and SELECT are used as supporting code in all of the audit routines, so if any of these function incorrectly the compiler may be dismissed immediately.

For the most part, the necessary data for the programs are created either internally or by the previous program within that module and level. The two Nucleus programs testing the ACCEPT verb use external data, located on the source program library immediately following each program.

There are certain language elements defined in the COBOL Standard which do not have to be supported for an implementation to conform to the Standard - the testing of these elements is accomplished through an "option code".

2.1 NAMING CONVENTIONS FOR THE AUDIT ROUTINES

COBOL 85 comprises twelve functional modules, most of which are divided into level 1 and level 2. Each audit routine in the CCVS is named to associate it with the module and level of ANS COBOL tested therein. Each module and level is represented by several programs. An audit routine name consists of six characters, the first two being alphabetic and identifying the functional module being tested:

NC - Nucleus
SQ - Sequential I-O
RL - Relative I-O
IX - Indexed I-O
IC - Inter-Program Communication
ST - Sort-Merge
SM - Source Text Manipulation
RW - Report Writer
CM - Communication
DB - Debug
SG - Segmentation
IF - Intrinsic Function

The third character, which is numeric, represents the level of the functional module, and the fourth and fifth characters, also numeric, provide each audit routine with a unique name and also indicate the sequence in which programs need to be run, eg DB204A is the fourth in a series of several programs which test Level 2 of the Debug module. The sixth character further indicates the sequence in which programs need to be run and also indicates the method of checking the results. For example, given programs SQ210M, SQ211A and SQ212A, SQ210M would be run first then SQ211A and then SQ212A. The "A" in the sixth character denotes that the test results are Automatically checked by the program, "M" denotes that some Manual checking is required.

The deviation from the above rules apply in the case of flagging test programs in which the third character is either a '3' or a '4'. Test Programs with the third character as '4' are above intermediate in a high implementation level, above 1COM, above 1DEB, and above 1SEG.

In the ANS-1985 COBOL Standard there are some features that have been described as obsolete and these are to be dropped from the next COBOL Standard. However, these must still be tested as they are currently supported features, but the programs that specifically test them have been named as OB_{xx}n where xx is the module abbreviation and n is the numeric identifier for that particular program.

2.2 CHOICE OF AUDIT ROUTINES

It is possible to test any combination of levels and modules of Federal Standard COBOL. The routines can be selected either by individual module, by a level within a module, or by individual program.

Appendix A shows the levels of each module required for testing to a particular COBOL subset. Note that, according to ANS specifications, the higher level of a given module technically consists of the lower level plus some unique higher level features. Therefore, in order to completely test any higher level module, it is necessary to select all lower level tests for the module as well. For example, in order to test Level 1 of Nucleus and Level 2 of Sequential I-O all programs beginning with 'NC1', 'SQ1' and 'SQ2' should be selected. The Intrinsic Function module is part of the required High Subset in FIPS COBOL. In ISO/ANSI COBOL this module is optional.

The exceptions to the third character rule are programs ST122A to ST124A, ST135A, and ST137A to ST147A as these use Level 2 features from other modules in order to thoroughly test features from the Sort-Merge Module, and therefore, although the Sort-Merge Module is Level 1, these particular programs need only be run when testing to High level.

The Report Writer, Communication, Debug and Segmentation modules are optional (also Intrinsic Function for ISO/ANSI), but if required to be tested, either level may be used regardless of the COBOL subset which is being tested.

2.3 REQUIRED RESOURCES

The following should exist on a computer system in order to validate a COBOL 85 compiler with the CCVS:

1. Sufficient storage to compile and execute programs of approximately 3300 lines of COBOL source code.
2. An input device for the executive input data.
3. At least one magnetic tape drive for the CCVS library (although this may, by special arrangement, be supplied on disk if a magnetic tape drive cannot be made available).
4. A line printer (at least 120 characters line width).
5. A mass storage device or at least two magnetic tape drives must be made available for processing work files produced by some of the tests.

3 RUNNING THE CCVS

It is advisable to read the whole of this section before attempting to run the CCVS, as some procedures offer more than one way of doing things and the method selected will affect the way in which future steps can be carried out.

3.1 LOADING THE CCVS

The first procedure is to load the CCVS onto the system on which the compiler that is to be validated is held. The programs comprising the CCVS are held in the POPULATION-FILE on magnetic tape with the following characteristics:

Tracks	-	9
Parity	-	odd
Code	-	ASCII
Density	-	1600 BPI, phase encoded
Record	-	thirty 80 character records per block
Blocking	-	deblocking is done by the program
Label	-	unlabelled (omitted)
File Mark	-	two tapes are used - one for the original CCVS and the other for the X/Open extensions. The contents of the tapes consist of one file, and the end-of-file follows the last record on the tape.

The CCVS can also be provided on cartridge media and in Tape Archive (TAR) format.

Under certain circumstances this file may have been supplied on some device other than the tape.

If required the POPULATION-FILE may be downloaded onto disk - it will take approximately 28 Megabytes of storage.

Note that the Executive Routine (see 3.3) assumes that the POPULATION-FILE will be in 80-character records.

3.2 EXTRACTING THE EXECUTIVE ROUTINE

Having downloaded the Population File onto the system the next step is to extract the Executive Routine from the Population File. The Executive Routine prepares the audit routines for compilation and execution. It inserts the user-defined information from the X-Cards and the Option Switches into the programs so that they are syntactically correct for the particular environment and compiler to be tested.

The Executive Routine, labelled EXEC85, is the first program in the Population File. It is held as source code and begins with the record *HEADER,COBOL,EXEC85 and ends with the record *END-OF,EXEC85. It is usually extracted using an editor. EXEC85 has been enhanced so that it is now capable of extracting itself from the population file (CCVS Version 2.0 and later). Therefore, after EXEC85 has been extracted once using an editor, it then can be used for future extractions of EXEC85.

3.3 EDITING AND COMPIILING THE EXECUTIVE ROUTINE

In order to run EXEC85 in a particular environment it is necessary to edit the SELECT ASSIGN statements for that environment. The parameter file that EXEC85 uses may also be modified to select out individual programs - see section 3.6, Running the Audit Routines.

When all required modifications have been effected, EXEC85 should be compiled to produce an object code ready for running.

3.4 PREPARING THE EXECUTIVE INPUT DATA

Input data must be prepared before the EXEC85 can be run. This should be set up as a series of 80-character records and include the optional "*" commands (as detailed in 3.4.1), option switches (3.4.2), x-cards (3.4.3) and any temporary program fixes as will be notified from time to time (3.4.4). A sample executive input listing (CONTROL- CARD-FILE) is shown in appendix E.

3.4.1 "!" COMMANDS

These are the first records in the file. They begin in column 1 and should be included only as required. The options are:

***EXTRACT-ALL** Selects all programs in the Population File.

OR

***EXTRACT-AUTO** Selects "Automatic" programs only, ie those ending in 'A'. This option cannot be specified if ***EXTRACT ALL** is used.

OR

***EXTRACT-MAN** Selects "Manual" programs only, ie those ending in 'M'. This option cannot be specified if ***EXTRACT ALL** is used.

ELSE

***SELECT-PROG XXXXXX** where XXXXXX is a program name. This program and any programs which are dependent on it are then automatically selected. Appendix B lists all programs with dependent programs. This option cannot be specified if any of the *EXTRACT options are used. There may be up to 50 of these entries.

ELSE

*SELECT-MODULE XX L where XX is a module abbreviation (eg RL, NC etc.) and L is the level number ie 1 or 2. If L is omitted both levels for that module are automatically selected. This option cannot be selected if any of the *EXTRACT options are used. There may be up to 10 of these entries.

Note: where a module contains OB programs (ie, programs testing for obsolete language elements) the OB module should be selected in addition to the required module.

*KILL-DELETIONS

Removes all record of unselected Option Switches from the Source File produced as output by the Executive Routine (see 3.4.2 for details of Option Switches). If this command is not specified all unused optional code is converted into comment lines.

***LIST X-CARDS**

Causes all "XXXXX" card replacements to be listed in the Executive Output Report, showing the actual text which replaced each "XXXXX" card (see 3.4.3 for of X-cards).

***LIST NO-UPDATES**

In the Executive Output Report no program updates (ie temporary program fixes, see 3.4.4) are to be printed. The default action is to print the updates.

AND

*LIST PROGRAMS	Causes each program selected to be listed in full in the Executive Output Report complete with Temporary Program Fixes and X-card replacements. This option can be selected only if the *LIST NO-UPDATES option is also selected.
*LIST COMPACT	Indicates that the printer output should be as compressed as possible; ie the report will not skip to a new page for each program processed.
*REMOVE-COMMENTS	Comment lines in the Population File will not be copied to the Source File. The default action is to copy the data to the Source File.
*NO-DATA	Some programs require data files which are not produced by the audit routines, notably some of the Nucleus programs, and this data is held in the Population File immediately following the program which uses it. This option indicates that this data is not to be copied to the Source File. The default is to be copied to the Source File.
*NO-LIBRARY	Some programs (notably the Source Text Manipulation module) require copy library files which have names starting with the letter "K" or "A" on the population file. This option indicates that this data is not to be copied to the Source File. The default is to be copied to the Source File.
*NO-SOURCE	This indicates that the Source File will not be created by running the Executive Routine. If this option is present the *NEW-POP option must also be specified. For further details on the file see 3.5.3. The default option is to create a Source File.
AND	
*NEW-POP	This indicates that a new Population File will be created by running the Executive Routine. For further details on the file see Section 3.5.3. If this option is present, the *EXTRACT-ALL option must also be specified. The default option is to not create a new Population File.

3.4.2 OPTION SWITCHES

The option switches permit the elimination or inclusion of source code governing features which are implementation dependent. By entering the option codes into the Executive Input prior to running the Executive Routine, programs containing these optional elements may be run without having to make updates to the source programs themselves.

The functions/language elements which can be included or eliminated through the use of option switches include the VALUE OF clause associated with mass storage files, switch status tests, CLOSE REEL statements and CLOSE UNIT statements. Additionally, the use of an option switch can control the type of WRITE statements used for printer destined files created by each audit routine.

The option switch cards are entered after the last "**" command, in the format:

```
*OPT01 x  
*OPT02 x  
*OPT10 x  
etc.
```

The asterisk must be in column 1 and there must be one space between the option switch number and the value to which it is to be set.

For a given switch there are two or more choices. The Executive Routine does not check that a valid setting has been entered for a particular switch but an invalid setting will affect the source programs which use that switch. If no deviations are necessary from the default switch settings then no *OPT cards need be present.

Appendix C shows the possible option switch settings, including the default value for each switch.

3.4.3 X-CARDS

X-cards provide the ability to transport the audit routines to different computer environments. They contain implementor-names needed in the Environment and Data Divisions of the audit routines. When the Executive Routine is run the data in the X-cards are placed in the source code of the audit routines, resulting in syntactically correct programs for a given environment and compiler.

The X-card is held in the source code in the format XXXXXnnn and in the executive input the X-card data is labelled X-nnn, so that the data held against X-001 replaces all occurrences of XXXXX001 in the audit routines.

Appendix D gives details of the X-card contents.

X-cards are entered into the executive input file immediately after the last *OPT record, in the format:

X-nnnpp SUBSTITUTION TEXT.

Columns 1 and 2 contain the characters "X-".

Columns 3 to 5 (nnn) contain the three-digit number of the X-card.

Columns 6 and 7 (pp) may be left blank or may contain a two-digit value representing the position, within the substitution text, at which the program name will be placed (see example below).

Column 8 is left blank.

Columns 9 to 69 contain the substitution text, followed by a period.

eg X-00104 PRTXXXXXX.

This will replace XXXXX001 in the source program with PRTXL101A for program RL101A, PRTNC101A for program NC101A etc.

Note: where programs contain subprograms, the name of the main program rather than the subprogram can be inserted by putting the character "J" as the first character of the text to be substituted (as indicated by columns 6 and 7 of the X-card). The program name is truncated to six characters.

eg X-01401 JXXXXXDISK.

This will replace XXXXX014 in the source program IC228A-1 with IC228ADISK.

3.4.4 FINAL "*" COMMANDS AND TEMPORARY PROGRAM FIXES

After the last X-card there are three more records which must be entered. These all begin in column 1 and are as follows:

```
*END-MONITOR  
*BEGIN-UPDATE  
*END-UPDATE
```

If any modifications are required to the test programs, such as temporary program fixes issued by NCC, these may be inserted immediately after the *BEGIN-UPDATE card. The format is as shown in the following example, where XXXXXX represents a program name and the "R", which is optional, instructs the Executive Routine to resequence the program line numbers. Where a line number is followed by text this will replace the text currently held on that line, and any further text will be inserted in the program immediately after the specified line number, in this case, line 000010. A line number followed by a comma will be deleted, and where a line number is followed by a comma and a second line number, that block of lines, inclusive of the two line numbers and any that may be out-of-sequence located within the block of lines, will be deleted from the program.

Example:

```
*BEGIN-UPDATE  
*START,XXXXXX,R  
000010 text.....  
text.....  
text.....  
000200 text.....  
000300,  
*START, XXXXXX  
000500,000600  
*END-UPDATE
```

Any number of programs may be updated in this way. The final update must be followed by the *END-UPDATE card. Appendix I contains the current temporary program fix list and this appendix will be re-issued as necessary.

3.5 RUNNING THE EXECUTIVE ROUTINE

When all previous steps have been carried out the Executive Routine may be run. The following output files are produced:

3.5.1 PRINT-FILE

A print file is produced reporting on the actions carried out by the Executive Routine. This is a standard 132 column printer file.

3.5.2 NEW POPULATION-FILE

This has the same format as the Population File; ie source programs are held contiguously within the file, each beginning with a card containing *HEADER, COBOL, program-name, and ending with a card containing *END-OF,program-name. Both these statements begin in column 1. This file would not normally be selected as an output file for a validation, as the user-defined information from the X-cards and Option Switches are not inserted in this file. If this file is not selected as an output file, then the Source Programs File must be.

3.5.3 SOURCE-COBOL-PROGRAMS FILE

This file contains the programs from the Population File, processed as specified by the implementor-defined executive input data. X-card data will have replaced the mnemonics in the source code and selected optional code will have been included. The format of this file is similar to that of the Population File, the major difference being that the *HEADER and *END-OF cards are indented by six places so that they become COBOL comment lines. This file may be selected instead of, or, in addition to the New Population File.

3.6 RUNNING THE AUDIT ROUTINES

When the Executive Routine has been run the audit routines will then be ready for compilation. They must first be extracted from the Source Programs File created by the Executive Routine and held as individual programs. This can be done by means of either an editor or a program such as the sample shown in Appendix F. Another possibility would be to modify the Executive Routine itself to do this.

Having extracted the source programs, they must then be compiled and run. While it does not matter in which order the modules are run, it is sometimes important that the individual audit routines within a module are run in numerical sequence as one routine may use a file created in the previous routine. Also, in the Inter-Program Communication Module, some programs call subsequent programs so it may be necessary to compile these in reverse order, to ensure that the linkages are correct. Appendix B lists the program dependencies, and further details of individual programs are given in chapter 4. Note that some of the DB programs need running twice with different switch settings. Also, it is required that the COPY library files ALTL1 and ALTLB are placed in libraries whose names are equated to the X-48 and X-47 cards.

Appendix A shows which sets of programs must be run to test for conformity to a particular level of FIPS COBOL.

All the audit routines should compile and execute without errors. The results of running each audit routine are printed and in most instances 'passes' and 'failures' are printed on the reports. In some cases, however, a visual check is necessary, and audit routines containing such tests have names ending with the letter "M". Flagging tests expect flagging messages from the compiler at compile time. Results in this case depend on these messages being displayed at the correct points in the source code. In these cases either the program description in section 4 or the actual output produced by the test explains the checking required.

3.7 FLAGGING TESTS

3.7.1 USE OF THE FLAGGING TESTS

These tests have been written with the purpose of testing the flagging capabilities of compilers.

With this in mind the programs are purely for the purpose of testing flagging at compile time and thus need only be syntactically correct.

The programs are not to be run once compiled for the features therein serve no purpose other than to induce flagging.

As the flagging tests use comment lines to show where a flagging message must appear, it is important that the *REMOVE-COMMENTS option is NOT used in the EXEC85 parameter file.

The reference for each flagging test should be consulted to determine the relevant syntax to be flagged. (See section 4.13)

Messages are acceptable provided that they identify the following:

- The level indicator, clause, statement or header that directly contains the nonconforming or obsolete syntax. (For the purpose of this requirement the definitions of the level indicator, clause, statement and header contained in the COBOL standard, apply.)
- The source program line and an indication of the beginning location within the line of the level indicator, clause, statement or header which contains the nonconforming or obsolete syntax.
- The syntax as "nonconforming standard" if the nonconforming syntax is included in the processor but is not within the user selected FIPS COBOL subset or optional modules except if monitoring is selected for the obsolete category then obsolete language elements are only flagged as "obsolete".
- The syntax as "obsolete" if the syntax identified is in the obsolete category within a FIPS COBOL subset or optional module included in the processor.

The position of the messages within the program are irrelevant provided that the above requirements are fulfilled.

Throughout the CCVS the flagging specifications are based on those contained in FIPS PUB 21-3 rather than ANS COBOL. See FIPS PUB 21-3 for a full statement of the flagging specifications.

For ISO/ANSI testing, only those tests that flag OBSOLETE features are required.

3.7.2 METHOD OF OPERATION

The compiler producer has to make a switch option available for checking any feature beyond the most basic components of the ANSI 85 Standard. The appropriate combination of tests should be run, with the appropriate compiler switches set, and the number of flags obtained should be compared with the total indicated for each test.

The tests are intended to check that at each level the appropriate message is output and the feature identified.

As with the other areas of the test suite, the flagging programs compiled will depend on the level of validation.

For low level validations the following will need to be compiled:

NC302M	-	flagging OBSOLETE elements.
SQ302M	-	flagging OBSOLETE elements.

For intermediate level validations the following will need to be compiled:

IX301M	-	flagging NON-CONFORMING STANDARD.
RL301M	-	flagging NON-CONFORMING STANDARD.
ST301M	-	flagging NON-CONFORMING STANDARD.
SM301M	-	flagging NON-CONFORMING STANDARD.
NC302M	-	flagging OBSOLETE elements.
IX302M	-	flagging OBSOLETE elements.
RL302M	-	flagging OBSOLETE elements.
SQ302M	-	flagging OBSOLETE elements.

For high level validations the following will need to be compiled:

IC401M	-	flagging NON-CONFORMING STANDARD.
IF401M	-	flagging NON-CONFORMING STANDARD.
IF402M	-	flagging NON-CONFORMING STANDARD.
IF403M	-	flagging NON-CONFORMING STANDARD.
IX301M	-	flagging NON-CONFORMING STANDARD.
IX401M	-	flagging NON-CONFORMING STANDARD.
NC401M	-	flagging NON-CONFORMING STANDARD.
RL301M	-	flagging NON-CONFORMING STANDARD.
RL401M	-	flagging NON-CONFORMING STANDARD.
SM301M	-	flagging NON-CONFORMING STANDARD.
SM401M	-	flagging NON-CONFORMING STANDARD.
SQ401M	-	flagging NON-CONFORMING STANDARD.
ST301M	-	flagging NON-CONFORMING STANDARD.
IX302M	-	flagging OBSOLETE elements.
NC302M	-	flagging OBSOLETE elements.
NC303M	-	flagging OBSOLETE elements.
RL302M	-	flagging OBSOLETE elements.
SQ302M	-	flagging OBSOLETE elements.
SQ303M	-	flagging OBSOLETE elements.

The following optional flagging tests are available for compilers that support the following features:

CM303M	-	flagging OBSOLETE elements.
CM401M	-	flagging NON-CONFORMING STANDARD elements.
DB301M	-	flagging NON-CONFORMING STANDARD elements.
DB302M	-	flagging OBSOLETE elements.
DB303M	-	flagging OBSOLETE elements.
DB304M	-	flagging OBSOLETE elements.
DB305M	-	flagging OBSOLETE elements.
RW301M	-	flagging OBSOLETE elements.
RW302M	-	flagging OBSOLETE elements.
SG401M	-	flagging NON-CONFORMING STANDARD elements.
SG302M	-	flagging OBSOLETE elements.
SG303M	-	flagging OBSOLETE elements.

3.7.3 TEST OVERVIEW

The flagging program portion of the C CVS 85 consists of 28 programs which have been divided into four sections, each testing the flagging of distinct types of COBOL features.

The four sections are:

Intermediate
High
Obsolete
Optional.

3.7.3.1 Intermediate

Tests all the features that are above minimum level and below high level. Four programs test the features that deal with Relative and Indexed I-O, Sort-Merge and Source Text Manipulation.

3.7.3.2 High

Tests high level COBOL features with programs divided into the categories of Nucleus, Inter-program Communication, Intrinsic Functions, Source-Text Manipulation and Sequential, Relative and Indexed I-O.

3.7.3.3 Obsolete

Tests features that are to be made obsolete in the next revision of Standard COBOL. The programs in this section of tests are divided into the categories of Nucleus, Report Writing, Communication, Segmentation, Debugging and Sequential, Relative and Indexed I-O.

4. LIST OF AUDIT ROUTINES

Section 4 gives a brief description of each of the audit routines, the features tested, the number of individual tests and the X-numbers used.

The organization of COBOL specifications in the ANSI X3.23-1985 is based on a functional processing module concept. The standard defines 12 functional processing modules: Nucleus, Sequential I-O, Relative I-O, Indexed I-O, Inter-Program Communication, Sort-Merge, Source Text Manipulation, Report Writer, Communication, Debug, Segmentation and Intrinsic Functions. Nine of the modules have the elements within the module divided into level 1 elements and level 2 elements. Level 1 elements of a module are a subset of level 2 elements of the same module. Three of the modules contain only level 1 elements.

The above organization has been followed in this section. The modules follow in alphabetic order and the page numbering in this section is XX-n where "XX" is the module and "n" is the page number relating to that module. This method of organization will enable updates to be easily incorporated into this user guide through the re-issue of independent modules as and when required.

Updates will be issued together with any temporary program fixes and will be accompanied by an amendment sheet as detailed in Appendix H.

4.1 COMMUNICATIONS MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION XIV

FUNCTION:

The Communication module provides the ability to access, process, and create messages or portions thereof, and to communicate through a Message Control System (MCS) with communication devices. The elements of the Communication module are divided into two levels. Level 1 provides elements for the basic facilities to send or receive complete messages. Level 2 provides elements for a more sophisticated facility including the capability to send or receive segments of a message.

All the CM module programs should be observed at run time and the output inspected to determine the success or failure of that program.

CM101M

This program tests the basic COBOL Communication functions applicable to a system configuration consisting of at least one terminal used for input.

X-numbers: 30, 31, 55, 82, 83, 84.

This program contains 17 tests.

CM102M

This audit routine tests the basic COBOL Communication functions applicable to a system configuration consisting of at least one terminal used for output.

CM102M is run twice, once to illustrate the effect of KILL within WAIT i.e. that it does not immediately die and input can be maintained until time has elapsed for the WAIT. The second run is to illustrate KILL (without WAIT) and that the effect is immediate.

X-numbers: 32, 33, 55, 82, 83, 84.

This program contains 33 tests.

CM103M

The routine CM103M tests a COBOL Communication program's capacity to contain both an input and an output communication description interfacing to the same terminal, and thus to permit the same terminal to be used both for input and output operations.

X-numbers: 30, 31, 32, 33, 55, 82, 83, 84.

This program contains 1 test.

CM104M

CM104M tests a COBOL Communication program's capacity to handle two terminals, each being used for both input and output. The program is built to alternately poll each terminal. When a message is received from one terminal it is logged and then sent to the other terminal.

Another possible test configuration is a computer-to-computer link-up with shared resources. In this case, each CPU would serve as a terminal for the other, besides having a non-shared message terminal of its own. A message received from one terminal by its CPU would then be dispatched to the other CPU which receives it and in turn sends it to its message terminal.

X-numbers: 30, 31, 32, 33, 34, 35, 36, 37, 55, 82, 83, 84.

This program contains 1 test.

CM105M

CM105M tests a COBOL Communication program's capacity to handle a hierarchical queue structure. The Communication Module permits the use of up to four queue levels. In this program four-level queues are used to test the handling of priorities in a queue structure. A system configuration including one input terminal is required.

X-numbers: 31, 38, 39, 40, 41, 55, 82, 83, 84.

This program contains 11 tests.

CM201M

This program is designed to be invoked by the system, process all messages available and then terminate execution. CM201M requires a system configuration which includes one terminal usable for both input and output. This program does not produce a print file, accuracy may be checked on the display.

X-numbers: 32, 33, 82, 83.

This program contains 1 test.

CM202M

This program tests special message handling functions supported only by a Level 2 Communication Module. The major items included are:

- DESTINATION TABLE
- INPUT TERMINAL
- RECEIVE SEGMENT
- Format 1 SEND
- SEND WITH ESI
- SEND WITH identifier

X-numbers: 30, 31, 32, 33, 35, 42, 43, 55, 82, 83, 84.

This program contains 28 tests.

4.2 DEBUG MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION XV

FUNCTION:

The Debug module provides a means by which the user can describe his debugging algorithm including the conditions under which data items or procedures are to be monitored during the execution of the object program. The elements of the Debug module are divided into two levels. Level 1 provides a basic debugging capability, including the ability to specify selective or full paragraph monitoring. Level 2 provides the full COBOL debugging capability.

DB101A DB101A tests the basic operation of the Debug Module when both the compile and object time debugging switches are turned on. The program contains both debug lines and simple debugging procedures. The debugging procedures are specified for procedure-names and procedure-name series. The following conditions are evaluated for the "DEBUG-ITEM" register:

- Start of program
- Reference by "ALTER"
- Reference by "GO TO"
- Reference by "PERFORM"
- Sequential passage of control (fall through)

Before beginning execution of the object program, the job control language commands necessary to activate (turn on) the object time debugging switch should be submitted.

X-numbers: 55, 82, 83, 84.

This program contains 34 tests.

DB102A DB102A tests the basic operation of the Debug Module when the compile time debugging switch is on and the object time switch is off. All debug lines and debugging procedures should be included in compilation and generate code.

Before beginning execution of the object program, the job control language commands necessary to deactivate (turn off) the object time debugging switch should be submitted.

At execution time, code generated from debug lines should be executed, but debugging procedures should be deactivated by the object time switch.

X-numbers: 55, 82, 83, 84.

This program contains 14 tests.

DB103M DB103M tests the basic operation of the Debug Module when the compile time debugging switch is off. All debug lines should be treated as comments and no code should be generated for either debug lines or debugging procedures.

The object program for DB103M should be executed twice; once with the object time debugging switch enabled (on) and once with the object time debugging switch disabled (off). Both execution runs should yield the same results; the setting of the object time switch should make no difference since the compile time debugging switch was initially disabled.

X-numbers: 55, 82, 83, 84.

This program contains 14 tests.

DB104A DB104A tests the Debug Module's capacity to handle procedures tied to sort input, sort output and file declarative procedures. This program is to be compiled and executed with both compile and object time debugging switches enabled. The program first builds a sequential file containing 99 eighty-character records. This file is then sorted.

All debugging procedures should be included in compilation and generate code. Before beginning execution of the object program the job control language commands necessary to activate the object time debugging switch must be submitted.

Execution of the program's sort should trigger debugging procedures at the beginning of the sort input and sort output procedures. During execution of the sort input procedure, an end-of-file condition on the input file should trigger a declarative procedure associated with the file, and this in turn should cause execution of a debugging procedure monitoring the file declarative procedure.

The performance of the SORT verb is not checked in DB104A.

X-numbers: 14, 27, 55, 69, 74, 75, 82, 83, 84.

This program contains 12 tests.

DB105A DB105A tests the Debug Module's capacity to monitor all procedures with a single debugging declarative. This program is to be compiled and executed with both compile and object time debugging switches on. The debugging procedure should be included in the compilation and code should be generated for the procedure. During execution each procedure should trigger the debugging procedure which stacks the name of the procedure calling it. Prior to being stacked, each name is potentially adjusted by modifying a fixed-location numeric subfield in the name. If the program executes properly, the names that are stacked will be unique, and in an ascending sequence in the numeric subfield. Near the end of the program the stacking function is disabled and the name stack is compared to a static table containing procedure-names in the order in which the procedures should have been stacked.

DB105A's output report differs slightly from the normal CCVS format. If execution is perfect the report will consist of 227 lines showing:

1. Program procedure name as it appears in the program.
2. Adjusted procedure name after its numeric subfield has been adjusted.
3. Adjusted debug-name that was stacked by the debugging procedure.

The numeric subfields of the procedure names should appear in ascending sequence. Any deviations from the expected stacking sequence will cause additional report lines to be generated with one or more columns blank. If nothing ever appears in the "ADJUSTED DEBUG-NAME" column it may be assumed that the debugging procedure was never executed.

It is a fundamental assumption of DB105A that when a section is entered, the debugging section will be called twice; once for the section-name and once for the paragraph-name that immediately follows the section-name. Additionally, DB105A traps any failures in program flow caused by a failure of verbs from the Nucleus Module. These failures are summed and reported at the bottom of DB105A's report. If any procedure-names beginning with "PROC-000" appear in the "ADJUSTED DEBUG-NAME" column of the report, these result from execution of procedures which should not have been executed if the program had followed the proper control flow sequence.

X-numbers: 55, 82, 83, 84.

This program contains 227 tests.

DB201A DB201A tests the Debug Module's capacity to handle debugging procedures, which are monitoring identifiers specified with and without the "ALL REFERENCES" option. This program is to be compiled and executed with both compile and object time debugging switches enabled. The debugging procedures should be included in the compilation and code should be generated for them. Debugging sections on the following conditions are analysed:

1. Reference to identifier within "VARYING", "AFTER" and "UNTIL" phrases of PERFORM statements.
2. Reference to changed and unchanged identifier fields.
3. Reference to subscripted identifiers.
4. Reference to qualified identifiers.
5. Reference to identifier used in GO TO DEPENDING.
6. Reference to identifier in unexecuted statements.
7. Multiple references to same identifier in same statements.

X-numbers: 55, 82, 83, 84.

This program contains 68 tests of which 4 have been deleted.

DB202A DB202A tests the Debug Module's capacity to handle debugging procedures which are monitoring I-O functions of the Sequential I-O Module. This program is to be compiled and executed with both compile and object time debugging switches enabled. The debugging procedures should be included in the compilation and code should be generated for them.

During execution a sequential file is created containing 80-character records. The file is then read. Execution of OPEN, READ and WRITE statements should trigger the appropriate debugging procedures.

X-numbers: 14, 55, 69, 74, 75, 82, 83, 84.

This program contains 24 tests.

DB203A DB203A tests the capacity of the Debug Module to handle debugging procedures which are monitoring I-O functions of the Relative I-O or Indexed I-O Modules. This program is to be compiled and executed with both compile and object time debugging switches enabled. The debugging procedures should be included in the compilation and code should be generated for them. During execution a file is assigned in dynamic mode, created sequentially and accessed both sequentially and randomly. Its records are 80-characters in length. Execution of OPEN, READ, WRITE, REWRITE, START and DELETE statements should trigger the appropriate debugging procedures.

X-numbers: 24, 44, 55, 56, 69, 74, 75, 82, 83, 84.

This program contains 20 tests.

DB204A DB204A tests the capacity of the Debug Module to handle a debugging procedure which is monitoring a MERGE OUTPUT procedure. This program is to be compiled and executed with both compile and object time debugging switches enabled. The debugging procedures should be included in the compilation and code should be generated for the procedure.

During execution two sequential files are created, each containing 80-character records in sorted order. The two files are then merged. Execution of the merge operation should trigger the debugging procedure linked to the merge output procedure name.

X-numbers: 14, 15, 16, 27, 55, 69, 74, 75, 76, 77, 82, 83, 84.

This program contains 4 tests.

DB205A DB205A tests the capacity of the Debug Module to handle a debugging procedure which is monitoring a Communication procedure. This program is to be compiled and executed with both compile and object time debugging switches enabled. The debugging procedure should be included in the compilation and code should be generated for the procedure.

*This program need not be run if the Communications Module is not supported.

X-numbers: 30, 31, 32, 33, 55, 82, 83, 84.

This program contains 15 tests.

4.3 INTER-PROGRAM COMMUNICATION MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION X

FUNCTION:

The Inter-Program Communication module provides a facility by which a program can communicate with one or more programs. This communication is provided by:

- a) the ability to transfer control from one program to another within a run unit and
- b) the ability to pass parameters between programs to make certain data value available to a called program.

The elements of the Inter-Program Communication module are divided into two levels. Level 1 provides elements for the transfer of a control to another program known at compile time; it also provides for the access of certain common data items by both programs. Level 2 provides elements for the transfer of control to another program not identified at compile time; it also provides for the nesting of programs within other programs.

IC101A IC101A checks the use of the CALL statement with one parameter in the USING phrase. Subsequent CALLs check that the called routine remains in the last used state. This program CALLs subprogram IC102A. There are no test deletion paragraphs since these are the basic CALL tests, and if a CALL statement is rejected there is no reason to run the routine.

The first three CALLs use a data-name the same as the name in the subprogram PROCEDURE DIVISION header USING phrase. The last two CALLs use a data-name different from the name in the subprogram. The PICTURE clauses for data-names in the USING phrases of the called and calling programs are identical.

X-numbers: 55, 82, 83.

This program contains 5 tests.

IC102A This subprogram is called by IC101A and tests the use of the LINKAGE SECTION and the USING phrase in the PROCEDURE DIVISION header.

X-numbers: 55, 82, 83.

This program contains 0 tests.

IC103A This program tests the use of multiple data-names in the USING phrase of the CALL statement. Two 01 group items and an elementary 77 item are the parameters. The data definitions for the group item parameters are not the same as in the subprogram but the number of characters is identical. This program CALLs subprograms IC104A and IC105A.

X-numbers: 55, 82, 83.

This program contains 10 tests.

- IC104A The subprogram IC104A is called by IC103A and has three operands in the USING phrase of the PROCEDURE DIVISION header. Two operands are 01 group items and the third is an elementary 77 item. The data descriptions of these operands in the LINKAGE SECTION are not the same as those in the WORKING-STORAGE SECTION of the calling program, but an equal number of character positions is defined.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.
- IC105A The subprogram IC105A has two operands in the PROCEDURE DIVISION header and the routine contains four EXIT PROGRAM statements. The calling program is IC103A.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC106A This program CALLs subprogram IC107A with two tables and an index data item referenced in the USING phrase of the CALL statement. Both tables contain an INDEXED BY clause. The tests in this program verify that:
1. The indices in the main program and the subprogram are separate.
 2. An index data item set in a main program can be used to set an index in a subprogram.
 3. Tables can be shared between a main program and a subprogram.
- X-numbers: 55, 82, 83.
- This program contains 14 tests.
- IC107A This subprogram contains tables and an index data item which are defined in the LINKAGE SECTION and named as operands in the USING phrase of the PROCEDURE DIVISION header. One of the tables has an index defined for it. This index should be separate from the index defined for the same table in the main program IC106A, but no space should be allocated for the tables defined in the LINKAGE SECTION. The index data item is set in the main program prior to calling IC107A and it is used in this subprogram to set an index for referencing the table in the subprogram.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.

- IC108A The program IC108A is the main program which starts a sequence of CALLs to the subprograms IC109A, IC110A and IC111A. Parameters are set in each of these subprograms and checked when control is returned to the main program.
- X-numbers: 55, 82, 83.
- This program contains 9 tests.
- IC109A This subprogram is the first of a series CALLed by the main program IC108A. IC109A CALLs IC110A with one operand in the WORKING-STORAGE SECTION and one operand in the LINKAGE SECTION.
- X-numbers: 55, 82, 83.
- This program contains 0 tests
- IC110A This subprogram is CALLed by the subprogram IC109A, which was itself CALLed by the main program IC108A. IC110A CALLs IC111A with operands in the LINKAGE SECTION and the WORKING-STORAGE SECTION.
- X-numbers: 55, 82, 83.
- This program contains 0 tests
- IC111A This subprogram is CALLed by the subprogram IC110A and is the last in a sequence of subprogram CALLs which started in the main program IC108A.
- X-numbers: 82, 83.
- This program contains 0 tests
- IC112A This main program has a file description for a sequential mass storage file with fixed-length records. The file is created, CLOSEd and OPENed as an input file. The main program reads and verifies the file. The file is then CLOSEd and OPENed again as an input file, a record is read and a CALL made to the subprogram IC113A with the file description 01 record listed as one of the operands of the USING phrase. IC113A compares the fields in the input record to the values written during file creation. This program was adapted from the Sequential I-O tests contained in program SQ104A. If any errors occur in running the routine SQ104A the results of the tests in routines IC112A and IC113A are inconclusive.
- X-numbers: 14, 55, 69, 74, 75, 82, 83.
- This program contains 3 tests.

- IC113A This subprogram is CALLED by the main program IC112A which references a file description record in the USING phrase of the CALL statement. IC113A checks the values in the file record described in the LINKAGE SECTION of the subprogram. If any errors are encountered the error flag is set to 1 and the RECORDS-IN-ERROR counter is incremented by 1.
- X-numbers: 82, 83.
- This program contains 0 tests
- IC114A This main program CALLS the subprogram IC115A. The purpose of these programs is to verify that a FILE SECTION, WORKING-STORAGE SECTION and a LINKAGE SECTION can appear in a subprogram, and that a file can be written and read within a subprogram.
- The program IC114A CALLS IC115A to create and verify the file. Subsequent CALLS to the subprogram are made to read the file and return a record to the main program which checks the record contents. The subprogram IC115A is adapted from the Sequential I-O program SQ104A. If SQ104A does not execute correctly then the results of these tests are inconclusive.
- X-numbers: 55, 82, 83.
- This program contains 3 tests
- IC115A This subprogram is CALLED by IC114A. It contains a FILE SECTION, WORKING-STORAGE SECTION and LINKAGE SECTION. A file is created and verified then opened and read again. Each record is checked by moving it to the LINKAGE SECTION and returning to the main program to verify the record contents. The printing of the test results is performed by returning to the main program. This subprogram is adapted from the Sequential I-O routine SQ104A. If that routine does not perform correctly then the results of these tests are inconclusive.
- X-numbers: 14, 69, 74, 75, 82, 83.
- This program contains 0 tests.
- IC116M This main program tests the CALL statement without the optional USING phrase. The subprograms IC117M and IC118M test the omission of the optional USING phrase in the PROCEDURE DIVISION header of a subprogram. The program IC116M calls the subprogram IC117M which in turn CALLS IC118M. The subprograms contain the DISPLAY statements which show the execution sequence as follows:
- IC117M called
IC118M called
Returned to IC117M.
- X-numbers: 55, 82, 83.
- This program contains 1 test.

- IC117M This subprogram does not contain a LINKAGE SECTION or the optional USING phrase in the PROCEDURE DIVISION header. IC117M is CALLED by the main program IC116M and CALLs IC118M. The CALL statement does not have the optional USING phrase. DISPLAY statements are used to verify the program execution sequence.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC118M This subprogram is CALLED by the subprogram IC117M. It does not contain a LINKAGE SECTION or the optional USING phrase in the PROCEDURE DIVISION header. A DISPLAY statement is used to verify that this subprogram was executed.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC201A The main program IC201A CALLs the subprogram IC202A and tests the CALL statement with an identifier as an operand, and four operands in the USING phrase. The repetition of a data-name in the USING phrase is tested and the use of the ON OVERFLOW phrase in a CALL statement is syntactically checked in the program.
- X-numbers: 55, 82, 83.
- This program contains 11 tests.
- IC202A This subprogram is CALLED by IC201A. It has four operands in the USING phrase of the PROCEDURE DIVISION header.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.
- IC203A The main program IC203A tests the use of the CANCEL statement. This program CALLs IC204A and IC205A and verifies that the initial CALL to a subprogram and the first CALL after a CANCEL results in a subprogram being entered in its initial state. The program also CANCELS a program which has not been CALLED, in which case control should pass to the next sentence.
- X-numbers: 55, 82, 83.
- This program contains 21 tests.

- IC204A The subprogram IC204A, which is CALLED by IC203A and IC205A, has two variables in WORKING-STORAGE which are initialised by a VALUE statement. The data contents of these variables are modified during execution of the subprogram. Indicators are set for variables in the LINKAGE SECTION which relate how many times the subprogram has been CALLED since it was initialised and whether or not the subprogram is in its initial state.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.
- IC205A IC205A tests the use of the CANCEL statement within a subprogram. This subprogram is CALLED by IC203A and CALLs subprograms IC204A and IC206A.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.
- IC206A The subprogram IC206A is CALLED by the subprogram IC205A. The subprogram is then CANCELled and CALLED again. The program IC205A checks if IC206A was in its initial state on the first CALL after the program was CANCELled. The LINKAGE parameter DN1 contains the number of times IC206A has been CALLED since initialisation.
- X-numbers: 55, 82, 83.
- This program contains 0 tests.
- IC207A This program defines a variable length table. The table and the variable containing the table length are operands in a CALL statement USING phrase. Also, an index is defined for the table and an index data item is used to pass an index value for a table reference to and from the subprogram IC208A.
- X-numbers: 55, 82, 83.
- This program contains 11 tests.
- IC208A This subprogram contains tables and an index data item which are defined in the LINKAGE SECTION and named as operands in the USING phrase of the PROCEDURE DIVISION header. One of the tables is defined with an OCCURS DEPENDING ON clause and has condition-name entries associated with it. The SEARCH statement is used to test a variable length table capability.
- X-numbers: 82, 83.
- This program contains 0 tests.

- IC209A This program tests the ability to CANCEL a subprogram CALLED by a subprogram which is not referenced by the main program.
1. IC209A CALLs IC210A
 2. IC210A CALLs IC211A
 3. IC210A CALLs IC212A
 4. IC210A CANCELS IC211A
- X-numbers: 55, 82, 83.
- This program contains 4 tests.
- IC210A This subprogram is CALLED by IC209A and then CALLs IC211A and IC212A, and CANCELS IC211A.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC211A This subprogram is CALLED by IC210A and tests the use of the LINKAGE SECTION and the USING phrase in the PROCEDURE DIVISION header.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC212A This subprogram is CALLED by IC210A and tests the use of the LINKAGE SECTION and the USING phrase in the PROCEDURE DIVISION header.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC213A IC213A also tests the ability to CANCEL a subprogram CALLED by a subprogram which is not referenced in the main program.
1. IC213A CALLs IC214A
 2. IC213A CALLs IC215A
 3. IC215A CANCELS IC214A
- X-numbers: 55, 82, 83.
- This program contains 3 tests.
- IC214A This subprogram is CALLED by IC213A and tests the use of the LINKAGE SECTION and the USING phrase in the PROCEDURE DIVISION header.
- X-numbers: 82, 83.
- This program contains 0 tests.

- IC215A This subprogram is CALLED by IC213A and tests the use of the LINKAGE SECTION and the USING phrase in the PROCEDURE DIVISION header. This subprogram also CANCELS subprogram IC214A.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC216A IC216A tests the use of data items within a program which are subordinate to the items specified in the USING phrase of the PROCEDURE DIVISION. IC216A CALLS IC217A.
- X-numbers: 55, 82, 83.
- This program contains 2 tests.
- IC217A This subprogram is CALLED by IC216A and tests the use of the LINKAGE SECTION and data items that are subordinate to those specified in the USING phrase of the PROCEDURE DIVISION.
- X-numbers: 82, 83.
- This program contains 0 tests.
- IC222A The source file contains two programs, IC222A and IC222A-1, which are intended to be compiled by a single invocation of the compiler. IC222A is the first program in the file, and contains the substantive tests. IC222A-1 is present only to provide a known program which exists and is available to be called. This test set does not examine the results returned by the CALL program, but is wholly concerned with the flow of control in the calling program during execution of a CALL statement.
- The program tests eight combinations of calling an available program, calling a non-available program, the ON EXCEPTION and NOT ON EXCEPTION phrases and the ON OVERFLOW phrase. Every CALL statement has an END-CALL scope delimiter which is followed by one or more statements in the sentence, and various tests check that this additional statement has been executed.
- X-numbers: 55, 82, 83.
- Reference: X-27, 28, 30.
- This program contains 16 tests.

- IC223A This program IC223A CALLs the subprogram IC223A-1 using the BY REFERENCE phrase. The programs are compiled in one invocation of the compiler.
- X-numbers: 55, 82, 83.
- Reference: X-27, 29, 30.
- This program contains 11 tests.
- IC224A The main program IC224A CALLs subprogram IC224A-1 using the BY CONTENT phrase. The programs are compiled in one invocation of the compiler.
- X-numbers: 55, 82, 83.
- Reference: X-27, 29, 30.
- This program contains 44 tests.
- IC225A The main program IC225A CALLs subprogram IC225A-1 using the BY REFERENCE and BY CONTENT phrases. The programs are compiled in one invocation of the compiler.
- X-numbers: 55, 82, 83.
- Reference: X-27, 29, 30.
- This program contains 36 tests.
- IC226A The main program IC226A CALLs subprogram IC226A-1 CALLs using the EXTERNAL clause in the WORKING-STORAGE SECTION. The programs are compiled in one invocation of the compiler.
- X-numbers: 55, 82, 83.
- Reference: X-20, 21, 23, 30.
- This program contains 4 tests.
- IC227A The main program IC227A CALLs subprogram IC227A-1 using the EXTERNAL clause in the File Description entry. The programs are compiled in one invocation of the compiler.
- X-numbers: 14, 55, 82, 83.
- Reference: X-15, 21, 23, 30.
- This program contains 23 tests of which 4 are deleted.

- IC228A The main program IC228A contains the subprogram IC228A-1 which it CALLs using the GLOBAL clause in the WORKING-STORAGE SECTION.
- X-numbers: 55, 82, 83.
- Reference: X-20, 21, 24, 30
- This program contains 4 tests.
- IC233A The main program IC233A contains the subprogram IC232A-1. These programs test that a USE procedure in a calling program is invoked by a qualifying condition occurring in a contained program.
- X-numbers: 18, 55, 82, 83.
- Reference: X-34.
- This program contains 1 test.
- IC234A The subprogram IC234A-3 is contained within the subprogram IC234A-2, which is contained within the subprogram IC234A-1, which is in turn contained within the main program IC234A. The programs test that a USE procedure in a calling program is invoked by a qualifying condition occurring in a contained program nested to four levels.
- X-numbers: 14, 55, 82, 83.
- Reference: X-34.
- This program contains 1 test.
- IC235A IC235A comprises a main program with two subprograms, namely IC235A-1 and IC235A-2, both of which are CALLED by the main program, IC235A. The programs are compiled in one invocation of the compiler. The EXIT PROGRAM, END PROGRAM and USING phrases are tested.
- X-numbers: 55, 82, 83.
- This program contains 12 tests.
- IC237A IC237A CALLs the subprogram IC237A-1 to test the accessing of a redefined LINKAGE SECTION item. The programs are compiled in one invocation of the compiler.
- X-numbers: 55, 82, 83.
- This program contains 1 test.

- OBIC1A** The main program OBIC1A CALLs the subprogram OBIC2A which contains a SORT statement. These programs verify that a SORT statement functions correctly in a subprogram. Control is not returned to this program since OBIC2A contains a STOP RUN statement.
- X-numbers: 82, 83.
- This program contains 0 tests.
- OBIC2A** The subprogram OBIC2A tests the use of a SORT statement in a segmented program. The first non-declarative section of the subprogram consists of a SORT statement and a STOP RUN statement in a fixed permanent segment. The sort input procedure and sort output procedure are contained in two independent segments. The main program OBIC1A CALLs this subprogram and the subprogram OBIC3A is CALLed from the output procedure section to print the output report.
- X-numbers: 27, 82, 83.
- This program contains 0 tests.
- OBIC3A** The subprogram OBIC3A prints the test results for the program set OBIC1A, OBIC2A and OBIC3A. It is CALLed by the main program OBIC1A and the subprogram OBIC2A. A linkage variable indicates whether the heading, footing or a report line is to be printed.
- X-numbers: 55, 82, 83.
- This program contains 1 test.

4.4 INTRINSIC FUNCTION MODULE

ANSI DOCUMENT X3.23A-1989

"INTRINSIC FUNCTION ADDENDUM TO ANSI
STANDARD COBOL X3.23-1985"

FUNCTION:

The intrinsic function module provides the capability to reference a data item whose value is derived automatically at the time of reference during the execution of the object program.

This module is optional for ISO/ANSI testing.

IF101A This program tests the intrinsic function ACOS, which returns a numeric values in radians that approximates the arccosine of argument-1. The type of this function is numeric. The valid domain is $-1 \leq \text{arg1} \leq 1$ and valid range is ≥ 0 and $\leq \pi$. The type of argument-1 must be of class numeric. The returned value is the approximation of the arccosine of argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-33, Section 2.5

This program contains 26 tests.

IF102A This program tests the intrinsic function ANNUITY, which returns a numeric value that approximates the ratio of an annuity paid at the end of each period for the number of periods specified by argument-2 to an initial investment of one. Interest is earned at the rate specified by argument-1 and it is applied at the end of the period, before the payment. The type of this function is numeric. Argument-1 must be of class numeric. The returned value depends on the value of argument-1 as follows:

```
if arg1 = 0:  
    1 / arg2  
if arg1 <> 0:  
    arg1 / (1 - (1 + arg1) ** (- arg2))
```

Arg1 = interest rate and must be a value ≥ 0 .

Arg2 = number of periods and must be a positive integer.

X-numbers: 55, 82, 83.

Reference: Page A-34, Section 2.6

This program contains 13 tests

- IF103A This program tests the intrinsic function ASIN, which returns a numeric value that approximates the arcsine of argument1. The type of this function is numeric. The valid domain is $-1 \leq \text{arg1} \leq 1$. and range is $\geq -\text{Pi}/2$ and $\leq +\text{Pi}/2$. Argument-1 must be of class numeric. The returned value is the approximation of the arcsine of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-35, Section 2.7
- This program contains 23 tests.
- IF104A This program tests the intrinsic function ATAN, which returns a numeric value in radians that approximates the arctangent of argument-1. The type of this function is numeric. The valid range is $> -\text{Pi}/2$ and $< \text{Pi}/2$. Argument-1 must be of numeric class. The returned value is the approximation of the arctangent of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-36, Section 2.8
- This program contains 27 tests.
- IF105A This program tests the intrinsic function CHAR, which returns a one-character alphanumeric value that is a character in the program collating sequence having the ordinal position equal to the value of argument-1. The type of this function is alphanumeric. Argument-1 must be an integer, whose value must be greater than zero and less than or equal to the number of positions in the collating sequence.
- X-numbers: 55, 82, 83.
- Reference: Page A-37, Section 2.9
- This program contains 8 tests.
- IF106A This program tests the intrinsic function COS, which returns a numeric value that approximates the cosine of an angle or arc expressed in radians, that is specified by argument-1. The type of this function is numeric. The valid range is $-1 \leq \& \leq 1$. Argument-1 must be class numeric. The returned value is the approximation of the cosine of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-38, Section 2.10
- This program contains 30 tests.

- IF107A This program tests the intrinsic function CURRENT-DATE, which returns a 21-character alphanumeric value that represents the calendar date, time of day and local time differential factor provided by the system on which the function is evaluated. The type of this function is alphanumeric. For additional information related to the returned values see pages A-39 & A-40 of X3.23A-1989, "INTRINSIC FUNCTION MODULE ADDENDUM TO AMERICAN NATIONAL STANDARD COBOL X3.23-1985".
- X-numbers: 55, 82, 83.
- Reference: Page A-39, Section 2.11
- This program contains 2 tests.
- IF108A This program tests the intrinsic function DATE-OF-INTEGER, which converts a date in the Gregorian calendar from integer date form to standard date form (YYYYMMDD). The type of this function is integer. The argument given must be a positive integer that corresponds to the number of days past December 31, 1600 in the Gregorian calendar. The returned value represents the ISO standard date equivalent of the integer specified by argument-1. The returned value is in the form (YYYYMMDD), where YYYY represents a year in the Gregorian calendar, MM represents the month of that year and DD represents the day of that month.
- X-numbers: 55, 82, 83.
- Reference: Page A-41, Section 2.12
- This program contains 10 tests.
- IF109A This program tests the intrinsic function DAY-OF-INTEGER, which converts a date in the Gregorian calendar from integer date form to Julian date form (YYYYDDD). The type of this function is integer. The argument must be a positive integer that corresponds to the number of days past December 31, 1600 in the Gregorian calendar. The returned value represents the Julian equivalent of the integer specified by argument-1. The returned value is in the form (YYYYDDD) where YYYY represents a year in the Gregorian calendar and DDD represents the day of that year.
- X-numbers: 55, 82, 83.
- Reference: Page A-42, Section 2.13
- This program contains 8 tests.

IF110A This program tests the intrinsic function FACTORIAL, which returns an integer that is the factorial of argument-1. The type of this function is integer. Argument-1 must be an integer greater than or equal to zero. If the value of argument-1 is zero, the value 1 is returned. If the value of argument-1 is positive, its factorial is returned.

X-numbers: 55,82, 83.

Reference: Page A-43, Section 2.14

This program contains 9 tests.

IF111A This program tests the intrinsic function INTEGER, which returns the greatest integer value that is less than or equal to the argument. The type of this function is integer. Argument-1 must be of class numeric.

X-numbers: 55, 82, 83.

Reference: Page A-44, Section 2.15

This program contains 23 tests.

IF112A This program tests the intrinsic function INTEGER-OF-DATE, which converts a date in the Gregorian calendar from standard date form (YYYYMMDD) to integer date form. The type of this function is integer. Argument-1 must be an integer of the form YYYYMMDD, where:

- a) YYYY - represents the year in the Gregorian calendar. It must be an integer greater than 1600.
- b) MM - represents a month and must be a positive integer less than 13.
- c) DD - represents a day and must be a positive integer less than 32 provided that it is valid for the specified month and year combination.

The returned value is an integer that is the number of days the date represented by argument-1 succeeds December 31, 1600 in the Gregorian calendar.

X-numbers: 55, 82, 83.

Reference: Page A-45, Section 2.16

This program contains 8 tests.

IF113A This program tests the intrinsic function INTEGER-OF-DAY, which converts a date in the Gregorian calendar year from Julian date form (YYYYDDD) to integer date form. The type of this function is integer. Argument-1 must be an integer of the form YYYYDDD where:

- a) YYYY - represents the year in the Gregorian calendar. It must be an integer greater than 1600.
- b) DDD - represents the day of the year. It must be an integer less than 367 provided that is valid for the year specified.

The returned value is an integer that is the number of days the date represented by argument-1 succeeds December 31, 1600, in the Gregorian calendar.

X-numbers: 55, 82, 83.

Reference: Page A-46, Section 2.17

This program contains 8 tests.

IF114A This program tests the intrinsic function INTEGER-PART, which returns an integer that is the integer portion of argument-1. The type of this function is integer. Argument-1 must be class numeric. If the value of argument-1 is zero, the returned value is zero. If the value of argument-1 is positive, the returned value is the greatest integer less than or equal to the value of argument-1. If the value of argument-1 is negative, the returned value is the least integer greater than or equal to the value of argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-47, Section 2.18

This program contains 23 tests.

IF115A This program tests the intrinsic function LENGTH, which returns an integer equal to the length of the argument in character positions. The type of the function is integer. Argument-1 may be a nonnumeric literal or a data item of any class or category.

X-numbers: 55, 82, 83.

Reference: Page A-48, Section 2.19

This program contains 8 tests.

- IF116A This program tests the intrinsic function LOG, which returns a numeric value that approximates the logarithm to the base e (natural log) of argument-1. The type of this function is numeric. Argument-1 must be class numeric and must be greater than zero. The returned value is the approximation of the logarithm to the base e of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-49, Section 2.20
- This program contains 25 tests.
- IF117A This program tests the intrinsic function LOG10, which returns a numeric value that approximates the logarithm to the base 10 of argument-1. The type of this function is numeric. Argument-1 must be class numeric and must be greater than zero. The returned value is the approximation of the logarithm to the base 10 of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-50, Section 2.21
- This program contains 32 tests.
- IF118A This program tests the intrinsic function LOWER-CASE, which returns a character string that is the same length as argument-1 with each uppercase letter replaced by the corresponding lowercase letter. The type of this function is alphanumeric. Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length. The character string returned has the same length as argument-1. If the computer character set does not include lower case letters, no changes take place in the character string.
- X-numbers: 55, 82, 83.
- Reference: Page A-51, Section 2.22
- This program contains 13 tests.

IF119A This program tests the intrinsic function MAX, which returns the content of the argument-1 that contains the maximum value. The type of this function depends upon the argument types as follows:

<u>Argument type</u>	<u>Function Type</u>
Alphabetic	Alphanumeric
Alphanumeric	Alphanumeric
All arguments integer	Integer
Numeric (some args. may be integer)	Numeric

If more than one argument-1 is specified, all arguments must be of the same class. If more than one argument-1 has the same greatest value, the content of the argument-1 returned is the leftmost argument-1 having that value. If the type of the function is alphanumeric, the size of the returned value is the same as the size of the selected argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-52, Section 2.23

This program contains 23 tests.

IF120A This program tests the intrinsic function MEAN, which returns a numeric value that is the arithmetic mean (average) of its arguments. The type of this function is numeric. Argument-1 must be class numeric. The returned value is defined as the sum of the argument-1 series divided by the number of occurrences referenced by argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-53, Section 2.24

This program contains 17 tests.

IF121A This Program tests the intrinsic function MEDIAN, which returns the content of the argument whose value is the middle value in the list formed by arranging the arguments in sorted order. The type of this function is numeric. Argument-1 must be class numeric. If the number of occurrences referenced by argument-1 is odd, the returned value is such that at least half of the occurrences referenced by argument-1 are greater than or equal to the returned value and at least half are less than or equal. If the number of occurrences referenced by argument-1 is even, the returned value is the arithmetic mean of the values referenced by the two middle occurrences.

X-numbers: 55, 82, 83.

Reference: Page A-54, Section 2.25

This program contains 17 tests.

IF122A This program tests the intrinsic function MIDRANGE, which returns a numeric value that is the arithmetic mean (average) of the values of the minimum argument and the maximum argument. The type of this function is numeric. Argument-1 must be class numeric. The returned value is the arithmetic mean of the greatest argument-1 value and the least argument-1 value.

FUNCTION MIDRANGE (arg1 ...)

X-numbers: 55, 82, 83.

Reference: Page A-55, Section 2.26

This program contains 17 tests.

IF123A This program tests the intrinsic function MIN, which returns the content of argument-1 that contains the minimum value. The type of this function depends upon the argument types as follows:

<u>Argument type</u>	<u>Function Type</u>
Alphabetic	Alphanumeric
Alphanumeric	Alphanumeric
All arguments integer	Integer
Numeric (some args. may be integer)	Numeric

If more than one argument is specified, all arguments must be of the same class. The returned value is the content of the argument-1 having the least value. If more than one argument-1 has the same least value, the content of argument-1 returned is the leftmost argument-1 having that value. If the type of the function is alphanumeric, the size of the returned value is the same as the size of the selected argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-56, Section 2.27

This program contains 23 tests.

IF124A This program tests the intrinsic function MOD, which returns an integer value that is argument-1 modulo argument-2. The type of this function is integer. Argument-1 and Argument-2 must be integers and the value of argument-2 must not be zero. The returned value is defined as:

$\text{arg1} - (\text{arg2} * \text{FUNCTION INTEGER}(\text{arg1}/\text{arg2}))$

X-numbers: 55, 82, 83.

Reference: Page A-57, Section 2.28

This program contains 21 tests.

- IF125A This program tests the intrinsic function NUMVAL, which returns the numeric value represented by the character string specified by argument-1. Leading and trailing spaces are ignored. The type of this function is numeric. The returned value is the numeric value represented by argument-1. The total number of digits in argument-1 must not exceed 18.
- X-numbers: 55, 82, 83.
- Reference: Page A-58, Section 2.29
- This program contains 20 tests.
- IF126A This function tests the intrinsic function NUMVAL-C, which returns the numeric value represented by the character string specified by argument-1. Any optional currency sign specified by argument-2 and any optional commas preceding the decimal point are ignored. The type of this function is numeric. The returned value is the numeric value represented by argument-1. If the DECIMAL-POINT IS COMMA clause is specified in the SPECIAL-NAMES paragraph, the functions of the comma and decimal point in argument-1 are reversed.
- X-numbers: 55, 82, 83.
- Reference: Page A-59, Section 2.30
- This program contains 30 tests.
- IF127A This program tests the intrinsic function ORD, which returns the ordinal position of argument-1 in the collating sequence for the program. The lowest ordinal position is 1. The type of this function is integer. Argument-1 must be one character in length and must be class alphabetic or alphanumeric. The returned value is the ordinal position of argument-1 in the collating sequence for the program.
- X-numbers: 55, 82, 83.
- Reference: Page A-60, Section 2.31
- This program contains 9 tests.

IF128A This program tests the intrinsic function ORD-MAX, which returns a value that is the ordinal number of the argument-1 that contains the maximum value. The type of this function is integer. If more than one argument-1 is specified, all arguments must be of the same class. The returned value is the ordinal number that corresponds to the position of the argument-1 having the greatest value in the argument-1 series. If more than one argument-1 has the same greatest value, the number returned corresponds to the position of the leftmost argument-1 having that value.

X-numbers: 55, 82, 83.

Reference: Page A-61, Section 2.32

This program contains 16 tests.

IF129A This program tests the intrinsic function ORD-MIN, which returns a value that is the ordinal number of the argument that contains the minimum value. The type of this function is integer. If more than one argument-1 is specified, all arguments must be of the same class. The returned value is the ordinal number that corresponds to the position of the argument-1 having the least value in the argument-1 series. If more than one argument-1 has the same least value, the number returned corresponds to the position of the leftmost argument-1 having that value.

X-numbers: 55, 82, 83.

Reference: Page A-62, Section 2.33

This program contains 17 tests.

IF130A This program tests the intrinsic function PRESENT-VALUE, which returns a value that approximates the present value of a series of future period-end amounts specified by argument-2 at a discount rate specified by argument-1. The type of this function is numeric. The returned value is an approximation of the summation of a series of calculations with each term in the following form:

$$\text{summation of } \{ \text{arg2} / (1 + \text{arg1})^{** n} \}$$

There is one term for each occurrence of argument-2. The exponent n, is incremented from one by one for each term in the series.

Arg1 = discount rate and must be greater than -1.

Arg2 = the series of integer and non-integer end-period amounts.

n = end-period position in arg2 series

X-numbers: 55, 82, 83.

Reference: Page A-63, Section 2.34

This program contains 21 tests.

IF131A This program tests the intrinsic function RANDOM, which returns a random number based on its argument (if any). The function is applied to non-negative numbers. If an argument is given then it is used as the seed value. All returned values should be in the range ≥ 0 and < 1 . For a given seed value on a given implementation, the sequence of pseudo-random numbers will always be the same. The implementor will specify the subset of the domain of argument-1 values that will yield distinct sequence of pseudo-random numbers. The subset must include the values from 0 through at least 32767.

X-numbers: 55, 82, 83.

Reference: Page A-64, Section 2.35

This program contains 8 tests.

IF132A This program tests the intrinsic function RANGE, which returns a value that is equal to the value of the maximum argument minus the value of the minimum argument. The returned value is equal to the greatest value of argument-1 minus the least value of argument-1. The type of this function depends upon the argument types as follows:

<u>Argument Type</u>	<u>Function Type</u>
All arguments integer	Integer
Numeric (some args. may be integer)	Numeric

Argument-1 must be class numeric.

X-numbers: 55, 82, 83.

Reference: Page A-65, Section 2.36

This program contains 15 tests.

IF133A This program tests the intrinsic function REM which returns a numeric value that is the remainder of argument-1 divided by argument-2. The type of this function is numeric. The returned value is specified by the expression:

$REM(arg1,arg2) = arg1 - (arg2 * FUNCTION INTEGER-PART (arg1 / arg2))$

argument-1 and argument-2 must be class numeric.
argument-2 must not be zero.

X-numbers: 55, 82, 83.

Reference: Page A-57, Section 2.28

This program contains 17 tests.

- IF134A This program tests the intrinsic function REVERSE, which returns a character string of exactly the same length as argument-1 and whose characters are exactly the same as those of argument-1, except that they are in reverse order. The type of this function is alphanumeric. Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length. If argument-1 is a character string of length n, the returned value is a character string of length n, such that for $1 <= j <= n$, the character in position j of the returned value is the character from position $n-j+1$ of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-67, Section 2.38
- This program contains 13 tests.
- IF135A This program tests the intrinsic function SIN, which returns a numeric value that approximates the sine of angle or arc, expressed in radians, that is specified by argument-1. The type of this function is numeric. Argument-1 must be class numeric. The returned value is the approximation of the sine of argument-1 and is greater than or equal to -1 and less than or equal to 1.
- X-numbers: 55, 82, 83.
- Reference: Page A-68, Section 2.39
- This program contains 31 tests.
- IF136A This program tests the intrinsic function SQRT, which returns a numeric value that approximates the square root of argument-1. The type of this function is numeric. Argument-1 must be class numeric and must be zero or positive. The returned value is the absolute value of the approximation of the square root of argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-69, Section 2.40
- This program contains 26 tests.

IF137A This program tests the intrinsic function STANDARD-DEVIATION, which returns a numeric value that approximates the standard deviation of its arguments. The type of this function is numeric. Argument-1 must be class numeric. The returned value is the approximation of the standard deviation of the argument-1 series. The returned value is calculated as follows:

- a) The difference between each argument-1 value and the arithmetic mean of the argument-1 series is calculated and squared.
- b) The values obtained are then added together. This quantity is divided by the number of values in the argument-1 series.
- c) The square root of the quotient obtained is then calculated. The returned value is the absolute value of this square root.

If the argument-1 series consists of only one value, or if the argument-1 series consists of all variable occurrences data items and the total number of occurrences for all of them is one, the returned value is zero.

X-numbers: 55, 82, 83.

Reference: Page A-70, Section 2.41

This program contains 17 tests.

IF138A This program tests the intrinsic function SUM, which returns a value that is the sum of the arguments. The type of this function depends upon the argument types as follows:

<u>Argument Type</u>	<u>Function Type</u>
All arguments integer	Integer
Numeric (some args. may be integer)	Numeric

Argument-1 must be class numeric.
The returned value is the sum of the arguments.

X-numbers: 55, 82, 83.

Reference: Page A-62, Section 2.33

This program contains 16 tests.

IF139A This program tests the intrinsic function TAN, which returns a numeric value that approximates the tangent of an angle or arc, expressed in radians, that is specified by argument-1. The type of this function is numeric. Argument-1 must be class numeric. The returned value is the approximation of the tangent of argument-1.

X-numbers: 55, 82, 83.

Reference: Page A-72, Section 2.43

This program contains 30 tests.

- IF140A This program tests the intrinsic function UPPER-CASE, which returns a character string that is the same length as argument-1 with each lowercase letter replaced by the corresponding uppercase letter. The type of this function is alphanumeric. Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length. The character string returned has the same length as argument-1.
- X-numbers: 55, 82, 83.
- Reference: Page A-73, Section 2.44
- This program contains 13 tests.
- IF141A This program tests the intrinsic function VARIANCE, which returns a numeric value that approximates the variance of its arguments. The type of this function is numeric. Argument-1 must be class numeric. The returned value is the approximation of the variance of the argument-1 series and it is defined as the square of the STANDARD DEVIATION of the argument-1 series. If the argument-1 series consists of only one value, or if the argument-1 series consists of all variable occurrence data items and the total number of occurrences for all of them is one, the returned value is zero.
- VARIANCE (arg1 ...)** = FUNCTION STANDARD-
DEVIATION(arg1...)**2
- X-numbers: 55, 82, 83.
- Reference: Page A-74, Section 2.45
- This program contains 16 tests.
- IF142A This program tests the intrinsic function WHEN-COMPILED, which returns the date and time the program was compiled as provided by the system on which the program was compiled. The type of this function is alphanumeric.
- X-numbers: 55, 82, 83.
- Reference: Page A-75, Section 2.46
- This program contains 2 tests.

4.5 INDEXED I-O MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION IX

FUNCTION:

The Indexed I-O module provides a capability to access records of a mass storage file in either a random or sequential manner. Each record in an indexed file is uniquely identified by the value of one or more keys within that record. The elements of the Indexed I-O module are divided into two levels. Level 1 provides elements for the basic facilities of definition and access of indexed files. Level 2 provides elements for more complete facilities, including alternate keys and the capability for accessing the file both randomly and sequentially in the same COBOL program.

IX101A This program creates and verifies a 500 record fixed length indexed file whose access mode is sequential. The file is passed to IX102A for further processing.

X-numbers: 24, 44, 55, 62, 82, 83.

This program contains 2 tests.

IX102A IX102A verifies the existence and accuracy of the 500 records in the indexed file created in IX101A, using access mode random. Certain records are then selectively updated and the accuracy of each record is verified again. The file is passed to IX103A for further processing.

X-numbers: 24, 44, 55, 62, 82, 83.

This program contains 11 tests.

IX103A This program verifies the indexed file updated in IX102A using access mode sequential. The file is verified for the accuracy of its 500 records, and then certain records are selectively deleted and the accuracy of each record verified again.

X-numbers: 24, 44, 55, 62, 82, 83.

This program contains 12 tests.

IX104A IX104A is used to create an indexed file, whose access mode is sequential, and then selectively update the file. The FILE STATUS code item is tested for each OPEN, CLOSE, READ and REWRITE statement. These statements are used without the AT END and INVALID KEY phrases.

X-numbers: 25, 45, 55, 62, 82, 83.

This program contains 13 tests.

- IX105A This program tests the capacity for variable length records in an indexed file. Three indexed files are created and verified, with access mode random. The record key data name is used to access specific records within each of the three files and to verify their accuracy.
- X-numbers: 24, 25, 26, 55, 62, 82, 83.
- This program contains 9 tests.
- IX106A This program tests the ability to use the three different types of files (sequential, indexed and relative) in one program.
- X-numbers: 14, 21, 24, 44, 55, 62, 82, 83.
- This program contains 10 tests.
- IX107A This program tests the use of the SAME AREA clause for two indexed files whose access mode is sequential. After each file is created it is processed and verified using various combinations of the READ statement with the AT END phrase.
- X-numbers: 24, 25, 44, 45, 55, 62, 82, 83.
- This program contains 14 tests.
- IX108A This program tests the ability to use two different types of file (sequential and indexed) in one program, and specifically the READ statement with the NOT AT END and END-READ phrases, the DELETE statement with the NOT INVALID KEY and END-DELETE phrases and the WRITE statement with the NOT INVALID KEY and END-WRITE phrases.
- X-numbers: 24, 25, 44, 45, 55, 62, 82, 83.
- This program contains 32 tests.
- IX109A This program creates a mass storage file containing 50 records of 240 characters, with two records per block. Its organisation is indexed and it is accessed sequentially. The I-O status code is checked at various points in the program after OPEN, READ, WRITE, REWRITE, DELETE and CLOSE statements. The file is then passed to IX110A and subsequently to IX111A for further processing.
- X-numbers: 24, 55, 62, 82, 83.
- This program contains 13 tests.

- IX110A This program uses the file created by IX109A. The file is opened in I-O mode and the status code checked after OPEN, WRITE (with duplicate primary record key), REWRITE (with duplicate primary record key) and READ (with a non-existent record key).
- X-numbers: 24, 55, 62, 82, 83.
- This program contains 4 tests.
- IX111A This routine uses the file created by IX109A. Attempts are made to open a non-existent file. The FILE STATUS code is checked in each case.
- X-numbers: 24, 25, 55, 62, 82, 83.
- This program contains 1 test.
- IX112A This creates a mass storage file of 50 records, 2 records per block, 240 characters per record, with indexed organisation and sequential access. The file is opened as I-O and each record is verified before an attempt is made to write a record with the wrong record length, ie, shorter than the one read. The FILE STATUS code is then checked.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) D1 and D2.
- This program contains 7 tests.
- IX113A This routine creates a mass storage file of 50 records of 240 characters, two records per block, indexed organisation and access sequential. The I-O status code is checked after OPEN, WRITE and CLOSE statements and then an attempt is made to CLOSE the file again and the status code checked (42 expected).
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) B.
- This program contains 4 tests.
- IX114A This program uses the file created by IX113A. An attempt is made to READ a record on a closed file. Status code 47 is expected, and the DECLARATIVES section should be actioned.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) F.
- This program contains 3 tests.

- IX115A This uses the file created by IX113A. An attempt is made to WRITE a record on a closed file. Status code 48 is expected, and the DECLARATIVES section should be actioned.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) G.
- This program contains 3 tests.
- IX116A This program uses the file created by IX113A. An attempt is made to DELETE a record on a closed file. Status code 49 is expected, and the DECLARATIVES section should be actioned.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) G.
- This program contains 3 tests.
- IX117A This program uses the file created by IX113A. An attempt is made to REWRITE a record on a closed file. Status code 49 is expected, and the DECLARATIVES section should be actioned.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) G.
- This program contains 3 tests.
- IX118A This program uses the file created by IX113A. The file is OPENed for INPUT twice. Status code 41 is expected, and the DECLARATIVES section should be actioned.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) A.
- This program contains 3 tests.
- IX119A The file created by IX113A is OPENed I-O then a record is READ and an attempt is made to REWRITE it using the wrong record key. The file status should be 21 or 22. An attempt is then made to DELETE a record, at which point the DECLARATIVES section should be actioned and the file status should be 43.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (3) A, H.
- This program contains 3 tests.

- IX120A The file created by IX113A is OPENed I-O then READ to the end and then READ again, and an attempt is made to REWRITE a record.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (3) H.
- This program contains 2 tests.
- IX121A This routine creates a mass storage file with 50 records, 2 records per block, 240 characters per record, organisation indexed and access sequential.
- The file is opened I-O and 5 records read then an attempt is made to rewrite a longer record than the one read. A file status code of either 0 or 44 is expected.
- X-numbers: 24, 55, 62, 82, 83.
- Reference: IX-5, 1.3.4 (5) D1 and D2.
- This program contains 3 tests.
- IX201A This creates and verifies a 500 record fixed length indexed file whose access mode is sequential. The file is passed to IX202A for further processing.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 2 tests.
- IX202A IX202A verifies the indexed file created in IX201A. The ACCESS MODE IS DYNAMIC clause is used for the file in this program. The existence and accuracy of the records verified, then certain records are selectively updated and the accuracy of each record verified again. The updated indexed file is passed to IX203A for further processing.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 11 tests.
- IX203A IX203A is the third in a series. It is used to verify the indexed file updated in IX202A, using the ACCESS MODE IS DYNAMIC clause. The file is verified sequentially for accuracy of its 500 records, then certain records are selectively deleted and the accuracy of each record is verified again.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 12 tests.

IX204A This program is used to create an indexed file sequentially using the ACCESS DYNAMIC clause, and then update selected records within the file using the REWRITE statement. The FILE STATUS code is tested for each OPEN, CLOSE, READ and REWRITE statement; these statements are used without the optional AT END or INVALID KEY phrases.

X-numbers: 25, 45, 55, 62, 82, 83.

This program contains 13 tests.

IX205A IX205A tests various syntactical constructs of level 2 of the Indexed I-O module, namely:

1. ACCESS MODE DYNAMIC
2. ALTERNATE RECORD KEY without duplicates option
3. RESERVE clause
4. SAME clause
5. BLOCK CONTAINS... TO... clause
6. VALUE OF... clause

X-numbers: 24, 25, 44, 45, 55, 62, 82, 83, 86.

This program contains 12 tests.

IX206A One file is created and accessed using the ACCESS DYNAMIC clause. A second is accessed using the ACCESS MODE IS SEQUENTIAL clause. The constructs tested are:

1. ACCESS DYNAMIC
2. ACCESS MODE IS SEQUENTIAL
3. ALTERNATE RECORD KEY without duplicates options
4. RESERVE clause
5. SAME clause
6. BLOCK CONTAINS... TO... clause
7. VALUE OF implementor-name series

X-numbers: 24, 25, 44, 45, 55, 62, 82, 83, 86, 87, 88.

This program contains 10 tests.

IX207A A file is created and accessed in the SEQUENTIAL ACCESS mode. The constructs tested are:

1. Ordering of clauses in file control entry
2. ALTERNATE RECORD KEY with duplicates options
3. USE AFTER STANDARD EXCEPTION file-name series
4. FILE STATUS clause

X-numbers: 24, 25, 44, 45, 55, 62, 82, 83.

This program contains 8 tests.

- IX208A Two indexed files are created and verified. One file is created with ACCESS MODE IS DYNAMIC, the other with ACCESS MODE IS SEQUENTIAL. After each file has been verified it is read sequentially, using the START statement to logically position the file.
- X-numbers: 24, 25, 44, 45, 55, 56, 62, 82, 83.
- This program contains 29 tests.
- IX209A IX209A tests the use of the START... EQUAL TO... statement using the prime record key and each of the alternate record keys as the key of reference. Included within the START statement is either the data name specified in the key clause or a data item that is subordinate to the key name. Different key values are used for testing. If a key value is provided which matches a record in the file when the START statement is executed, then the record is expected to be made available by the subsequent READ statement. If a key value is provided which does not match any record in the file, then the INVALID KEY path is expected to be taken. The contents of FILESTATUS code are checked after the execution of each START statement.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 56 tests.
- IX210A IX210A tests the use of the START... GREATER THAN... statement using the prime record key and an alternate record key as the key of reference. Included within the START statement is either the data name specified in the key clause or a data item that is subordinate to the key name. Different key values are used for testing. If a key value is provided which matches a record in the file when the START statement is executed, then the record is expected to be made available by the subsequent READ statement. If a key value is provided which does not match any record in the file, then the INVALID KEY path is expected to be taken. The FILE STATUS code is checked after the execution of each START statement.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 39 tests.

IX211A This program tests the capacity to change (update) index keys of records in an Indexed file and retrieve the records from the file in the proper sequence. An Indexed I-O file is created, one of the indexed keys updated with a new key value and then the file is read sequentially. The records of the file are expected to be retrieved in the updated sequence by alternate record key value.

Record modification for the file involves the updating of the update-number field, the record-key or alternate-key, and the ODO-number field of the record. Each time a given record is modified the update-number field will be incremented by one. To keep track of those records modified, the ODO-number field will always carry the sequential location of the record within the file which reflects the last key value position before the record was modified. This location number will be used for verifying that the sequential re-ordering of the record for the file was accomplished successfully. Only one of the three record keys of the record will be modified for any given REWRITE of the record. A test is also made to ensure that the current record pointer is not affected by execution of the REWRITE statement.

X-numbers: basic set plus 24, 44, 55, 62, 82, 83.

This program contains 17 tests.

IX212A This creates a 100 record fixed length indexed file with 10 alternate keys, access mode DYNAMIC. The file is manipulated by the alternate keys using:

1. DELETE
2. READ... NEXT RECORD
3. READ... RECORD KEY
4. REWRITE
5. START

X-numbers: 24, 44, 55, 62, 82, 83.

This program contains 24 tests.

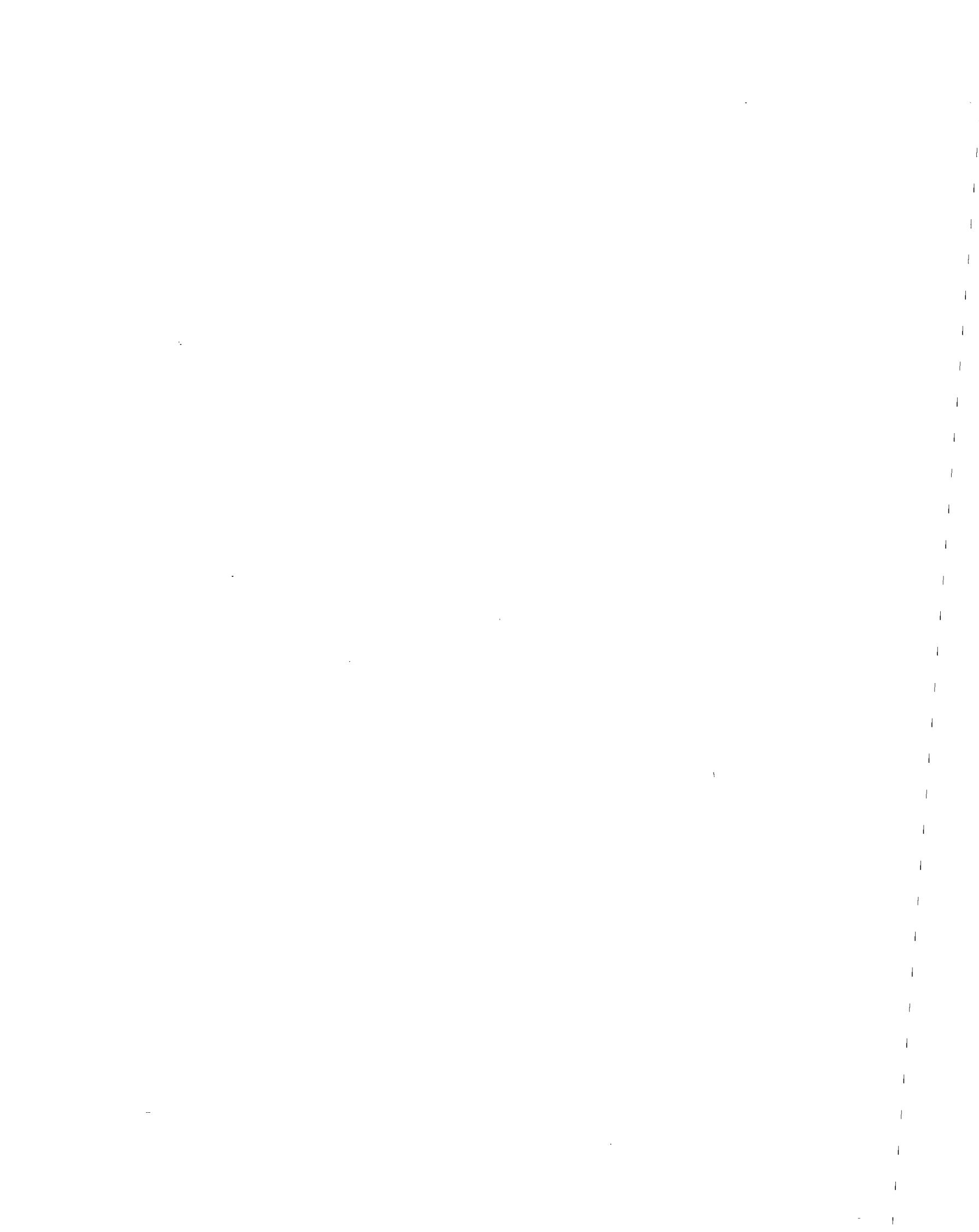
IX213A IX213A creates a 100 record fixed length indexed file with 10 duplicate alternate keys, access mode DYNAMIC. The file is manipulated by the duplicate alternate keys, using:

1. DELETE
2. READ... NEXT RECORD
3. READ... RECORD KEY
4. REWRITE
5. START

X-numbers: 24, 44, 55, 62, 82, 83.

This program contains 21 tests.

- IX214A The START... NOT LESS THAN... statement is tested, using the prime record key and an alternate record key as the key of reference. Included within the START statement is either the data name specified in the key clause or a data item that is subordinate to the key name. If a key value is provided which matches a record in the file when the START statement is executed, then the record is expected to be made available by the subsequent READ statement. Otherwise, the INVALID KEY path is expected to be taken.
- X-numbers: 24, 44, 55, 62, 82, 83.
- This program contains 39 tests.
- IX215A IX215A tests the capability to describe the prime record key and the alternate record keys in a REDEFINES clause, and the use of qualification of the record keys. Three Indexed files are created containing a record key and two alternate keys, with access mode DYNAMIC. The second alternate key uses the WITH DUPLICATES phrase. The statements READ, WRITE, DELETE and REWRITE are used in the testing, but the START statement is used as the test.
- X-numbers: 24, 25, 26, 44, 45, 46, 55, 62, 82, 83.
- This program contains 33 tests.
- IX216A This program creates an indexed file with access mode SEQUENTIAL, and then updates selected records. The FILE STATUS code is checked after each OPEN, CLOSE, READ and REWRITE statement. These statements are used without the appropriate AT END or INVALID KEY phrases.
- X-numbers: 25, 45, 55, 62, 82, 83.
- This program contains 15 tests, of which 1 is deleted.
- IX217A This program creates optional indexed files by the OPEN I-O and OPEN EXTEND statements. The file status code is verified after processing the OPEN statements. The file contents are checked; one file should contain 50 records after execution and the other should contain 25 records of 240 characters and 25 records of 200 characters.
- X-numbers: 24, 25, 44, 45, 55, 82, 83.
- This program contains 6 tests.
- IX218A This program attempts a sequential READ on a non-existent optional indexed file, using the AT END phrase. The START and random READ statements are also executed with the INVALID KEY phrase. In all cases the status code is checked.
- X-numbers: 24, 25, 44, 45, 55, 62, 82, 83.
- This program contains 6 tests.



4.6 NUCLEUS MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION VI

FUNCTION:

The Nucleus module provides a language capability for the internal processing of data within the structure of the four divisions of a program. The Nucleus also provides a capability for defining tables of contiguous data items and accessing an item relative to its position in a table. The Nucleus provides a debugging capability consisting of a compile time switch and debugging lines.

The elements of the Nucleus are divided into two levels. Level 1 supplies elements that perform basic internal operations i.e. the more elementary options of the various clauses and verbs. Level 2 provides elements for more extensive and sophisticated internal processing capabilities.

- NC101A This program tests the Format 1 MULTIPLY statement. The ON SIZE ERROR, NOT ON SIZE ERROR, END-MULTIPLY and ROUNDED phrases are used with various combinations of signed and unsigned literals, DISPLAY and computational fields.
- X-numbers: 55, 82, 83.
- Reference: VI-42, 67, 107, IV-41.
- This program contains 93 tests.
- NC102A This program tests the GO TO, EXIT and PERFORM statements.
- X-numbers: 55, 82, 83.
- Reference: VI-88, 89, 109.
- This program contains 42 tests.
- NC103A NC103A tests the IF statement, including initialisation of items and the NEXT SENTENCE and END-IF phrases.
- X-numbers: 55, 81, 82, 83.
- Reference: VI-90, IV-41.
- This program contains 102 tests.
- NC104A This program tests the MOVE statement.
- X-numbers: 55, 82, 83, 84.
- Reference: VI-103.
- This program contains 141 tests.

- NC105A This program tests the MOVE statement.
 X-numbers: 55, 82, 83, 84.
 Reference: VI-103.
 This program contains 132 tests, of which 3 have been deleted.
- NC106A This program tests the Format 1 SUBTRACT statement with the optional ROUNDED and ON SIZE ERROR and END-SUBTRACT phrases. Truncation of resultant data items is also tested.
 X-numbers: 55, 82, 83, 84.
 Reference: VI-67, 134, IV-40.
 This program contains 126 tests.
- NC107A This program tests the following COBOL features:
 1. Figurative Constants
 2. Continuation Lines
 3. Separators
 4. JUSTIFIED
 5. SYNCHRONIZED
 6. BLANK WHEN ZERO
 7. Long-Names and Literals
 8. REDEFINES
 9. USAGE
 10. VALUE
 11. CURRENCY SIGN IS
 12. DECIMAL-POINT IS COMMA
 13. CONTINUE
 14. Numeric Paragraph-names
 X-numbers: 55, 82, 83.
 Reference: VI-13, 22, 38, 45, 47, 48, 77, IV-10, 42.
 This program contains 177 tests, 5 of which require visual checking.
- NC108M This program tests the following COBOL features:
 1. Compact IDENTIFICATION DIVISION
 2. Switches
 3. Abbreviations
 4. COBOL Character Set
 5. ALPHABET clause
 X-numbers: 51, 55, 82, 83.
 Reference: VI-6, 13, 15, 20, 21, III-3.
 This program contains 14 tests.

NC109M This program tests the ACCEPT and DISPLAY statements.

Special considerations - the tests for the ACCEPT statement process data from the system input device in the following order:

```
Entry 1:ABCDEFHIJKLMNOPQRSTUVWXYZ
Entry 2:0123456789
Entry 3:(.).+-*/$, =
Entry 4:9
Entry 5:0
Entry 6: ABC XYZ
Entry 7:0123456789
Entry 8:single space
Entry 9:""
Entry 10:ABCD
Entry 11:A B C D E F G H I J K L M N O P Q R S T
U V W X Y Z 0123456789
```

X-numbers: 55, 82, 83.

Reference: VI-71, 78.

This program contains 11 tests.

NC110M The PROCEDURE DIVISION in this audit routine consists entirely of paragraph-names and DISPLAY literal statements. A visual inspection of the test results must be made, to check for "pass" or "fail". The entire report should be printed on the system output device. If a copy of the output report cannot be printed on the output device, a visual inspection of the display device must be made to ensure the validity of the tests.

X-numbers: 82, 83.

Reference: VI-78, 79.

This program contains 1 test, the output of which displays on the screen only.

NC111A The routine NC111A tests truncation of resultant data items using ADD, SUBTRACT and MULTIPLY statements with the optional ROUNDED and ON SIZE ERROR phrases.

X-numbers: 55, 82, 83.

Reference: VI-67.

This program contains 7 tests.

- NC112A This program tests the use of multiple operands in ADD, SUBTRACT and MOVE statements.
- X-numbers: 55, 82, 83.
- Reference: VI-73, 103, 134.
- This program contains 32 tests.
- NC113M This program verifies that a statement which must begin in Area A can start in any of the character positions 8 to 11.
- X-numbers: 55, 82, 83.
- Reference: IV-41.
- This program contains 15 tests, all of which require visual inspection.
- NC114M This program tests the use of alphanumeric editing, sequence numbering and comment lines using characters in columns 1-6. The compilation listing should be examined to verify this.
- X-numbers: 55, 82, 83.
- Reference: IV-2, VI-33.
- This program contains 6 tests, 1 of which requires visual inspection.
- NC115A This program tests Formats 1, 2 and 3 of the INSPECT statement.
- X-numbers: 55, 82, 83.
- Reference: VI-94.
- This program contains 31 tests.
- NC116A This program tests the SIGN clause and its optional phrases using the IF and MOVE statements.
- X-numbers: 55, 82, 83.
- Reference: VI-42.
- This program contains 66 tests.
- NC117A This program tests the MULTIPLY and DIVIDE statements, which contain operands defined with the SIGN clause.
- X-numbers: 55, 82, 83.
- Reference: VI-42.
- This program contains 40 tests.

- NC118A This tests the ADD statement with the optional clauses ROUNDED and ON SIZE ERROR. The operands in these tests are defined with the SIGN clause.
- X-numbers: 55, 82, 83.
- Reference: VI-42.
- This program contains 29 tests.
- NC119A The routine NC119A tests the SUBTRACT statement with the optional ROUNDED and ON SIZE ERROR phrases. The operands in these tests are defined with the SIGN clause and truncation of results is tested.
- X-numbers: 55, 82, 83.
- Reference: VI-42.
- This program contains 36 tests.
- NC120A This program tests the MULTIPLY statement using operands defined with the SIGN clause.
- X-numbers: 55, 82, 83.
- Reference: VI-42.
- This program contains 39 tests.
- NC121M This routine tests the MULTIPLY, DIVIDE, PERFORM and DISPLAY statements, using indexed identifiers with one and two levels of indexing as well as relative indexing.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 41 tests, 2 of which require visual inspection.
- NC122A This program tests the INSPECT statement with indexed identifiers. One level of indexing is used as well as relative indexing.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 24 tests.

- NC123A This program tests Formats 1 and 2 of the ADD and SUBTRACT statements and Format 2 of the GO statement. One and two levels of indexing are used as well as relative indexing.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 34 tests.
- NC124A This program tests the use of picture characters P, \$, +, -, Z and *.
- X-numbers: 55, 82, 83.
- Reference: VI-31, 33.
- This program contains 169 tests.
- NC125A The use of picture characters \$ + - * . , with the MOVE, ADD and SUBTRACT statements is tested.
- X-numbers: 55, 82, 83.
- Reference: VI-33, 34.
- This program contains 110 tests.
- NC126A This program tests the use of level numbers 01 to 49 inclusive with a variety of PICTURE clauses and Group and Elementary Item comparisons.
- X-numbers: 55, 82, 83.
- Reference: VI-21.
- This program contains 145 tests.
- NC127A NC127A is written using lower case letters (with the exceptions of some comments, part of the WORKING-STORAGE SECTION and some alphanumeric literals).
- X-numbers: 55, 82, 83.
- Reference: IV-5.
- This program contains 2 tests.

- NC131A Use of the Format 1 SET statement is tested with various combinations of identifiers and index-names being set to identifiers, index names and integers.
X-numbers: 55, 82, 83.
Reference: VI-127.
This program contains 10 tests.
- NC132A The audit routine NC132A tests the use of subscripts to access a single level table using integer, DISPLAY and COMPUTATIONAL fields as subscripts.
X-numbers: 55, 82, 83.
Reference: IV-21.
This program contains 25 tests.
- NC133A The program NC133A tests the use of the SET statement in conjunction with a redefined table. Index-names are SET to numeric integers and USAGE index data items and indices are assigned to other tables.
X-numbers: 55, 82, 83.
Reference: VI-127.
This program contains 25 tests.
- NC134A This program tests accessing of a three dimensional table using numeric literals and data-names as subscripts. Relative subscripting is also tested.
X-numbers: 55, 82, 83.
Reference: IV-21, 22.
This program contains 20 tests.
- NC135A The routine NC135A tests the use of index-names to reference tables that have been redefined, omitting the INDEXED BY clause. Also tested are the SET... UP BY... and SET... DOWN BY... statements.
X-numbers: 55, 82, 83.
Reference: VI-127, IV-21.
This program contains 8 tests, 1 of which requires visual inspection.

- NC136A The routine NC136A is used to verify the accuracy of building and accessing a three dimensional table using various combinations of numeric literal and data-name subscripts.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 8 tests.
- NC137A This program verifies the accuracy of building and accessing a three dimensional table using various combinations of internal indices.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 8 tests.
- NC138A This program tests the use of spaces, commas and left and right parentheses as separators in statements which reference table items.
- X-numbers: 55, 82, 83.
- Reference: IV-4.
- This program contains 36 tests.
- NC139A This program tests the use of numeric literals with relative indexing when accessing two- and three-dimensional tables. The use of indexes and subscripts together is also tested.
- X-numbers: 55, 82, 83.
- Reference: IV-21.
- This program contains 41 tests.
- NC140A The program NC140A tests the use of the SET... UP BY... and SET... DOWN BY... statements using various combinations of positive, negative, zero and unsigned numeric literals and data-names as indexes.
- X-numbers: 55, 82, 83.
- Reference: VI-127.
- This program contains 70 tests.

- NC141A This program tests Formats 1 and 2 of the SET statement, using identifiers indexed by relative indexes and numeric literals.
- X-numbers: 55, 82, 83.
- Reference: VI-127.
- This program contains 9 tests.
- NC170A This program tests the Format 2 MULTIPLY statement. The ON SIZE ERROR and NOT ON SIZE ERROR phrases are used, as is END-MULTIPLY.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 107, IV-41.
- This program contains 96 tests.
- NC171A The routine NC171A tests the Format 1 DIVIDE statement. The ON SIZE ERROR, NOT ON SIZE ERROR and END-DIVIDE phrases are used.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 81, IV-41.
- This program contains 108 tests.
- NC172A NC172A tests the Format 2 DIVIDE statement. The ON SIZE ERROR, NOT ON SIZE ERROR, ROUNDED and END-DIVIDE phrases are used.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 81.
- This program contains 101 tests.
- NC173A This program tests the Format 3 DIVIDE statement. The ON SIZE ERROR, NOT ON SIZE ERROR, ROUNDED and END-DIVIDE phrases are used.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 82, IV-41.
- This program contains 102 tests.

- NC174A This program tests the use of the CLASS clause in the SPECIAL-NAMES paragraph, as well as relational operators and switch settings.
- X-numbers: 51, 52, 55, 81, 82, 83, 90, 91.
- This program contains 77 tests, of which 4 have been deleted.
- NC175A This program tests Format 2 of the SUBTRACT statement. Various combinations of data-items and all optional phrases are tested.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 73, 74, IV-41.
- This program contains 97 tests.
- NC176A NC176A tests Format 1 of the ADD statement. Various combinations of data-items and all optional phrases are tested.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 73, 74, IV-41.
- This program contains 124 tests.
- NC177A NC177A tests Format 2 of the ADD statement. Various combinations of data-items and all optional phrases are tested.
- X-numbers: 55, 82, 83.
- Reference: VI-67, 73, 74, IV-41.
- This program contains 108 tests.
- OBNC1M The program OBNC1M tests Level 1 language elements defined as "obsolete" in the 1985 standard. They must still be tested as they are currently supported but will be dropped from the next COBOL standard.
- X-numbers: 55, 67, 82, 83.
- Reference: VI-6, 11, 75, 89, 130, IV-9.
- This program contains 6 tests.
- NC201A NC201A tests Formats 3 and 4 of the PERFORM statement. The use of the TEST BEFORE and TEST AFTER phrases is tested, with PERFORM...VARYING statements.
- X-numbers: 55, 82, 83.
- Reference: VI-108, 109, 110, 112, 114, 117, 119.
- This program contains 59 tests.

NC202A This program tests the ADD CORRESPONDING statement.

X-numbers: 55, 82, 83.

Reference: VI-73, 74.

This program contains 77 tests.

NC203A This tests Format 4 of the DIVIDE statement.

X-numbers: 55, 82, 83.

Reference: VI-81, 82.

This program contains 57 tests.

NC204M This program tests the General Format of the DISPLAY statements and Format 1 of the ACCEPT statement.

Special Considerations - the tests for the ACCEPT statements process data from the system input device in the following order:

```
Entry  1:ABCDEFGHIJKLMNPQRSTUVWXYZ Z
Entry  2:0123456789
Entry  3:( ).+-*$/;, =
Entry  4:9
Entry  5:0
Entry  6: ABC           XYZ
Entry  7: 9
Entry  8: "
Entry  9:Q
Entry 10:ABCD
Entry 11:ABCD
Entry 12:A B C D E F G H I J K L M N O P Q R S T
          U V W X Y Z 0123456789
Entry 13:D001*002*003*004*005*006*007*008
          *009*010*011*012*013*014*015*016
          *017*018*019*020D021*022*023*024
          *025*026*027*028*029*030*031*032
          *033*034*035*036*037*038*039*040
          D041*042*043*044*045*046*047*048
          *049*050
Entry 14:ABCDEFGHIJ
Entry 15:KLMNOPQRST
```

X-numbers: 55, 56, 57, 82, 83.

Reference: VI-71, 72, 78, 79.

This program contains 15 tests.

- NC205A This program tests continuation of lines.
X-numbers: 55, 82, 83.
Reference: IV-44, VI-2.
This program contains 10 tests.
- NC206A This tests the accessing of elementary items, using Format 1 Qualification with up to 5 levels of Qualifiers. Single dimension tables are also accessed using subscripts qualified to one level.
X-numbers: 55, 82, 83.
Reference: IV-18, 19.
This program contains 53 tests.
- NC207A This program tests ADD, SUBTRACT, MULTIPLY and DIVIDE statements which contain qualified data-names as operands.
X-numbers: 55, 82, 83.
Reference: IV-18, 19, VI-2.
This program contains 85 tests.
- NC208A This program tests Formats 1 and 2 of Qualification, using Formats 1 and 2 of the MOVE statement and Format 2 of the MULTIPLY statement.
X-numbers: 55, 82, 83.
Reference: IV-18, 19, 20.
This program contains 24 tests.
- NC209A The audit routine NC209A continues the tests of the MOVE CORRESPONDING statement and the MOVE statement with multiple receiving areas which were begun in NC208A.
X-numbers: 55, 82, 83.
Reference: VI-102.
This program contains 32 tests.
- NC210A This program tests nested IF statements, using 63 statements and 6 levels of nesting in one sentence, and 22 levels of nesting in a second test.
X-numbers: 55, 82, 83.
Reference: VI-89.
This program contains 85 tests.

- NC211A This tests the General Format of the IF statement, using compound conditional statements with abbreviated conditions, condition-names and qualified data-names. It uses SWITCH-1, which must be set ON at execution time.
- X-numbers: 51, 52, 55, 82, 83.
- Reference: VI-89.
- This program contains 51 tests.
- NC214M This routine tests Format 2 of the ACCEPT statement. It tests the use of a double quote in a literal, logical connectives, and the ACCEPT FROM statement for DATE, DAY, DAY-OF-WEEK and TIME.
- X-numbers: 55, 82, 83.
- Reference: VI-71, 72.
- This program contains 1 test which requires visual inspection.
- NC215A This tests the literal phrase of the ALPHABET clause of the SPECIAL-NAMES paragraph, and the PROGRAM COLLATING SEQUENCE clause of the OBJECT-COMPUTER paragraph.
- X-numbers: 55, 82, 83.
- Reference: VI-11, 13, 15.
- This program contains 7 tests.
- NC216A This program tests the level 2 INSPECT statement features. It uses various combinations of the optional phrases CHARACTERS, ALL, LEADING, FIRST, BEFORE and AFTER.
- X-numbers: 55, 82, 83.
- Reference: VI-93, 94, 95.
- This program contains 57 tests.
- NC217A The program NC217A tests the use of the STRING statement.
- X-numbers: 55, 82, 83.
- Reference: VI-130, 132.
- This program contains 81 tests, of which 1 has been deleted.

- NC218A This program tests the use of the UNSTRING statement.
X-numbers: 55, 82, 83.
Reference: VI-135, 138.
This program contains 125 tests.
- NC219A This audit routine tests the PROGRAM COLLATING SEQUENCE clause and the ALPHABET NAME clause. It tests the use of the HIGH-VALUE and LOW-VALUE figurative constants within the literals for the ALPHABET NAME clause associated with the PROGRAM COLLATING SEQUENCE statement.
X-numbers: 55, 82, 83.
Reference: VI-11, 13, 16.
This program contains 9 tests.
- NC220M This program tests the use of indexed identifiers and qualified data names with Format 1 of the MULTIPLY statement, Formats 1, 4 and 5 of the DIVIDE statement and the General Format of the DISPLAY statement.
X-numbers: 55, 56, 82, 83.
Reference: VI-21, 23.
This program contains 25 tests, 2 of which require visual inspection.
- NC221A The program NC221A tests the use of indexed identifiers with Formats 1, 2 and 3 of the INSPECT statement.
X-numbers: 55, 82, 83.
Reference: VI-21.
This program contains 17 tests.
- NC222A This program tests the use of indexed identifiers with Format 2 of the ADD, SUBTRACT and MOVE statements. De-editing by use of the MOVE statement is also tested.
X-numbers: 55, 82, 83.
Reference: VI-21.
This program contains 8 tests.

- NC223A The INITIALIZE statement is tested using various combinations of optional phrases and a variety of receiving areas.
- X-numbers: 55, 82, 83.
- Reference: VI-91, 92.
- This program contains 94 tests.
- NC224A NC223A tests the use of Reference Modification on a variety of data items, using literals, data names and arithmetic expressions as parameters. Subscripted and qualified data items are also tested.
- X-numbers: 55, 82, 83.
- Reference: IV-22.
- This program contains 14 tests.
- NC225A This program tests the use of the EVALUATE statement. Various combinations of identifiers, literals, arithmetic expressions and conditional expressions as well as the words TRUE and FALSE are used as selection subjects and selection objects. Multiple selection subjects and sets of selection objects are also tested.
- X-numbers: 55, 82, 83.
- Reference: VI-84, 86.
- This program contains 63 tests.
- NC231A This program verifies the accuracy in building and accessing a seven dimensional table that uses indexes and subscripts to reference individual table items. It tests the use of the Format 1 SEARCH statement with VARYING, AT END and END-SEARCH phrases and the Format 1 SET statement.
- X-numbers: 55, 82, 83.
- Reference: VI-2, 121, 122, IV-41.
- This program contains 24 tests.
- NC232A This program verifies the accuracy in building and accessing a three dimensional table using indexing and subscripting to reference individual table items. It tests the use of the Format 1 SEARCH statement with VARYING and AT END phrases.
- X-numbers: 55, 82, 83.
- Reference: VI-121, 122.
- This program contains 17 tests.

- NC233A NC233A verifies accuracy in building and accessing a seven dimensional table that uses indexes and subscripts. The table is defined using the OCCURS, ASCENDING KEY and INDEXED BY clauses. It also tests the use of the Format 2 SEARCH statement with the AT END and END-SEARCH phrases and the Format 1 SET statement.
- X-numbers: 55, 82, 83.
- Reference: VI-2, 121, 122, IV-41.
- This program contains 14 tests.
- NC234A This program verifies accuracy in building and accessing a three dimensional table that uses indexes. The table is defined using the OCCURS and INDEXED BY clauses and is then redefined with a second table. This program tests the use of the Format 1 SEARCH statement with the VARYING and AT END phrases, and the Format 1 SET statement.
- X-numbers: 55, 82, 83.
- Reference: VI-121, 122.
- This program contains 17 tests.
- NC235A This program defines a one dimensional table which has a variable number of occurrences of table items, and then references the items using indexing. The table is defined using the Format 2 OCCURS clause and consists of several condition-name entries. The program tests the use of the Format 1 and 2 SEARCH statements.
- X-numbers: 55, 82, 83.
- Reference: VI-121, 122.
- This program contains 13 tests.
- NC236A This program defines a one dimensional table which redefines an elementary item. The program then tests the use of the Format 1 SET statement and the Format 1 SEARCH statement with VARYING and AT END clauses. An index assigned to one table or an index data item is then varied when searching the second table. A comparison is made to verify that the indexes or the index and index data items are varied properly.
- X-numbers: 55, 82, 83.
- Reference: VI-121, 122.
- This program contains 10 tests.

- NC237A NC237A defines a three dimensional table containing various ascending and descending keys that are referenced by subscripts and indexes which have computational values, or have been set to computational items. Some of the optional words have been eliminated from the syntax.
- X-numbers: 55, 82, 83.
- Reference: VI-121, 122.
- This program contains 13 tests.
- NC238A This program defines a two dimensional table containing various ascending and descending keys and multiple indexes. The program then tests the use of the Format 1 and Format 2 SEARCH statements, with and without multiple conditions in the WHEN phrase, to reference the table.
- X-numbers: 55, 82, 83.
- Reference: VI-51, 121, 122.
- This program contains 10 tests.
- NC239A This defines a three dimensional table and tests the referencing of individual table items using indexing. Values are verified using the IF statement.
- X-numbers: 55, 82, 83.
- Reference: IV-21, II-15.
- This program contains 8 tests.
- NC240A This program defines a three dimensional table and tests the referencing of individual table items using signed and unsigned numeric items as subscripts. The contents of table elements are verified using the Format 4 PERFORM statement.
- X-numbers: 55, 82, 83.
- Reference: VI-109, IV-21, II-14.
- This program contains 11 tests.
- NC241A This program defines a three dimensional table and tests the referencing of individual table items using indexing. The contents of table elements are verified using the Format 4 PERFORM statement.
- X-numbers: 55, 82, 83.
- Reference: VI-2, 109, IV-21, II-15.
- This program contains 11 tests.

- NC242A This program defines three and seven dimensional tables and tests the referencing of individual table items using indexing and subscripting.
X-numbers: 55, 82, 83.
Reference: VI-2, 109, IV-21, II-14.
This program contains 12 tests.
- NC243A This program defines three and seven dimensional tables, and tests the referencing of individual table items using indexing and subscripting.
X-numbers: 55, 82, 83.
Reference: VI-2, 109, IV-21, II-14.
This program contains 16 tests.
- NC244A This routine defines a two dimensional table containing the OCCURS clause with the INDEXED BY series clause. The program tests relative addressing and various combinations of the Format 2 SET statement.
X-numbers: 55, 82, 83.
Reference: VI-126, IV-21, II-15.
This program contains 6 tests.
- NC245A This program defines two and three dimensional tables, and tests the use of semicolon, comma and spaces as separators in referencing table items via subscripting and indexing.
X-numbers: 55, 82, 83.
Reference: IV-4, 21.
This program contains 28 tests.
- NC246A This routine defines three and seven dimensional tables which must be qualified in order to reference the table items. The use of qualified subscripts in referencing both unqualified and qualified table items is tested. Additionally, there are conditional statements which reference qualified condition-names with both unqualified and qualified subscripts.
X-numbers: 55, 82, 83.
Reference: VI-2, IV-18, 19, 21.
This program contains 49 tests.

- NC247A The program NC247A tests various verbs acting on WORKING-STORAGE records containing subordinate entries, which in turn contain Format 2 OCCURS clauses; ie, ...OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1 ... The tests ascertain whether the compiler treats the number of table occurrences as variable depending on the current value of the data-name-1 identifier. Operations tested include IF, INSPECT, MOVE, SEARCH, STRING and UNSTRING.
- X-numbers: 55, 82, 83.
- Reference: VI-26, 28.
- This program contains 21 tests, of which 1 has been deleted.
- NC248A NC248A tests the SET statement using qualification with indexed and relative indexed identifiers.
- X-numbers: 55, 82, 83.
- Reference: VI-126, 128.
- This program contains 11 tests.
- NC250A This tests the General Format of the IF statement using figurative constant operands, unequal length operands, condition-name conditions, sign conditions, class conditions, arithmetic-expressions operands, abbreviated conditions and relation conditions..
- X-numbers: 55, 82, 83.
- Reference: VI-51, 58, 90.
- This program contains 115 tests.
- NC251A This program tests Format 5 of the DIVIDE statement. The ON SIZE ERROR, NOT ON SIZE ERROR phrases and the use of subscripted remainders are tested.
- X-numbers: 55, 82, 83.
- Reference: VI-82.
- This program contains 59 tests.
- NC252A This routine tests the REDEFINES and RENAMES clauses of the Data Description entries in the DATA DIVISION and the COMPUTE statement. Use of numeric data names is also tested.
- X-numbers: 55, 82, 83.
- Reference: VI-21, 40, 76.
- This program contains 75 tests.

- NC253A This program tests Format 3 of the SUBTRACT statement; the SUBTRACT CORRESPONDING is used and SUBTRACT statements with multiple resultant data items in the GIVING phrase are also tested.
- X-numbers: 55, 82, 83.
- Reference: VI-133, 134.
- This program contains 61 tests.
- NC254A This program tests the use of the CLASS clause in the SPECIAL-NAMES paragraph, as well as switch settings using Level 2 operators AND, OR and NOT.
- X-numbers: 51, 52, 55, 81, 82, 83, 90, 91.
- This program contains 9 tests.
- OBNC2M This program contains tests of Level 2 language elements defined as "obsolete" in the 1985 standard. These elements must still be tested as they are currently supported, but will be dropped from the next COBOL standard.
- X-numbers: 55, 82, 83.
- This program contains 16 tests.

4.7 RELATIVE I-O MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION VIII

FUNCTION:

The Relative I-O module provides a capability to access records of a mass storage file in either a random or sequential manner. Each record in a relative file is uniquely identified by an integer value greater than zero which specifies the record's logical ordinal position in the file. The elements of the Relative I-O module are divided into two levels. Level 1 provides elements for the basic facilities of definition and access of relative files. Level 2 provides elements for more complete facilities, including the capability of accessing the file both randomly and sequentially in the same COBOL program.

RL101A This program creates and verifies a 500 record fixed length relative file, whose access mode is sequential and record size is 120 characters. On completion of this program the file is passed to RL102A for further processing.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 1 test.

RL102A This program verifies and updates the 500 record fixed length relative file created in RL101A. The access mode for the file in this program is random. After the file is verified for accuracy of data, it is opened as I-O and every fifth record is updated. The file is then verified again for correctness. Upon completion of the program the file is passed to RL103A for further processing.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 11 tests.

RL103A RL103A verifies the 500 record fixed length file updated in RL102A. The access mode for the file in this program is sequential. After the file is verified for accuracy, it is opened as I-O and every fourth record is deleted. Following this the file is again verified for correctness. There should be 375 records remaining in the file.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 11 tests.

- RL104A This program creates and updates a 500 record fixed length relative file whose access mode is sequential. The record size is 120 characters. After the file is created, it is opened as I-O and every fifth record is updated. All READ, WRITE and REWRITE statements are used without the AT END or INVALID KEY phrases. A USE procedure for file-name is specified and the contents of the FILE STATUS data item are tested following each I-O operation.
- X-numbers: 22, 55, 69, 74, 76, 82, 83.
- This program contains 12 tests.
- RL105A This program creates and verifies three fixed length record relative files whose access mode is sequential. The three files contain 19, 16 and 16 records respectively. The record size for all files is 100 characters.
- X-numbers: 21, 22, 23, 55, 69, 74, 75, 76, 77, 82, 83.
- This program contains 4 tests.
- RL106A This program creates and verifies three variable length record relative files, whose access mode is random. The three files contain 18, 10 and 12 records respectively, and their record lengths are 56, 100 or 102 characters. The RELATIVE KEY data item is used to access specific records within each of the three files and also to verify their accuracy.
- X-numbers: 21, 22, 23, 55, 69, 74, 75, 76, 77, 82, 83.
- This program contains 4 tests.
- RL107A This program creates and verifies two fixed length relative files whose access mode is random. One file contains 125 records and the other file contains 25 records. The record size for both files is 120 characters. The first (or larger) file is only partially created during the OUTPUT mode, but is subsequently completed in the I-O mode. The file is not created in sequential order, but the end result is that there are no null records in the file. Various records in both files are tested for accuracy, while using the RELATIVE KEY data items to access the specific records to be tested. The first file is tested while opened for I-O, and the second file is tested while opened for INPUT. A test is also made to read a record which has yet to be created.
- X-numbers: 21, 22, 55, 69, 74, 75, 76, 82, 83.
- This program contains 19 tests.

RL108A This program creates and verifies a 500 record fixed length relative file, whose access mode is sequential and record size is 120 characters. On completion of this program, the file is passed to RL109A for further processing. The use of the following elements is tested:

1. ASSIGN
2. ORGANIZATION
3. ACCESS
4. READ
5. WRITE

X-numbers: 55, 61, 69, 74, 75, 82, 83.

Reference: VIII-8, 10, 26, 27, 29, 37, 38.

This program contains 1 test.

RL109A This program verifies the 500 record fixed length relative file created in RL108A. Access mode is random. The file is opened as I-O, selected records are updated and the file is verified again for correctness. On completion of the program the file is passed to RL110A for further processing. The following language elements are tested:

1. ORGANIZATION
2. ACCESS
3. READ
4. REWRITE
5. INVALID KEY clause

X-numbers: 55, 61, 69, 74, 75, 82, 83.

Reference: VIII-2, 8, 10, 29, 30, 31.

This program contains 11 tests.

RL110A RL110A verifies the 500 record fixed length file updated in RL109A. Access mode is sequential. The file is opened as I-O, selected records deleted and each record checked again for accuracy.

X-numbers: 55, 61, 69, 74, 75, 82, 83.

Reference: VIII-2, 19.

This program contains 10 tests.

RL111A RL111A creates and updates a 500 record fixed length relative file, record size 120 characters, access mode sequential. The file is opened as I-O and selected records are updated. The FILE STATUS code is tested for accuracy, for each OPEN, CLOSE, READ and REWRITE statement. The READ, WRITE and REWRITE statements are used with the AT END, NOT AT END, INVALID KEY and NOT INVALID KEY phrases.

NOTE: For this test, RL-FS2 and RL-FS3 MUST reference the same physical file. The value of which is given the x-card 22. (X- 22).

X-numbers: 22, 55, 69, 74, 76, 82, 83.

Reference: VIII-8, 10, 26, 28, 29, 30, 31, 37, 38.

This program contains 24 tests.

RL112A RL112A creates and updates a 500 record fixed length relative file with access mode random and record size 120 characters. The USE statement is tested.

X-numbers: 22, 55, 69, 74, 76, 82, 83.

Reference: VIII-36.

This program contains 12 tests.

RL113A This program tests the application of the USE statement. A 500 record fixed length relative file with 120-character records is created, and accessed randomly.

X-numbers: 22, 55, 69, 74, 76, 82, 83.

Reference: VIII-35, 36.

This program contains 11 tests.

RL114A This program tests the application of the USE statement. A 500 record fixed length relative file with 120-character records is created, and accessed randomly.

X-numbers: 22, 55, 69, 74, 76, 82, 83.

Reference: VIII-35, 36.

This program contains 13 tests.

- RL115A This program tests the application of the USE statement. A 500 record fixed length relative file with 120-character records is created, and accessed randomly.
- X-numbers: 22, 55, 69, 74, 76, 82, 83.
- Reference: VIII-35, 36.
- This program contains 13 tests.
- RL116A This tests the application of the FILE STATUS clause. A 500 record fixed length relative file with 120-character records is created, and accessed randomly. The file status is checked, following the execution of an I-O statement and a READ statement where the length of the record being processed does not conform to the fixed file attributes for the file.
- X-numbers: 22, 55, 69, 74, 76, 82, 83.
- Reference: VIII-3.
- This program contains 3 tests.
- RL117A This program tests the syntactical constructs and semantic actions associated with the FILE STATUS clause. Two 500 record fixed length relative files with a record size of 120 characters are created, and are accessed randomly. The status is checked for each file where an attempt has been made to read a record when the end of the file has been reached; also where a READ statement is attempted, where the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file; also where a READ statement is attempted and the AT END condition already exists.
- X-numbers: 22, 55, 69, 74, 76, 82, 83.
- Reference: VIII-3.
- This program contains 8 tests, 2 of which have been deleted.

RL118A This program tests the syntactical constructs and semantic actions associated with the FILE STATUS clause. Two 500 record fixed length relative files with a record size of 120 characters are created, and accessed randomly. For each file the value of the status is checked for the following circumstances:

1. An attempt is made to write a record that would create a duplicate key in a relative file.
2. An attempt is made to randomly access a record that does not exist in the file.
3. An attempt is made to write beyond the externally defined boundaries of a relative file.
4. A sequential WRITE statement is attempted for a relative file where the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file.

X-numbers: 22, 55, 69, 74, 76, 82, 83.

Reference: VIII-3.

This program contains 4 tests, 2 of which have been deleted.

RL119A RL119A tests the use of the FILE STATUS clause. Four 500 record fixed length relative files are created, with a record size of 120 characters and random access. The value of the status code is checked where a permanent error exists because an OPEN statement with the INPUT or I-O phrase is attempted on a file that is not present.

X-numbers: 55, 69, 74, 76, 82, 83, 92.

Reference: VIII-4.

This program contains 1 test.

RL201A RL201A creates and verifies a 500 record fixed length relative file whose access mode is sequential. The record size is 120 characters. Upon completion of this program the relative file is passed to RL202A for further processing.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 1 test.

RL202A This program is used to verify and update the 500 record fixed length relative file created in RL201A. The access mode is dynamic. After the file is verified for accuracy of data, it is opened as I-O and every fifth record is updated. The file is then verified again. The RELATIVE KEY data item is used to access the next record to be read or updated. The verification of the initial input file is performed in ascending sequence and the verification of the updated file is performed in descending sequence. Upon completion of this program the updated relative file is passed to RL203A for further processing.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 11 tests.

RL203A This program is used to verify the 500 record fixed length relative file updated in RL202A. The access mode is dynamic. After the file is verified for accuracy of data, it is opened as I-O and every fourth record is deleted. Following this operation, the file is again verified for correctness. There should be 375 records remaining in the file. All read operations are performed using the READ NEXT RECORD statement.

X-numbers: 21, 55, 69, 74, 75, 82, 83.

This program contains 11 tests.

RL204A This program creates and updates a 500 record fixed length relative file whose access mode is dynamic. The record size is 120 characters. After the file is created, it is opened as I-O and every fifth record is updated. All READ, WRITE and REWRITE statements are used without the appropriate AT END or INVALID KEY phrases. A USE procedure for file-name is specified, and the contents of the FILE STATUS data item are tested following each I-O operation.

X-numbers: 22, 55, 69, 74, 76, 82, 83.

This program contains 12 tests.

RL205A This program creates and verifies two 300 record fixed length relative files. One file is created with access mode dynamic and the other with access mode sequential. The record length for both files is 240 characters. This program is designed to test the use of the READ and START statements. Various syntactical constructs of the READ statement are used to sequentially and randomly verify specific records on the dynamic accessed file. Also, both files are processed sequentially using various syntactical constructs of the START statement to logically position within the file.

X-numbers: 21, 22, 55, 56, 69, 74, 75, 76, 82, 83.

Reference: VIII-34.

This program contains 67 tests, 1 of which has been deleted.

- RL206A This program creates a relative file, sequentially, with 500 variable length records, and record size 120-140 characters, and verifies that it was created correctly. On completion of this program the file is passed to RL207A for further processing.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-14, 26, 27, 30, 31.
- This program contains 501 tests.
- RL207A This program verifies the file created in RL206A, and then records of different size are selectively updated. Each record is again verified and the file is passed to RL208A for further processing.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-14, 30, 31.
- This program contains 20 tests.
- RL208A This program verifies the file updated in RL207A and then selectively deletes certain records before verifying the file again.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-14, 19.
- This program contains 11 tests.
- RL209A This program creates a relative file, sequentially, with 500 variable length records, and record size 120-140 characters, and verifies that it was created correctly. The RECORD IS VARYING clause is tested.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-3, 14, 31.
- This program contains 1 test.
- RL210A This program creates a relative file, sequentially, with 500 variable length records, and record size 120-140 characters, and verifies that it was created correctly. The RECORD IS VARYING clause is tested.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-14, 31.
- This program contains 1 test.

- RL211A This program creates a relative file, sequentially, containing 500 variable length records with record sizes of 120-140 characters, and verifies that it was created correctly. The RECORD IS VARYING clause is tested.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- Reference: VIII-14, 31.
- This program contains 501 tests.
- RL212A This program creates a relative file sequentially (ACCESS MODE IS SEQUENTIAL), containing 500 fixed length records with a record size of 120 characters. On completion the file is passed on to RL213A for processing.
- X-numbers: 21, 55, 69, 74, 75, 82, 83.
- This program contains 1 test.
- RL213A This program processes the relative I-O file created in RL212A. The access mode is sequential. The file is opened in EXTEND mode, updated, and the accuracy of each record verified.
- X-numbers: 21, 22, 55, 69, 74, 75, 82, 83.
- Reference: VIII-8, 9, 21, 24.
- This program contains 521 tests.

4.8 REPORT WRITER MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION XIII

FUNCTION:

The Report Writer module provides a facility for producing semi-automatic printed reports by specifying the physical appearance of a report rather than requiring specification of the detailed procedure necessary to produce that report. A hierarchy of levels is used in defining the logical organisation of a report. Each report is divided into report groups, which in turn are divided into sequences of items. Such a hierarchical structure permits explicit reference to a report group with implicit reference to other levels in the hierarchy. A report group contains one or more items to be presented on zero, one or more lines. The Report Writer module contains only level 1 elements.

RW101A The routine RW101A tests basic Report Writer module functions. The report description in this routine contains a PAGE LIMIT IS 20 LINES without the optional HEADING, FIRST DETAIL, LAST DETAIL or FOOTING phrases. A single Detail Report Group is defined for the report. An output report in the usual audit routine format is produced using the WRITE statements, and a one-page report of 20 lines is produced by the RWCS. The sequential file RW-FS1 is the report file and is assigned to the RWCS output device through X-number 49.

X-numbers: 49, 55, 69, 74, 75, 82, 83, 84.

This program contains 8 tests.

RW102A This routine tests basic Report Writer module functions. The report description in this program contains PAGE LIMIT 20, FIRST DETAIL 1 and LAST DETAIL 25 without the optional HEADING or FOOTING phrases. A single Detail Report Group is defined for the report. An output report in the usual audit routine format is produced using WRITE statements, and a one page report of 20 lines is produced by the RWCS. The sequential file RW-FS2 is the report file and is assigned to the RWCS output device through X-number 49.

X-numbers: 49, 55, 69, 74, 75, 82, 83, 84.

This program contains 4 tests.

RW103A The routine RW103A tests basic Report Writer module functions. The report description in this program contains PAGE LIMIT 30, HEADING 1, FIRST DETAIL 6 and LAST DETAIL 25 without the optional FOOTING phrase. A Page Heading Report Group and a Detail Report Group are defined for the report. An output report in the usual audit routine format is produced using WRITE statements, and a three page report of 20 lines is produced by the RWCS. The sequential file RW-FS3 is the report file, and is assigned to the RWCS output device through X-number 49.

X-numbers: 49, 55, 69, 74, 75, 82, 83.

This program contains 14 tests.

RW104A This routine tests basic Report Writer module functions. The report description in this program contains PAGE LIMITS ARE 30 LINES, HEADING 1, FIRST DETAIL 1, LAST DETAIL 25 and FOOTING 29. A Page Heading, Page Footing and one Detail Report Group are defined for the report. An output report in the usual audit routine format is produced using WRITE statements, and a three page report with one page heading line, one page footing line and 20 detail lines is produced by the RWCS. The sequential file RW-FS4 is the report file and is assigned to the RWCS output device through X-number 49.

X-numbers: 49, 55, 69, 74, 75, 82, 83, 84.

This program contains 14 tests.

4.9 SEGMENTATION MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION XVI

FUNCTION:

The Segmentation module provides a means by which the user may communicate with the compiler to specify the overlaying at object time of Procedure Division sections. The elements of the Segmentation module are divided into two levels. Level 1 provides for section segment-numbers and fixed segment limits. Level 2 adds the capability for varying the segment limit.

SG101A This program exercises transfer of control, via the PERFORM statement, to various fixed permanent segments and independent segments from fixed permanent segments. Control is also passed from independent segments to fixed permanent segments. There are 50 fixed permanent segments and 50 independent segments. There is no attempt to test either the last used or the initial state in this program.

X-numbers: 55, 82, 83, 84.

This program contains 151 tests.

SG102A This program exercises various ALTER and PERFORM statements and prepares a directory in each test to trace program flow. Transfers of control are from fixed permanent segments to fixed permanent segments, fixed permanent segments to independent segments, independent segments to fixed permanent segments and independent segments to independent segments. There are no tests for the initial state per se, but there are tests to ensure that the last used state is present when required.

X-numbers: 55, 82, 83, 84.

This program contains 8 tests.

SG103A The program SG103A exercises various ALTER, PERFORM and GO TO statements to check initial and last used segment states. Transfers of control are from fixed permanent segments to fixed permanent segments, fixed permanent segments to independent segments, independent segments to fixed permanent segments and independent segments to independent segments.

Control is passed through the use of PERFORM and GO TO statements, and fall-through logic.

X-numbers: 55, 82, 83, 84.

This program contains 7 tests.

- SG104A The program SG104A tests the compiler's ability to segment a program, using a SORT statement which requires both an INPUT PROCEDURE and an OUTPUT PROCEDURE. The sections containing the SORT statement, the INPUT PROCEDURE and the OUTPUT PROCEDURE are all in the same independent segment. The results for SG104A are identical to the unsegmented ST108A.
- X-numbers: 27, 55, 82, 83, 84.
- This program contains 9 tests.
- SG105A SG105A tests the compiler's ability to segment a program SORT statement which requires both an INPUT PROCEDURE and an OUTPUT PROCEDURE. The section containing the SORT statement is in an independent segment, while the INPUT and OUTPUT PROCEDURES are located in fixed permanent segments. The results are identical to those expected for ST108A.
- X-numbers: 27, 55, 82, 83, 84.
- This program contains 9 tests.
- SG106A This tests the compiler's ability to segment using a SORT statement which requires both an INPUT PROCEDURE and an OUTPUT PROCEDURE. The section containing the SORT statement is in a fixed permanent segment while the INPUT and OUTPUT PROCEDURES are located in different independent segments. The results are identical to those expected for ST108A.
- X-numbers: 27, 55, 82, 83, 84.
- This program contains 9 tests.
- SG201A This program tests the initial state of independent segments and the last used state of fixed overlayable segments. A SEGMENT-LIMIT of 30 is specified in the OBJECT-COMPUTER paragraph, thereby causing segments from 30 to 49 to become fixed overlayable segments. The first section encountered in the PROCEDURE DIVISION is an independent segment. There are 100 segments present in this program.
- X-numbers: 55, 82, 83, 84.
- This program contains 79 tests.
- SG202A This program contains tests designed to alter fixed overlayable segments that have not yet been called for execution, to test code which falls through to independent segments, and to PERFORM fixed overlayable segments. Due to a SEGMENT-LIMIT OF 25, segments 25 to 49 are considered fixed overlayable segments. This program contains no independent segments.
- X-numbers: 55, 82, 83, 84.
- This program contains 5 tests.

SG203A This program tests the initial state for independent segments and the last used state for fixed overlayable segments. The PERFORM and GO TO statements, and the code which falls through to an independent segment are used to determine whether the initial state or last used state is found in each segment.

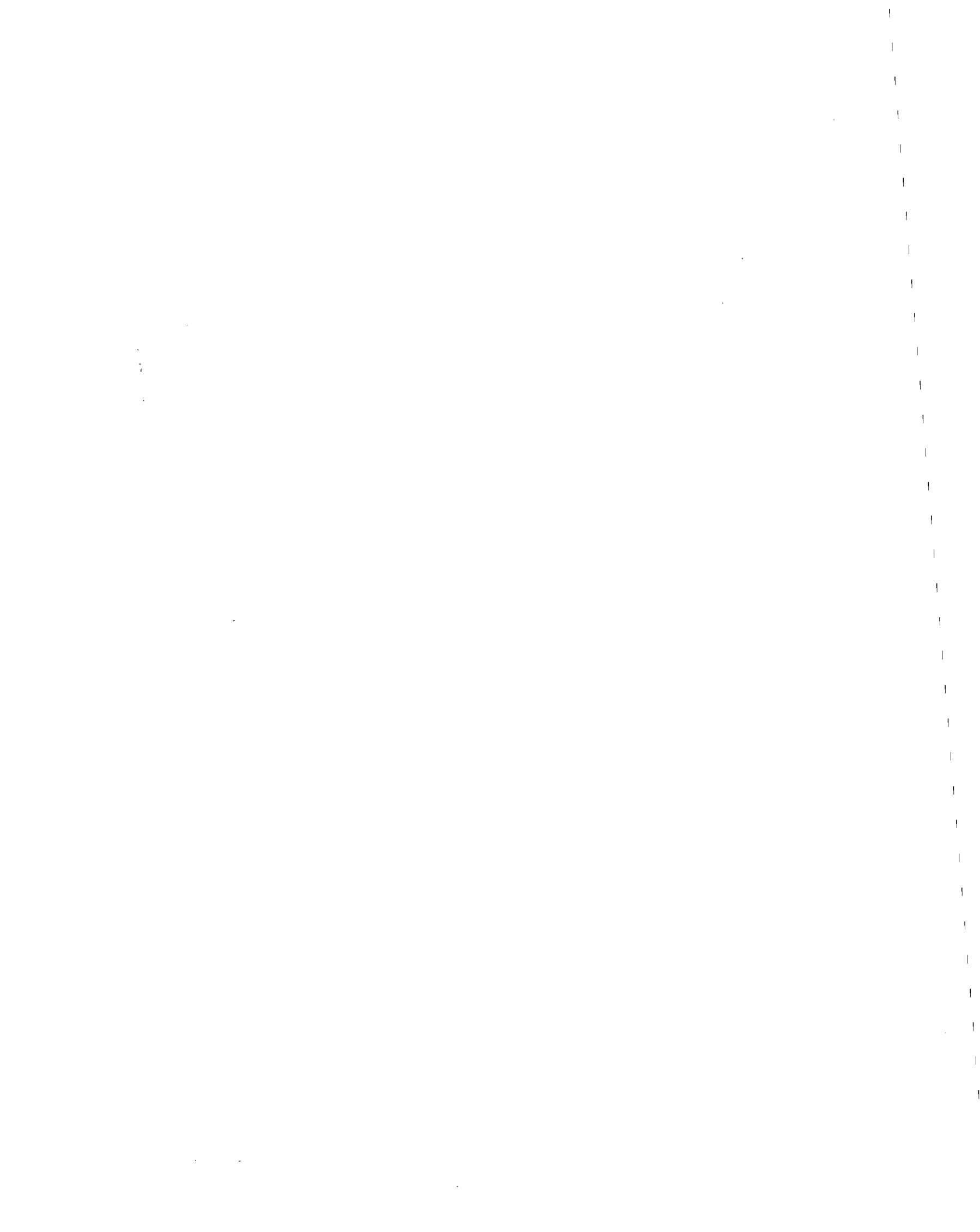
X-numbers: 55, 82, 83, 84.

This program contains 18 tests.

SG204A This program tests the compiler's ability to segment a program using three SORT statements consisting of a variety of INPUT and OUTPUT PROCEDUREs, as well as the USING and GIVING phrases. The SORT statements and all related INPUT and OUTPUT PROCEDUREs are located in fixed permanent segments, fixed overlayable segments and independent segments. The results are identical to those expected for the unsegmented ST131A.

X-numbers: 01, 14, 15, 27, 28, 29, 55, 74, 75, 76, 77, 82, 83, 84.

This program contains 15 tests.



4.10 SOURCE TEXT MANIPULATION MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION XII

FUNCTION:

The Source Text Manipulation module contains the language elements for the insertion and replacement of source program text as part of the compilation of the source program. The elements of the Source Text Manipulation module are divided into two levels. Level 1 provides the facility for copying text from a single library into the source program. Level 2 provides the additional capability of replacing library text during the copying process, specifying more than one COBOL library at compile time, and replacing source program text.

SM101A This program tests the use of the COPY statement in a File Description with its related 01 entries, in the WORKING-STORAGE SECTION and in the PROCEDURE DIVISION. It creates a sequential file which is input to SM102A to check the proper execution of the COPY statements in SM101A. It also tests the effect of a COPY statement appearing on a debugging line.

X-numbers: 01, 55, 82, 83.

Reference: XII-2, 3, 5.

This program contains 8 tests.

SM102A This tests the file produced by SM101A to ensure the proper execution of the COPY statement.

X-numbers: 01, 55, 69, 74, 75, 82, 83.

Reference: XII-2.

This program contains 4 tests.

SM103A Program SM103A tests the use of the COPY statement in the IDENTIFICATION DIVISION and ENVIRONMENT DIVISION (SOURCE-COMPUTER, OBJECT-COMPUTER, SPECIAL-NAMES, FILE-CONTROL and I-O CONTROL entries). A sequential file is produced which is read and checked by SM104A. The maximum and minimum lengths of a library text word are also tested. Note that as the SELECT ... ASSIGN statement for the print file is in the copy library file, if 'JJJJ' is used in the X-card to substitute the program name, the print file will be called K3FCA.

X-numbers: 69, 74, 75, 76.

Reference: XII-2, 5.

This program contains 6 tests.

- SM104A This program reads and checks the file produced by SM103A to verify the proper execution of the COPY statement in that program.
- X-numbers: 01, 55, 69, 74, 75, 82, 83.
- Reference: XII-2.
- This program contains 7 tests.
- SM105A This program tests the use of the COPY statement in the DATA DIVISION for a Sort Description entry and the associated record description entries.
- X-numbers: 01, 27, 55, 69, 74, 75, 82, 83.
- Reference: XII-2.
- This program contains 9 tests.
- SM106A This program contains a single COPY statement to copy the ENVIRONMENT, DATA and PROCEDURE DIVISIONS into the program.
- X-numbers:
- Reference: XII-2, 3.
- This program contains 1 test which requires inspection.
- SM107A This program tests for the ability to copy 1599 records by a single COPY statement in the PROCEDURE DIVISION.
- X-numbers: 55, 82, 83.
- Reference: XII-2.
- This program contains 200 tests.
- SM201A Program SM201A tests the REPLACING phrase of the COPY statement in the WORKING-STORAGE SECTION and PROCEDURE DIVISION, and produces a sequential output file, using COPYed code, which is subsequently checked by program SM202A.
- X-numbers: 01, 55, 82, 83.
- Reference: XII-2.
- This program contains 11 tests.

- SM202A This program reads the file produced by SM201A to verify the proper execution of the COPY REPLACING statements in SM201A. A number of further tests using various numeric and alphanumeric literals, qualified data names and multiple REPLACING operands are also carried out.
- X-numbers: 01, 55, 69, 74, 75, 82, 83.
- Reference: XII-2.
- This program contains 7 tests.
- SM203A This program tests the use of the COPY statement REPLACING phrase in the ENVIRONMENT DIVISION. A sequential file is produced using COPYed text and this is checked in program SM204A.
- X-numbers: 69, 74, 77, 82, 83.
- Reference: XII-2.
- This program contains 1 test.
- SM204A This program checks the file produced by program SM203A to verify the proper execution of the COPYed statements in that program's ENVIRONMENT DIVISION.
- X-numbers: 02, 55, 69, 74, 77, 82, 83.
- Reference: XII-2.
- This program contains 4 tests.
- SM205A This program tests the use of the COPY statement, with its REPLACING phrase, for a Sort Description and related entries. (This program assumes that program ST101A performs correctly.)
- X-numbers: 01, 27, 55, 69, 74, 76, 82, 83.
- Reference: XII-2.
- This program contains 9 tests.
- SM206A This program tests the REPLACING phrase of the COPY statement using a variety of pseudo-text operands. Maximum and minimum length text words are also tested.
- X-numbers: 55, 82, 83.
- Reference: XII-2, 5.
- This program contains 18 tests, of which 2 have been deleted.

SM207A Program SM207A tests the COPY statement using two different library names to qualify the same text name.

X-numbers: 47, 48, 55, 82, 83.

Reference: XII-2.

This program contains 2 tests.

SM208A This program tests Formats 1 and 2 of the REPLACE statement with various combinations of pseudo-text.

X-numbers: 55, 82, 83.

Reference: XII-6, 7, 8.

This program contains 10 tests, 1 of which has been deleted.

4.11 SEQUENTIAL I-O MODULE

ANSI DOCUMENT REFERENCE X3.23-1985 SECTION VII

FUNCTION:

The Sequential I-O module provides a capability to access records of a file in established sequence. The sequence is established as a result of writing the records to the file. The elements of the Sequential I-O module are divided into two levels. Level 1 provides elements for the basic facilities of definition and access of sequential files. Level 2 provides elements for the complete facilities of definition and access of sequential files.

A number of SQ test programs are applicable to both magnetic tape and mass storage files. Each of these programs is annotated with an "*" next to the program name. If support for both magnetic tape and mass storage is claimed, the identified programs must be run twice - once on each media. Otherwise, run the program on whichever media is supported.

SQ101M The program SQ101M tests all Level 1 combinations of the WRITE statement for a printer output file, as well as the line overprint capability. It also tests that column 1 does not act as a carriage control field. Use of the FROM and ADVANCING phrases with identifier-2 is also tested.

X-numbers: 55, 62, 82, 83, 84.

This program contains 57 tests.

SQ102A* This program creates a magnetic tape file containing 750 fixed length records. The record length is 120 characters. The file is verified and then READ using all four variants of the READ statement that can be produced by including or omitting the optional words "RECORD" and "AT". The program omits the optional words "ORGANIZATION IS" from the "ORGANIZATION IS SEQUENTIAL" clause of the file control entry, and places the ASSIGN clause in a position other than the first in the same entry.

X-numbers: 01, 55, 62, 82, 83, 84.

This program contains 11 tests.

SQ103A* The program SQ103A creates and then verifies a magnetic tape file containing 500 fixed length records of 120 characters. The File-Control entry contains a File Status clause, and a USE statement defines the procedures to be executed when an end-of-file is encountered. The first hundred records are read using a READ statement without the AT END phrase, and the remainder without it. The tape file is passed three times; ie, rewound twice.

X-numbers: 01, 55, 62, 82, 83, 84.

This program contains 30 tests.

- SQ104A This program creates a mass storage file of 649 fixed length records of 120 characters. The file is then verified and the OPEN, CLOSE, READ and WRITE statements are tested for Level 1 features.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 11 tests.
- SQ105A The routine SQ105A creates a mass storage file containing 980 fixed length records of 125 characters. The file is then verified. There are two USE procedures in the DECLARATIVE SECTION; one for exceptions on output, and one for exceptions on input. The AT END phrase is not included in any READ statement, so the end-of-file condition should cause execution of the USE procedure for exception on input.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 22 tests.
- SQ106A* This program creates and verifies a magnetic tape file containing 450 variable length records of 120 or 151 characters. The OPEN, CLOSE, READ and WRITE statements are tested for Level 1 features. The tape file is passed three times (read or written); ie, rewound twice.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 75 tests, of which 6 have been deleted.
- SQ107A The routine SQ107A creates a mass storage file containing 450 variable length records. The records are either 120 or 151 characters in length. The file is then verified and the OPEN, CLOSE, READ and WRITE statements are tested for Level 1 features.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 6 tests.
- SQ108A This program creates and verifies a mass storage file containing 710 fixed length records of 141 characters. The various forms of the READ INTO statement are tested.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 8 tests.

SQ109M* This routine creates and verifies a multi-reel magnetic tape file containing 750 fixed length records of 120 characters. The CLOSE REEL option is used to create a two reel file. There are 325 records on the first reel and 425 records on the second. The two reel tape file is passed three times (read or written); ie, rewound twice.

Special considerations:

1. A two reel magnetic tape file is required if allowed by the operating system.
2. The H-option is used to include all CLOSE REEL statements in the source program when a multi-reel file is specified.
3. The I-option is used to delete all CLOSE REEL statements from the source program when a multi-reel file is not specified.

X-numbers: 06, 55, 62, 82, 83, 84.

This program contains 6 tests.

SQ110M This program creates and verifies a multi-unit mass storage file containing 649 fixed length records of 120 characters. The CLOSE UNIT option is used to create a two unit file. There are 196 records on the first unit and 453 records on the second unit.

Special considerations:

1. A two unit mass storage file is required if allowed by the operating system.
2. The E-option is used to include all CLOSE UNIT statements in the source program when a multi-unit file is specified.
3. The F-option is used to delete all CLOSE UNIT statements from the source program when a multi-unit file is not specified.

X-numbers: 19, 55, 62, 82, 83, 84.

This program contains 6 tests.

- SQ111A* This program creates and verifies a magnetic tape file which includes a CODE-SET clause in the File Description entry. The file contains 595 fixed length records of 156 characters per record. The tape file is passed twice (read or written); ie, rewound once. The following COBOL features are included in this program:
- SPECIAL NAMES.
ALPHABET alphabet-name IS STANDARD-1.
CODE-SET IS alphabet-name
Record description for the file contains
SIGN IS LEADING SEPARATE CHARACTER
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ112A* SQ112A creates a sequential file and verifies the contents. The file is then opened for output a second time, and accessed to make sure that the results of the second OPEN OUTPUT are in accordance with 4.3.4 General Rule 21. This rule indicates that when a file is opened for output it should contain no data records.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 7 tests.
- SQ113A* This program creates a magnetic tape file containing 750 fixed length records, each 120 characters long. The file is read twice. The first pass checks that all the expected records are present. The second pass performs similar checks, but uses all four variants of the READ statement with the END phrase that can be produced by including or omitting the optional words "RECORD" and "AT". The program omits the optional words "ORGANIZATION IS" from the "ORGANIZATION IS SEQUENTIAL" clause of the file control entry, and places the ASSIGN clause in a position other than the first in the same entry.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 22 tests.
- SQ114A* This program creates and verifies a fixed length magnetic tape file and a fixed length mass storage file. The magnetic tape file contains 750 fixed length records of 120 characters. The mass storage file contains 649 fixed length records of 120 characters. The SAME AREA clause is specified for the two files. The tape file is passed three times (read or written); ie, rewound twice.
- X-numbers: 01, 14, 55, 62, 82, 83, 84.
- This program contains 15 tests.

- SQ115A This program creates and verifies a mass storage file, which is then opened as an I-O file and every tenth record is rewritten. The updated file is verified to ensure that the REWRITE statements are executed correctly.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 3 tests.
- SQ116A This program creates and verifies a mass storage file containing 550 fixed length records of 130 characters, which is then opened as I-O. Records are updated using REWRITE FROM statements. The file is then verified again.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 10 tests.
- SQ117A This routine creates a mass storage file using WRITE FROM statements. The file contains 495 fixed length records of 141 characters. The file is opened in the input mode, and records are checked to ensure that the WRITE FROM statements are executed correctly.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 8 tests.
- SQ121A This tests the USE AFTER STANDARD ERROR PROCEDURE ON I-O statement. A file with 550 fixed length records of 126 characters is created and verified, then opened in the I-O mode and every tenth record is rewritten. The updated file is then verified.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 3 tests.
- SQ122A A one record file with two characters per block is created with the intention that it should end part-way through a block. The file is re-opened and three read statements executed. The first should be executed successfully, the second raise the AT END condition, and the third, which is a read after end of file, should cause the I-O status code 46.
- X-numbers: 14, 55, 82, 83, 84.
- This program contains 7 tests.

- SQ123A A file assigned to a medium which is not a reel/unit medium is opened and one record written to the file. A CLOSE UNIT statement is executed - this should have no effect on the file, except to cause I-O status 7. The file should remain open. A second record is then written and a normal, unqualified CLOSE statement is executed. This should be successful and close the file.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 9 tests.
- SQ124A A file assigned to a medium which is not a reel/unit medium is opened then a CLOSE UNIT statement is executed - this should cause I-O status 7. The file should remain open. A second record is then written and a normal, unqualified CLOSE statement is executed. This should be successful and close the file. The file is then reopened for input and the two records checked. A CLOSE UNIT statement is executed before the first record is read, and again this should have no effect on subsequent operations on the file. After the two records have been read, a further READ statement is executed to raise the AT END condition.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 19 tests.
- SQ125A* SQ125A attempts to OPEN for OUTPUT a magnetic tape file which is already open in the output mode. This should result in a recognition of a logic error condition and an I-O status of "41". The program contains an applicable declarative procedure, which should be implemented.
- X-numbers: 01, 55, 82, 83, 84.
- This program contains 2 tests.
- SQ126A* The program SQ126A tests the use of the NOT AT END and END-READ phrases of the READ statement.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 7 tests.
- SQ127A SQ127A opens creates a one-record mass storage file. The record size is defined using the RECORD CONTAINS clause. The file is then opened for I-O and a READ and a REWRITE statement executed. The record size for the REWRITE statement is greater than the maximum size for the file, and should cause status "44" to be raised and the declaratives executed.
- X-numbers: 55, 60, 62, 82, 83, 84.
- This program contains 6 tests.

- SQ128A* SQ128A tests Level 1 OPEN series and CLOSE series statements. INPUT and OUTPUT clauses are used in series, together and separately. Several files are created and processed both on tape and mass storage.
- X-numbers: 01, 14, 15, 55, 62, 82, 83, 84.
- This program contains 9 tests.
- SQ129A* SQ129A attempts to open for input a magnetic tape file which is not present. This should result in a permanent error and an I-O status of "35".
- This program has been split from V2.0 onwards. The new programs are SQ141A and SQ142A.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ130A SQ130A attempts to open for I-O a mass storage file which is not present. This should result in a permanent error and an I-O status of "35". The program does not contain a declarative procedure and in these circumstances the standard allows the implementor to terminate execution of the program or continue.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ131A This attempts to OPEN in the I-O mode a mass storage file which is already open in the output mode. This should raise status "41", and as no declarative procedure exists the implementation may either terminate execution or continue processing.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 2 tests.
- SQ132A This program checks for the correct response to closing an unopened file. The test for correct I-O status code 42 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84.
- This program contains 1 test.

- SQ133A This program opens for output a file which is assigned to a mass storage medium, writes one record and closes the file. The file is then opened for I-O, and two READ statements executed. The second one should cause an AT END and thus be unsuccessful. A REWRITE statement is then executed - this should cause an exception condition with Status "43" and entry to the applicable declarative.
- This program has been split from V2.0 onwards. The new program is SQ144A.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 15 tests.
- SQ134A This program opens for output a file which is assigned to a mass storage medium, writes one record and closes the file. Two record sizes are defined for the file. The file is then opened for I-O, the record is READ, and an attempt is made to REWRITE it using the other record size. This should raise error status "44" and cause execution of the appropriate declarative procedure.
- This program should be run only if an implementation provides variable-length records for the RECORDS CONTAINS integer TO integer clause.
- This program has been split from V2.0 onwards. The new program is SQ145A.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 15 tests.
- SQ135A This program checks for the correct response to closing an already closed file. The test for correct I-O status code 42 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84.
- This program contains 1 test.
- SQ136A Split from SQ122A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to reading past the end of a file. (See SQ122A).
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.

- SQ137A Split from SQ122A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to reading past the end of a file. (See SQ122A).
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ138A Split from SQ122A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to reading past the end of a file. (See SQ122A).
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ139A* Split from SQ125A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to an OPEN for OUTPUT for a magnetic tape file which is already open in the output mode. (See SQ125A).
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ140A* Split from SQ125A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to an OPEN for OUTPUT for a magnetic tape file which is already open in the output mode. (See SQ125A).
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ141A* Split from SQ129A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to an OPEN for INPUT on a tape file which is not present. (See SQ129A).
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ142A* Split from SQ129A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to an OPEN for INPUT on a tape file which is not present. (See SQ129A).
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 1 test.

- SQ143A* This program checks for the correct response to closing an unopened file. The test for correct I-O status code 42 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 42 occurred or that an error is present relative to line 078600 of the original source program.
- X-numbers: 01, 55, 82, 83, 84.
- This program contains 1 test.
- SQ144A Split from SQ133A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response when an attempt is made to REWRITE AFTER AT END. (See SQ133A).
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ146A* This program checks for the correct response to closing an already closed file. The test for correct I-O status code 42 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 42 occurred or that an error is present relative to line 090400 of the original source program.
- X-numbers: 01, 55, 82, 83, 84.
- This program contains 1 test.
- SQ147A This program checks for the correct response to reading a closed file. The test for correct I-O status code 47 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 1 test.
- SQ148A This program checks for the correct response to reading a file open in the output mode. The test for correct I-O status code 47 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 2 tests.

- SQ149A* This program checks for the correct response to reading a file that is not open (not open in the input or I-O mode). The test for correct I-O status code 47 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 47 occurred or that an error is present relative to line 093000 of the original source program.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 1 test.
- SQ150A* This program checks for the correct response to reading a file open in the output mode. The test for correct I-O status code 47 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 47 occurred or that an error is present relative to line 093000 of the original source program.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 1 test.
- SQ151A* This program checks for the correct response to writing a closed file. The test for correct I-O status code 48 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 01, 55, 82, 83, 84
- This program contains 1 test.
- SQ152A This program checks for the correct response to writing a file not open in the output mode. The test for correct I-O status code 48 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 1 test.
- SQ153A This program checks for the correct response to writing a file open in the I-O mode. The test for correct I-O status code 48 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84
- This program contains 1 test.

- SQ154A* This program checks for the correct response to writing a file that is not open (not open in the output or extend mode). The test for correct I-O status code 48 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 48 occurred or that an error is present relative to line 093700 of the original source program.
- X-numbers: 01, 55, 82, 83, 84
- This program contains 1 test.
- SQ155A* This program checks for the correct response to writing to a file open in the input mode. The test for correct I-O status code 48 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 48 occurred or that an error is present relative to line 093700 of the original source program.
- X-numbers: 01, 55, 82, 83, 84
- This program contains 1 test.
- SQ156A* This program checks for the correct response to writing to a file open in the I-O mode. The test for correct I-O status code 48 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 48 occurred or that an error is present relative to line 093700 of the original source program.
- X-numbers: 01, 55, 82, 83, 84
- This program contains 1 test.
- OBSQ1A* This tests obsolete features ie, they are still valid features in COBOL 85 and as such must be validated within the Level 1 Sequential I-O Module, in order to show conformity to the current Standard, but will be dropped from the next COBOL Standard.
- X-numbers: 01, 55, 62, 69, 74, 75, 82, 83.
- This program contains 6 tests.

SQ201M This program tests the Level 2 WRITE statement. The File Description for the printer file contains the following LINAGE clause:

```
LINAGE IS 50 LINES
    WITH FOOTING AT 45
    LINES AT TOP 10
    LINES AT BOTTOM 6
```

The program also tests:

```
LINAGE-COUNTER values after the execution of
    OPEN OUTPUT...
    WRITE... AFTER ADVANCING PAGE
    page overflow
    WRITE
    WRITE... integer LINE
    WRITE... identifier-2 LINES
    WRITE... end-of-page combinations
```

X-numbers: 55, 62, 82, 83, 84.

This program contains 23 tests.

SQ202A* The routine SQ202A creates a magnetic tape containing 750 fixed length records of 120 characters. The file is then validated in this program and passed to SQ203A for further processing.

X-numbers: 01, 55, 62, 82, 83, 84.

This program contains 1 test.

SQ203A* This program tests the use of the OPTIONAL clause and the RESERVE integer AREA clause in the SELECT statement. The optional files SQ-FS2 and SQ-FS4 are not present during execution. The following COBOL constructs are used:

```
    SELECT OPTIONAL...
    RESERVE... AREAS
    RESERVE... AREA
    USE AFTER STANDARD EXCEPTION PROCEDURE
    ON INPUT
    READ...; AT END...
    READ...
```

X-numbers: 01, 03, 17, 18, 55, 62, 82, 83, 84.

This program contains 4 tests.

- SQ204A* The OPEN EXTEND statement for both a magnetic tape file and a mass storage file is tested here. A magnetic tape file is created which contains 750 fixed length records of 126 characters. The file is then extended by 250 records and the extended file is verified. The same test is repeated for a mass storage file.
- X-numbers: 01, 14, 55, 62, 82, 83, 84.
- This program contains 2 tests.
- SQ205A* This program tests the USE statement with file-name series, and the FILE STATUS clause in the SELECT clause. A READ statement without the AT END phrase causes the USE procedure to be executed when an end-of-file is encountered. A magnetic tape file and a mass storage file are created, each with 500 fixed length records of 120 characters.
- X-numbers: 01, 14, 55, 62, 82, 83, 84.
- This program contains 2 tests.
- SQ206A* SQ206A tests the SAME AREA and SAME RECORD AREA clauses in the I-O-CONTROL paragraph. Two magnetic tape files and two mass storage files are created and verified.
- X-numbers: 01, 02, 14, 15, 55, 62, 82, 83, 84.
- This program contains 4 tests.
- SQ207M This tests the Level 2 WRITE statement containing the FROM and ADVANCING mnemonic-name phrases.
- X-numbers: 55, 62, 73, 82, 83, 84.
- This program contains 8 tests.
- SQ208M The Level 2 WRITE statement is tested in SQ208M. The File Description for a printer file containing the following LINAGE clause is also tested:
- ```
LINAGE data-name
 FOOTING data-name
 TOP data-name
 BOTTOM data-name
```
- Contents of the data-names are changed to check the redefinitions of logical page formats after page overflow or WRITE ADVANCING PAGE operations.
- X-numbers: 55, 62, 82, 83, 84.
- This program contains 7 tests.

- SQ209M      The routine SQ209M tests the Level 2 WRITE statement and the File Description for a printer file containing the following LINAGE clause:
- LINAGE 40  
      TOP 2
- X-numbers: 55, 62, 82, 83, 84.
- This program contains 3 tests.
- SQ210M      This program tests the Level 2 WRITE statement and the File Description for a printer file containing the following LINAGE clause:
- LINAGE IS data-name LINES  
                  TOP 2
- The contents of data-name are changed during execution in order to verify the redefinition of logical page formats.
- X-numbers: 55, 62, 82, 83, 84.
- This program contains 3 tests.
- SQ211A\*     This program tests the CLOSE statement with the WITH LOCK phrase. A magnetic tape file with one record is created and is closed with lock. The file is then re-opened after it has been closed with lock. There are no declarative procedures. The test for correct I-O status code is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 38 occurred or that an error is present relative to line 089540 of the original source program.
- X-numbers: 01, 55, 82, 83, 84.
- This program contains 4 tests.
- SQ212A      This builds and validates a sequential mass storage file containing 2031 records varying in length from 18 to 2048 characters, then attempts to write smaller and larger records than those contained in the file and the FILE STATUS codes are checked.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.

- SQ213A\* This tests the USE AFTER ERROR PROCEDURE for EXTEND and file-name series. A magnetic tape file of 750 fixed length 126-character records is created, then extended by 250 records and verified. A mass storage file containing 1000 fixed length records of 126 characters is also created and verified.
- X-numbers: 01, 14, 55, 62, 82, 83, 84.
- This program contains 7 tests.
- SQ214A This program tests Sequential I-O operations involving records containing subordinate entries, which in turn contain Format 2 OCCURS clauses; ie, ... OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1 .... A file is created with 1000 fixed-length 140-character records while varying the value of data-name-1. The file is then read again while varying the value of data-name-1. The effect of varying the number of occurrences during READ and WRITE operations is evaluated.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 5 tests.
- SQ215A This program tests the CLOSE statement with the WITH LOCK phrase. A mass storage file is created, one record is written to it, and it is closed with lock. An attempt is then made to reopen the file. I-O status code 38 is expected and tested in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 14, 55, 82, 83, 84.
- This test contains 4 tests.
- SQ216A\* SQ216A tests the clause PADDING CHARACTER IS "9" (LITERAL). A tape file is created containing 750 fixed length records. The file is then CLOSEd and OPENed as an input file, read, and fields in the input records compared to the values written, to ensure that the records were processed correctly. The file is again CLOSEd and OPENed as an input file and four READ format options used to read the file and verify fields in the records.
- X-numbers: 01, 55, 62, 82, 83, 84.
- This program contains 7 tests.

SQ217A\* This tests the .PADDING CHARACTER IS data-name-1 (VALUE "Z") clause. A tape file is created with 750 fixed length records. It is then CLOSEd and OPENed as an input file, read, and fields in the input records compared to the values written, to ensure that the records were processed correctly. The file is again CLOSEd and OPENed as an input file and four READ format options used to read the file and verify fields in the records.

X-numbers: 01, 55, 62, 82, 83, 84.

This program contains 7 tests.

SQ218A This checks the RECORD DELIMITER IS STANDARD-1 clause. A tape file containing both 120 and 151 character records is created then read and fields in the records are checked against the expected values.

Note: the results of this program are only valid if tape files are being tested i.e. if tape files are not supported this program need not be run.

X-numbers: 01, 55, 62, 82, 83, 84.

This program contains 6 tests.

SQ219A checks the RECORD DELIMITER IS implementor-name clause. A sequential tape file containing both 120 and 151 character records is created. The tape is read and fields in the records are checked against the expected values.

Note: the results of this program are only valid if tape files are being tested i.e. if tape files are not supported this program need not be run.

: The implementor-name must not be STANDARD-1 for the purpose of this test.

: If the implementor-name phrase is not implemented, this program need not be run.

X-numbers: 01, 55, 62, 70, 82, 83, 84.

This program contains 6 tests.

SQ220A This routine checks the RECORD IS VARYING and the NEXT RECORD clause. A sequential mass storage file containing both 120-character and 151-character records is created. The file is read and fields in the records are checked against the expected values.

X-numbers: 14, 55, 62, 82, 83, 84.

This program contains 6 tests.

- SQ221A This test builds a sequential mass storage file with 120 and 151 character records. Use of the RECORD IS VARYING... DEPENDING ON... phrase is tested.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 6 tests.
- SQ222A This routine builds a sequential mass storage file which contains both 120 character and 151 character records. The use of the RECORD IS VARYING... phrase is tested.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 6 tests.
- SQ223A This routine builds a sequential mass storage file which contains both 120 character and 151 character records. The file is read and fields in the records are checked against the expected values. The use of the RECORD IS VARYING IN SIZE FROM integer-1 TO integer-2 CHARACTERS clause is tested.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 6 tests.
- SQ224A This routine builds a sequential mass storage file which contains 2031 records of a length of 18 to 2048 characters. The file is read and fields in the records are checked against the expected values. The use of the RECORD IS VARYING IN SIZE FROM integer-1 TO integer-2 CHARACTERS DEPENDING ON data-name clause is tested. The READ and WRITE statements are used, with and without the INTO clause.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 3 tests.
- SQ225A SQ225A attempts to open for extend a mass storage file which is not present. This should result in a permanent error and an I-O Status of "35" and cause the declarative section to be executed.
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 3 tests.

- SQ226A      This program opens for output a file which is assigned to a mass storage medium, writes one record and closes the file. The file is then opened for input and two read statements executed. The second should cause an AT END condition. An OPEN EXTEND statement is then executed. This should cause an exception condition, with I-O status "41" and entry to the applicable error declarative. There are declaratives for all four OPEN modes, and either the INPUT or the EXTEND declarative may be entered at this point.
- X-numbers: 14, 55, 62, 82, 83, 84.
- Reference: VII-51, 4.6.4, GR (5) B and E.
- This program contains 37 tests.
- SQ227A      This program opens for output a file which is assigned to a mass storage medium. One record is then written to this file which is then closed. The file is then opened for I-O, and a READ statement on the file is carried out. A RE-WRITE on a record that is too long for the file is attempted which should cause an exception condition with I-O status "44". This logic error should cause entry to the applicable error declarative.
- This program has been split from version 2.0 onwards. The new program is SQ228A.
- X-Numbers: 14, 55, 62, 82, 83, 84.
- This program contains 16 tests.
- SQ228A      Split from SQ227A, this program repeats the sequence of file handling routines in order to carry out the isolated split test that checks for the correct response to a RE-WRITE on a record that is too long for the file. (See SQ227A).
- X-numbers: 14, 55, 62, 82, 83, 84.
- This program contains 1 test.
- SQ229A\*      This program checks for the correct response to reading a file open in the extend mode. The test for correct I-O status code 47 is in the declaratives. An abnormal termination is possible after the test of the I-O status code is accomplished but before control is returned to the main line code.
- X-numbers: 01, 55, 82, 83, 84.
- This program contains 1 test.

SQ230A\* This program checks for the correct response to reading a file open in the extend mode. The test for correct I-O status code 47 is in the main line code, therefore an abnormal termination is possible before the test of the I-O status code is accomplished. If an abnormal termination occurs an execution message should be produced indicating that either I-O status 47 occurred or that an error is present relative to line 093000 of the original source program.

X-numbers: 01, 55, 82, 83, 84.

This program contains 1 test.

OBSQ3A This tests the MULTIPLE FILE TAPE clause in the I-O CONTROL paragraph and the NO REWIND phrase in the OPEN and CLOSE statements. Two magnetic tapes, each containing four files, are created and passed to the programs OBSQ4A and OBSQ5A. Files placed on the magnetic tape specified by X-08 are passed to both OBSQ4A and OBSQ5A, whereas files on X-09 are passed to SQ209A only.

This routine tests features which are deemed obsolete in COBOL 85; ie, they are still valid features, and as such must be validated within the Level 2 Sequential I-O Module in order to show conformity to the current Standard, but will be dropped from the next COBOL Standard.

Note: The results of this program are only meaningful if magnetic tape files are being tested; therefore, if magnetic tape media is not supported this program need not be run.

X-numbers: 04, 05, 08, 09, 10, 11, 12, 13, 55, 62, 82, 83.

This program contains 0 tests.

OBSQ4A This program verifies four of the eight magnetic tape files created in OBSQ3A using the MULTIPLE FILE clause. The files are verified in an order which differs from the order in which they were created. The magnetic tape files are passed to OBSQ5A for further processing.

This routine tests features which are deemed obsolete in COBOL 85; ie, they are still valid features, and as such must be validated within the Level 2 Sequential I-O Module in order to show conformity to the current Standard, but will be dropped from the next COBOL Standard.

Note: The results of this program are only meaningful if magnetic tape files are being tested; therefore, if magnetic tape media is not supported this program need not be run.

X-numbers: 04, 08, 09, 10, 55, 62, 82, 83.

This program contains 4 tests.

OBSQ5A

The program verifies five of the eight magnetic tape files created in OBSQ3A using the MULTIPLE FILE clause. The files are verified in an order which differs from the order in which they were created.

This routine tests features which are deemed obsolete in COBOL 85; ie, they are still valid features, and as such must be validated within the Level 2 Sequential I-O Module in order to show conformity to the current Standard, but will be dropped from the next COBOL Standard.

Note: The results of this program are only meaningful if magnetic tape files are being tested; therefore, if magnetic tape media is not supported this program need not be run.

X-numbers: 05, 08, 09, 11, 12, 13, 55, 62, 82, 83.

This program contains 5 tests.



## 4.12 SORT-MERGE MODULE

### ANSI DOCUMENT REFERENCE X.3.23-1985 SECTION XI

#### FUNCTION:

The Sort-Merge module provides the capability to order one or more files of records, or to combine two or more identically ordered files of records, according to a set of user-specified keys contained within each record. Optionally, a user may apply some special processing to each of the individual records by input or output procedures. This special processing may be applied before and/or after the records are ordered by the SORT or after the records have been combined by the MERGE. The Sort-Merge module contains only level 1 elements.

A number of ST test programs are applicable to both magnetic tape and mass storage files. Each of these programs is annotated with an "\*" next to the program name. If support for both magnetic tape and mass storage is claimed, the identified programs must be run twice - once on each media. Otherwise, run the program on whichever media is supported.

ST101A\*      ST101A tests the INPUT PROCEDURE and the OUTPUT PROCEDURE of the SORT statement. An output file is created on magnetic tape containing 100 fixed length records of 120 characters, each with 10 records per block. Five sort key data items of 1, 2, 3, 4 and 5 characters are defined when the sort file is created, and they are tested in the sort output records in the OUTPUT PROCEDURE. The sorted file from program ST101A is passed to ST102A for further processing. The SORT statement is:

```
SORT file-name
 ON ASCENDING KEY...
 ON DESCENDING KEY...
 ON ASCENDING KEY...
 DESCENDING...
 INPUT PROCEDURE IS...
 OUTPUT PROCEDURE IS... THRU...
```

X-numbers: 01, 27, 55, 69, 74, 75, 82, 83.

This program contains 9 tests.

**ST102A\*** This program tests the USING and GIVING phrases of the SORT statement. The file created in ST101A is used as input by referencing it in the USING phrase. The file created in this program is passed to ST103A for further processing. The output file is referenced in the GIVING phrase. The SORT statement in this program is:

```
SORT file-name
 ON DESCENDING KEY...
 ON ASCENDING KEY...
 ON DESCENDING KEY...
 ASCENDING...
 USING...
 GIVING...
```

X-numbers: 01, 02, 27, 69, 74, 75, 76, 82, 83.

This program does not produce a report.

**ST103A\*** This program checks the contents of various records and checks for premature end-of-file in the file created by ST102A. No SORT statement is present.

X-numbers: 02, 55, 69, 74, 76, 82, 83.

This program contains 9 tests.

**ST104A\*** This program builds a 203 record file to be sorted in ST105A. Each 18-character record consists of a 6-character SORT key. The file is not created in sorted order.

X-numbers: 01, 55, 69, 74, 75, 82, 83.

This program contains 1 test.

**ST105A\*** This program tests the USING phrase and the OUTPUT PROCEDURE of the SORT statement. The file created in ST104A is used as input to the SORT and is referenced in the USING phrase. The SORT statement in this program is:

```
SORT file-name ON
 DESCENDING...
 key-item
 USING...
 OUTPUT PROCEDURE... THRU...
```

X-numbers: 01, 02, 27, 55, 69, 74, 75, 76, 82, 83.

This program contains 2 tests.

**ST106A\*** This program tests the GIVING phrase and the INPUT PROCEDURE of the SORT statement. An output file is created on magnetic tape, containing 9 fixed length records of 27 characters each. Three sort key data items of 1, 8 and 18 characters make up the entire record. After the file is sorted it is passed, through the GIVING phrase, to ST107A for further processing. The SORT statement in this program is:

```
SORT file-name ON
 ASCENDING...
 DESCENDING...
 ASCENDING...
 INPUT PROCEDURE... THRU...
 GIVING...
```

X-numbers: 01, 27, 55, 69, 74, 75, 82, 83.

This program contains 1 test.

**ST107A\*** This program checks the contents of various records in the file created by ST106A. No SORT statement is present.

X-numbers: 01, 55, 69, 74, 75, 82, 83.

This program contains 6 tests.

**ST108A** This program tests the use of the INPUT PROCEDURE and the OUTPUT PROCEDURE of the SORT statement, and the ability of the compiler to handle eight ascending key data items in one file. The program is completely self-contained. The SORT statement in this program is:

```
SORT file-name ON
 ASCENDING KEY...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 INPUT PROCEDURE...
 OUTPUT PROCEDURE... THRU...
```

X-numbers: 27, 55, 82, 83.

This program contains 9 tests.

**ST109A\*** This program builds a magnetic tape file of 40 variable length records which is sorted in ST110A and checked in ST111A. The file is created in unsorted order and consists of records of 50, 75 and 100 characters.

X-numbers: 01, 55, 69, 74, 75, 82, 83.

This program contains 0 tests.

**ST110A\*** This program tests the USING and GIVING phrases of the SORT statement for a file with variable length records. The file created in ST109A is used as input to the SORT and is referenced in the USING phrase. The sorted output file is named in the GIVING phrase and is passed to ST111A for further processing. Two sort key data items of 10 and 38 characters are defined. The SORT statement in this program is:

```
SORT file-name
 DESCENDING KEY
 key-1
 key-2
 USING...
 GIVING...
```

X-numbers: 01, 02, 27, 55, 69, 74, 75, 76, 82, 83.

This program does not produce a report.

**ST111A\*** ST111A checks the contents of various records in the file created in ST109A and sorted in ST110A. No SORT statement is present in ST111A.

X-numbers: 02, 55, 69, 74, 76, 82, 83.

This program contains 7 tests.

**ST112M\*** This program builds a three reel sequential tape file consisting of 26 records, with each record containing 33 identical alphabetic characters from A to Z. The file is passed to ST113M where it is sorted and to ST114M where it is tested.

Special considerations:

1. A three reel magnetic tape file is required if allowed by the operating system.
2. The H-option is used to include all CLOSE REEL statements in the source program when a multi-reel file is specified.
3. The I-option is used to delete all CLOSE REEL statements from the source program when a multi-reel file is not specified.

X-numbers: 06, 55, 69, 74, 79, 82, 83.

This program contains 0 tests.

**ST113M\*** This program tests the USING and GIVING phrases of the SORT statement for a multi-reel tape file. The three reel tape file created in ST112M is used as input to the SORT and is referenced in the USING phrase. The sorted output file is named in the GIVING phrase and is passed to ST114M as a single reel file. A single sort key data item of 33 characters makes up the entire record. The SORT statement in this program is:

```
SORT file-name DESCENDING
 key
 USING...
 GIVING...
```

X-numbers: 01, 06, 27, 55, 69, 74, 75, 79, 82, 83.

This program does not produce a report.

**ST114M\*** This program checks the contents of various records in the file created in ST112M and sorted in ST113M. The records were created in ascending order and sorted in descending order.

X-numbers: 01, 55, 69, 74, 75, 82, 83.

This program contains 10 tests.

**ST115A\*** This program builds a sequential file, SQ-FS1, which is passed to ST116A to be sorted. File SQ-FS1 has fixed length records (507 characters/record) and is unblocked. The number of records in this file is set by a four digit integer in X-65. It should be large enough to force the SORT routine in ST116A to be non-core resident; ie, force the system to use some means of auxiliary storage for the SORT sub-strings.

X-numbers: 01, 55, 65, 69, 74, 75, 82, 83.

This program contains 0 tests.

**ST116A\*** This program is a test of the SORT statement using fixed length records. The collating sequence is native; ie, no collating statement is used in the actual SORT statement. Qualified alphanumeric and numeric sort keys are used. The actual SORT statement is as follows:

```
SORT sort-file-name
 ON ASCENDING KEY key-1 OF data-name-1
 ASCENDING key-2 OF data-name-2
 USING file-name-1
 GIVING file-name-2
```

The input file SQ-FS1 is created in ST115A. The output file SQ-FS2 is passed to ST117A for checking.

X-numbers: 01, 02, 27, 69, 74, 75, 76, 82, 83.

This program does not produce a report.

**ST117A\*** This program checks the file SQ-FS2, which was sorted into native ascending order by program ST116A. X-63 is used to input the system's native ascending collating sequence as a literal of the 51 COBOL characters. Because of confusion over the use of a quote sign embedded within a non-numeric literal, the quote sign is omitted and the dollar sign is repeated. An example of an X-63 card for a machine with ASCII as the native collating sequence is as follows:

```
X-63 " $$(*+,-./0123456789;<=>)ABCDEFGHIJKLM
NOPQRSTUVWXYZ".
```

The X-65 card shows how many records are to be used in the sequence of programs ST115A, ST116A and ST117A.

X-numbers: 02, 55, 65, 69, 74, 76, 82, 83.

This program contains 1 test.

**ST118A** ST118A tests the use of the INPUT PROCEDURE and OUTPUT PROCEDURE of the SORT statement, and the ability of the compiler to handle eight ascending key data items in one file. Each key identified in the sort statement is an elementary data item, and uses various combinations of picture character-string symbols and clauses for describing the general characteristics of the data item. The program is a rewrite of ST108A and specifically tests the use of the SIGN clause, with sort key fields for the file.

```
SORT file-name ON
 ASCENDING KEY...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 ASCENDING...
 INPUT PROCEDURE...
 OUTPUT PROCEDURE... THRU...
```

X-numbers: 27, 55, 82, 83.

This program contains 9 tests.

- ST119A\*** ST119A tests the SORT statement INPUT PROCEDURE and OUTPUT PROCEDURE phrases, and specifically the transfer of control into and out of the INPUT and OUTPUT PROCEDURE and the use of paragraph names to specify those PROCEDURES. The NOT AT END and END-RETURN phrases of the RETURN statement are also tested. An output file is created on magnetic tape containing 100 fixed length records of 120 characters each with 10 records per block. Five sort key data items of 1, 2, 3, 4 and 5 characters are defined when the sort file is created, and they are tested in the sorted output records in the OUTPUT PROCEDURE. The sorted file is passed to ST120A for further processing.
- X-numbers: 01, 27, 55, 69, 74, 75, 82, 83.
- Reference: XI-14, 16, 19.
- This program contains 27 tests.
- ST120A\*** This program SORTs the file produced by ST119A using the GIVING and USING phrases, and creates a resequenced file which is passed to ST121A for further processing.
- X-numbers: 01, 02, 27, 69, 74, 75, 76, 82, 83
- This program does not produce a report.
- ST121A\*** Program ST121A verifies the contents of the resequenced file produced by the SORT procedure in program ST120A.
- X-numbers: 02, 55, 69, 74, 76, 82, 83.
- This program contains 9 tests.
- ST122A\*** Program ST122A sets up a data file for subsequent processing by ST123A and ST124A. This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.
- X-numbers: 01, 55, 69, 74, 75, 82, 83.
- This program contains 0 tests.
- ST123A\*** This program tests the sorting of variable length records on the file created in ST122A. The file is then passed to ST124A for further processing. This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.
- X-numbers: 01, 02, 27, 55, 69, 74, 75, 76, 82, 83.
- Reference: XI-7, VII-30.
- This program does not produce a report.

- ST124A\* This program checks the contents of the file produced by ST123A. This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.
- X-numbers: 02, 55, 69, 74, 76, 82, 83.
- This program contains 7 tests.
- ST125A\* This program tests the facility of multiple files in the GIVING phrase of the SORT statement. The contents of the three output files produced in this program are verified in ST126A.
- X-numbers: 01, 02, 03, 27, 55, 69, 74, 75, 82, 83.
- Reference: XI-20.
- This program contains 1 test.
- ST126A\* This program verifies the contents of the three files produced by ST125A.
- X-numbers: 01, 02, 03, 55, 69, 74, 75, 82, 83.
- Reference: XI-20.
- This program contains 18 tests.
- ST127A\* This routine tests the use of the DUPLICATES phrase of the SORT statement. The order of records having duplicate keys after the execution of a SORT statement must be the same as the order of those records on input to the SORT statement. ST127A is based on ST108A and generates 20 records, nine of which have identical keys, and all the records contain a unique identifier. These records are not released to the SORT contiguously but should appear in the output file consecutively and in the same order in which they were presented.
- X-numbers: 27, 55, 82, 83.
- Reference: XI-18.
- This program contains 27 tests.

ST131A\* This program tests the use of three SORT statements in the same program, and also the use of the SAME RECORD AREA clause. Numeric and alphabetic key data items are used. A 100 record file with a blocking factor of 10 is created and then sorted, GIVING a file with the same file description for use in the second sort. The file is again sorted, GIVING a file for use in the third SORT. No file is output from the third SORT. The SORT statements in this program are:

SORT file-name  
    ON DESCENDING KEY...  
    ON ASCENDING KEY...  
    USING...  
    GIVING...

SORT file-name  
    ASCENDING...  
    DESCENDING...  
    ASCENDING...  
    INPUT PROCEDURE...  
    GIVING...

SORT file-name  
    ON DESCENDING KEY...  
    ASCENDING...  
    INPUT PROCEDURE IS...  
    OUTPUT PROCEDURE IS... THRU...

X-numbers: 01, 14, 15, 27, 28, 29, 55, 74, 75, 76, 77, 82, 83.

This program contains 15 tests.

ST132A\*

This program tests the use of three SORT statements in the same program. The first SORT generates input data in the INPUT PROCEDURE. The SORT results are tested in the OUTPUT PROCEDURE, which also creates a two-reel 100 record tape file (via the CLOSE REEL statement) for input to the second SORT. The third SORT exercises the sorted file for the second time. Successful execution is the sole test of this SORT. Alphabetic and numeric sort key data items of 10 characters each are defined. Also tested in this program is the SAME SORT AREA clause. The SORT statements in this program are:

SORT file-name  
    DESCENDING...  
    INPUT PROCEDURE...  
    OUTPUT PROCEDURE...

SORT file-name  
    ASCENDING...  
    USING...  
    OUTPUT PROCEDURE...

SORT file-name  
    ASCENDING...  
    USING...  
    OUTPUT PROCEDURE...

Special considerations:

1. A two-reel magnetic tape file is required if allowed by the operating system.
2. The H-option is used to include all CLOSE REEL statements in the source program when a multi-reel file is specified.
3. The I-option is used to delete all CLOSE REEL statements from the source program when a multi-reel file is not specified.

X-numbers: 06, 27, 28, 55, 69, 74, 75, 82, 83.

This program contains 6 tests.

ST133A\* This tests the use of two SORT statements being applied to the same 80 character record, and the use of the SAME AREA clause. The output from the first SORT becomes the input for the second. The file is copied tape to tape in the mainline section between the sorts, and both copies are checked in the OUTPUT PROCEDURE of the second SORT. The sort key is defined as an 8 character signed numeric data item. The SORT statements in this program are:

```
SORT file-name ON DESCENDING KEY
key
INPUT PROCEDURE...
GIVING...
```

```
SORT file-name ON ASCENDING KEY
key
USING...
OUTPUT PROCEDURE...
```

X-numbers: 01, 02, 27, 28, 55, 69, 74, 75, 76, 82, 83.

This program contains 18 tests.

ST134A\* ST134A tests the use of the SORT statement on a file already in sorted order. A magnetic tape of eleven 80-character records is created in an independent section of the program, and sorted employing the INPUT PROCEDURE and OUTPUT PROCEDURE clauses. Finally the program checks the contents of various records in the file. Use of the SAME RECORD AREA clause is also tested. The SORT statement is:

```
SORT file-name ON ASCENDING
key
INPUT PROCEDURE IS...
OUTPUT PROCEDURE IS...
```

X-numbers: 01, 02, 27, 55, 69, 74, 75, 76, 82, 83.

This program contains 4 tests.

ST135A\* This builds a 21 record file, sorts it and checks it in the OUTPUT PROCEDURE of the SORT statement. The files share a network of SAME AREA, SAME RECORD AREA and SAME SORT AREA clauses. In order to exercise the SAME clause options, two dummy files are opened, closed, read and rewritten. (The contents of the two files are never checked.) The sort keys are defined as eight-character numeric data items.

This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.

X-numbers: 01, 02, 03, 04, 27, 55, 69, 74, 75, 76, 77, 78, 82, 83.

This program contains 9 tests.

ST136A\* This tests the use of the FROM phrase of the RELEASE statement. Ten records are created with numerically descending keys, then sorted into ascending sequence and the results checked. The sort keys are defined as eight-character signed numeric data items.

X-numbers: 01, 27, 55, 69, 74, 75, 82, 83.

This program contains 5 tests.

ST137A This program is used to test the SORT statement, using variable length records which contain an OCCURS DEPENDING ON clause. The input file is a 51 record mass storage file, containing records varying in size from 148 to 1435 characters. This program also tests the use of qualified alphanumeric and numeric sort key data items. The SORT statement in this program is:

```
SORT file-name
 ON ASCENDING KEY... OF...
 ASCENDING KEY... OF...
 USING...
 GIVING...
```

This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.

X-numbers: 14, 15, 27, 55, 63, 69, 74, 75, 76, 82, 83.

This program contains 6 tests.

ST139A ST139A tests the MERGE statement using the ASCII collating sequence and fixed length records. Two 51 record mass storage files consisting of 132 characters per record are created by the program, in ascending ASCII order. Then the alphabet-name clause and the MERGE statement with the COLLATING SEQUENCE phrase are used to test the compiler's ability to merge two files into a third file in ascending ASCII order.

ST139A also tests the use of the SAME SORT-MERGE AREA clause, qualified alphanumeric and numeric merge key data items, and the USING file-name series of the MERGE statement. The MERGE statement in this program is:

```
MERGE file-name
 ASCENDING... OF...
 ON ASCENDING KEY... OF...
 SEQUENCE...
 USING...
 GIVING...
```

This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.

X-numbers: 14, 15, 16, 27, 55, 69, 74, 75, 76, 77, 82, 83.

This program contains 10 tests.

**ST140A\***

This program tests the MERGE statement using the ASCII collating sequence and a multiple file tape. Two files are written on a multiple file tape and a third file is created on mass storage. The program then tests the compiler's ability to MERGE the second file of the multiple file tape with the mass storage file, to produce a new mass storage file.

All input files have 51 fixed length records with 132 characters per record. The MERGE file should have 102 records. This program also tests the SAME SORT AREA clause, qualified alphanumeric and numeric merge key data items, USING file-name series, and RETURN RECORD INTO statement. The MERGE statement in this program is:

```
MERGE file-name ON
 DESCENDING KEY... OF...
 ASCENDING... OF...
 COLLATING SEQUENCE IS...
 OUTPUT PROCEDURE IS...
```

This program uses Level 2 features and need not be run when testing to Low or Intermediate level.

X-numbers: 08, 09, 14, 15, 27, 55, 69, 74, 75, 76, 77, 78, 82, 83.

This program contains 11 tests.

**ST144A\***

This program tests the MERGE statement using a native collating sequence and a multiple file tape. Initially, two 51 record files are written on a multiple file tape in ascending native order. Then a third file is written on mass storage. Finally, the MERGE statement uses the second and third files to create a new 102 record mass storage file. All files consist of 132 character records. The MERGE statement in this program is:

```
MERGE file-name
 ON DESCENDING KEY... OF...
 ASCENDING... OF...
 USING...
 OUTPUT PROCEDURE IS...
```

This program uses Level 2 features and need not be run when testing to Low or Intermediate level.

X-numbers: 08, 09, 14, 15, 27, 55, 64, 69, 74, 75, 76, 77, 78, 82, 83.

This program contains 11 tests.

**ST146A** ST146A tests SORT operations involving records containing subordinate entries, which in turn contain Format 2 OCCURS clauses; ie, ... OCCURS integer-1 TO integer-2 TIMES DEPENDING ON data-name-1.... A file (SQ-FS1) is created with 1000 fixed-length 140-character records. The file is then SORTed, RELEASEing and RETURNing records using record descriptions with the Format 2 OCCURS clause. The effect of varying data-name-1 is evaluated. The performance of the SORT verb is not checked.

X-numbers: 14, 15, 27, 55, 69, 74, 75, 76, 82, 83.

This program contains 4 tests.

**ST147A** This program creates two files then MERGEs them to produce three output files from a single MERGE statement. This program uses Level 2 features and therefore need not be run when testing to Low or Intermediate level.

X-numbers: 14, 15, 16, 17, 18, 27, 55, 63, 69, 74, 75, 76, 77, 78, 79, 82, 83.

This program contains 26 tests.

#### 4.13 THE FLAGGING TESTS.

The following is a list of all the programs that are featured in the Flagging Test suite, with a description of the language elements included in each test.

##### INTERMEDIATE

IX301M      Tests the flagging of Intermediate subset features that are used in Indexed input-output.

The following features are tested:

- 1 ) ORGANIZATION IS INDEXED
- 2 ) ACCESS MODE IS RANDOM
- 3 ) RECORD KEY IS
- 4 ) DELETE filename INVALID KEY  
      NOT INVALID KEY
- 5 ) READ filename INVALID KEY  
      NOT INVALID KEY
- 6 ) REWRITE record INVALID KEY  
      NOT INVALID KEY
- 7 ) WRITE record INVALID KEY  
      NOT INVALID KEY

NUMBER OF FLAGS EXPECTED = 7.

X-numbers: 24,82,83

RL301M      Tests the flagging of Intermediate subset features that are used in relative input-output.

The following features are tested:

- 1 ) ORGANIZATION IS RELATIVE
- 2 ) ACCESS MODE IS RANDOM
- 3 ) DELETE filename INVALID KEY  
      NOT INVALID KEY
- 4 ) READ filename INVALID KEY  
      NOT INVALID KEY
- 5 ) REWRITE record INVALID KEY  
      NOT INVALID KEY
- 6 ) WRITE record INVALID KEY  
      NOT INVALID KEY

NUMBER OF FLAGS EXPECTED = 6.

X-numbers: 21,82,83

SM301M      Tests the flagging of Intermediate subset features that are used in the copy function.

The following feature is tested:

1 )      COPY

NUMBER OF FLAGS EXPECTED = 1.

X-numbers:    82,83

ST301M      Tests the flagging of Intermediate subset features that are used in sort-merge functions.

The following features are tested:

- 1 )      SAME SORT-MERGE
- 2 )      SD filename
- 3 )      MERGE
- 4 )      RELEASE
- 5 )      RETURN
- 6 )      SORT filename

NUMBER OF FLAGS EXPECTED = 6.

X-numbers:    01,02,03,04,27,82,83

### HIGH

IC401M      Tests the flagging of high subset features that are used in inter-program communication.

The following features are tested:

- 1 )      GLOBAL names
- 2 )      EXTERNAL names
- 3 )      INITIAL program
- 4 )      COMMON programs
- 5 )      CANCEL
- 6 )      Nested programs
- 7 )      CALL...BY REFERENCE
- 8 )      CALL...BY CONTENT
- 9 )      USE GLOBAL
- 10)     END PROGRAM header

NUMBER OF FLAGS EXPECTED = 11.

X-numbers:    82,83

IF401M Tests the flagging of high subset intrinsic function features.

The following features are tested:

- 1 ) ACOS
- 2 ) ANNUITY
- 3 ) ASIN
- 4 ) ATAN
- 5 ) CHAR
- 6 ) COS
- 7 ) CURRENT-DATE
- 8 ) DATE-OF-INTEGER
- 9 ) DAY-OF-INTEGER
- 10) FACTORIAL
- 11) INTEGER
- 12) INTEGER-OF-DATE
- 13) INTEGER-OF-DAY
- 14) INTEGER-PART

NUMBER OF FLAGS EXPECTED = 14.

X-numbers: 82,83

IF402M Tests the flagging of high subset intrinsic function features.

The following features are tested:

- 1 ) LENGTH
- 2 ) LOG
- 3 ) LOG10
- 4 ) LOWER-CASE
- 5 ) MAX
- 6 ) MEAN
- 7 ) MEDIAN
- 8 ) MIDRANGE
- 9 ) MIN
- 10) MOD
- 11) NUMVAL
- 12) NUMVAL-C
- 13) ORD
- 14) ORD-MAX
- 15) ORD-MIN

NUMBER OF FLAGS EXPECTED = 17.

X-numbers: 82,83

IF403M      Tests the flagging of high subset intrinsic function features.

The following features are tested:

- 1 ) PRESENT-VALUE
- 2 ) RANDOM
- 3 ) RANGE
- 4 ) REM
- 5 ) REVERSE
- 6 ) SIN
- 7 ) SQRT
- 8 ) STANDARD-DEVIATION
- 9 ) SUM
- 10) TAN
- 11) UPPER-CASE
- 12) VARIANCE
- 13) WHEN-COMPILED

NUMBER OF FLAGS EXPECTED = 13.

X-numbers: 82,83

IX401M      Tests the flagging of high subset features that are used in indexed input-output.

The following features are tested:

- 1 ) SELECT OPTIONAL
- 2 ) RESERVE
- 3 ) ALTERNATE RECORD KEY
- 4 ) ACCESS MODE DYNAMIC
- 5 ) RECORD VARYING
- 6 ) CLOSE WITH LOCK
- 7 ) OPEN EXTEND
- 8 ) READ NEXT  
      KEY IS
- 9 ) START

NUMBER OF FLAGS EXPECTED = 10.

X-numbers: 25,26,82,83

NC401M      Tests the flagging of High subset Nucleus features.

The following features are tested:

- 1 )      Reference modification,:;
- 2 )      Qualification of data names
- 3 )      Symbolic character
- 4 )      4 - 7 subscripts
- 5 )      ALL figurative constants
- 6 )      Word/numeric literal/PIC  
character string break
- 7 )      END PROGRAM
- 8 )      DATE COMPILED (OBS)
- 9 )      ALPHABET literal
- 10)     OCCURS...ASCENDING KEY
- 11)     OCCURS...DESCENDING KEY
- 12)     Nested REDEFINES
- 13)     66 level
- 14)     88 level
- 15)     RENAMES
- 16)     Arithmetic expressions
- 17)     Sign conditions
- 18)     Complex conditions
- 19)     CORRESPONDING  
ADD  
MOVE  
SUBTRACT
- 20)     ALTER procedure-name series
- 21)     COMPUTE
- 22)     ACCEPT FROM
- 23)     DISPLAY UPON
- 24)     DIVIDE REMAINDER
- 25)     EVALUATE
- 26)     GO TO without procedure name (2 times)
- 27)     Nested conditional statement in an IF statement
- 28)     INITIALIZE
- 29)     INSPECT CONVERTING
- 30)     PERFORM...WITH TEST
- 31)     PERFORM...VARYING  
VARYING
- 32)     SEARCH
- 33)     SET TO TRUE
- 34)     STRING
- 35)     UNSTRING

NUMBER OF FLAGS EXPECTED = 40.

X-numbers:    01,02,56,82,83

RL401M      Tests the flagging of high subset features that are used in relative input-output.

The following features are tested:

- 1 )    SELECT OPTIONAL
- 2 )    RESERVE
- 3 )    ACCESS MODE DYNAMIC
- 4 )    SAME RECORD AREA
- 5 )    RECORD VARYING
- 6 )    CLOSE WITH LOCK
- 7 )    OPEN EXTEND
- 8 )    READ NEXT
- 9 )    START

NUMBER OF FLAGS EXPECTED = 9.

X-numbers:    21,22,82,83

SM401M      Tests the flagging of high subset features that are used in source text manipulation.

The following features are tested:

- 1 )    COPY REPLACING
- 2 )    REPLACE

NUMBER OF FLAGS EXPECTED = 2.

X-numbers:    82,83

SQ401M      Tests the flagging of High subset features that are used in sequential input-output.

The following features are tested:

- 1 )    I-O CONTROL
- 2 )    SAME RECORD AREA
- 3 )    MULTIPLE FILE TAPE (OBS)
- 4 )    BLOCK CONTAINS integer TO
- 5 )    RECORD VARYING
- 6 )    LINAGE
- 7 )    VALUE OF data name (OBS)
- 8 )    CLOSE...FOR REMOVAL  
              WITH NO REWIND/LOCK
- 9 )    OPEN...REVERSED  
              WITH NO REWIND  
              EXTEND
- 10)    READ...NEXT
- 11)    WRITE...AT END-OF-PAGE

NUMBER OF FLAGS EXPECTED = 18.

X-numbers:    02,03,08,74,82,83,86

#### OBSOLETE

IX302M      Tests the flagging of obsolete features that are used in intermediate subset indexed input-output.

The following features are tested:

- 1 )    RERUN
- 2 )    LABEL RECORDS
- 3 )    VALUE OF
- 4 )    DATA RECORDS

NUMBER OF FLAGS EXPECTED = 4.

X-numbers:    13,24,53,74,75,82,83

NC302M      Tests the flagging of obsolete minimum subset nucleus features.

The following features are tested:

- 1 )    AUTHOR/INSTALLATION/DATE-WRITTEN/SECURITY  
              paragraphs
- 2 )    Memory size
- 3 )    ALTER procedure-name TO PROCEED TO procedure-name
- 4 )    STOP + literal

NUMBER OF FLAGS EXPECTED = 7.

X-numbers:    68,82,83

NC303M      Tests the flagging of obsolete features that are used in the high subset nucleus.

The following features are tested:

- 1 )    DATE-COMPILED paragraph
- 2 )    GO TO without procedure-name (2 times)
- 5 )    ALTER procedure-name series

NUMBER OF FLAGS EXPECTED = 4.

X-numbers:    82,83

RL302M      Tests the flagging of obsolete features that are used in intermediate subset relative input-output.

The following features are tested:

- 1 )    RERUN
- 2 )    LABEL RECORDS
- 3 )    VALUE OF
- 4 )    DATA RECORDS

NUMBER OF FLAGS EXPECTED = 4.

X-numbers:    13,21,53,74,75,82,83

SQ302M      Tests the flagging of obsolete features that are used in minimum subset sequential input-output.

The following features are tested:

- 1 )    RERUN
- 2 )    LABEL RECORDS clause
- 3 )    VALUE OF clause
- 4 )    DATA RECORDS clause

NUMBER OF FLAGS EXPECTED = 4.

X-numbers:    13,14,53,74,75,82,83

SQ303M      Tests the flagging of obsolete features that are used in high subset sequential input-output.

The following features are tested:

- 1 )    Multiple FILE TAPE
- 2 )    OPEN REVERSED

NUMBER OF FLAGS EXPECTED = 2.

X-numbers:    08,14,82,83

OPTIONAL

CM303M Tests the flagging of obsolete features that are used in communications.

The following features are tested:

- 1 ) ENABLE
- 2 ) DISABLE

NUMBER OF FLAGS EXPECTED = 2.

X-numbers: 82,83

CM401M Tests the flagging of level 2 features that are used in communications.

The following features are tested:

- 1 ) CD...INITIAL
- 2 ) SYMBOLIC SUB-QUEUE
- 3 ) DESTINATION TABLE
- 4 ) DISABLE
- 5 ) ENABLE
- 6 ) PURGE
- 7 ) SEND...FROM identifier
- 8 ) SEND...WITH identifier
- 9 ) SEND...WITH ESI
- 10) SEND...REPLACING LINE

NUMBER OF FLAGS EXPECTED = 10.

X-numbers: 82,83

DB301M Tests the flagging of level 1 features that are used in debugging.

The following feature is tested:

- 1 ) USE FOR DEBUGGING ON ALL PROCEDURES

NUMBER OF FLAGS EXPECTED = 1.

X-numbers: 14,82,83

DB302M Tests the flagging of obsolete features that are used in debugging level 1.

The following features are tested:

- 1 ) USE FOR DEBUGGING ON procedure-name

NUMBER OF FLAGS EXPECTED = 1.

X-numbers: 14,82,83

- DB303M      Tests the flagging of obsolete features that are used in debugging level 2.
- The following features are tested:
- 1 )    USE FOR DEBUGGING ON ALL REFERENCES  
2 )    USE FOR DEBUGGING ON file-name
- NUMBER OF FLAGS EXPECTED = 2.
- X-numbers:    14,82,83
- DB304M      Tests the flagging of obsolete communication features used in debugging.  
  
THIS TEST NEED NOT BE EXECUTED IF COMMUNICATIONS ARE NOT IMPLEMENTED.
- The following feature is tested:
- 1 ) USE FOR DEBUGGING ON cd-name
- NUMBER OF FLAGS EXPECTED = 1.
- X-numbers:    82,83
- DB305M      Tests the flagging of obsolete communication features used in debugging.
- The following feature is tested:
- 1 ) USE FOR DEBUGGING ON ALL PROCEDURES
- NUMBER OF FLAGS EXPECTED = 1.
- X-numbers:    14,82,83
- RW301M      Tests the flagging of features that are used in report writing.
- The following features are tested:
- 1 )    REPORT clause  
2 )    REPORT section  
3 )    RD entry  
4 )    TYPE  
5 )    SOURCE  
6 )    COLUMN  
7 )    LINE  
8 )    INITIATE  
9 )    GENERATE  
10)   TERMINATE
- NUMBER OF FLAGS EXPECTED = 10.
- X-numbers:    01,02,74,75,82,83

RW302M      Tests the flagging of obsolete features that are used in report writing.

The following features are tested:

- 1 )    LABEL RECORDS
- 2 )    VALUE OF
- 3 )    MULTIPLE FILE TAPE

NUMBER OF FLAGS EXPECTED = 3.

X-numbers:    01,02,74,75,82,83

SG302M      Tests the flagging of obsolete features that are used in segmentation level 1.

The following feature is tested:

- 1 )    SECTION + segment number

NUMBER OF FLAGS EXPECTED = 1.

X-numbers:    82,83

SG303M      Tests the flagging of obsolete features that are used in segmentation level 2.

The following features are tested:

- 1 )    SEGMENT-LIMIT
- 2 )    Sections with the same SEGMENT no

NUMBER OF FLAGS EXPECTED = 4.

X-numbers:    82,83

SG401M      Tests the flagging of level 2 features that are used in segmentation.

The following features are tested:

- 1 )    SEGMENT-LIMIT
- 2 )    Sections with the same segment number

NUMBER OF FLAGS EXPECTED = 2.

X-numbers:    82,83



## APPENDIX A

### SUBSETS OF FEDERAL STANDARD COBOL

| MODULES          |                             | LEVELS OF REQUIRED MODULES |              |                 |
|------------------|-----------------------------|----------------------------|--------------|-----------------|
| Required Modules |                             | Minimum                    | Intermediate | SUBSETS<br>High |
| NC               | Nucleus                     | 1                          | 1            | 2               |
| SQ               | Sequential I-O              | 1                          | 1            | 2               |
| RL               | Relative I-O                | -                          | 1            | 2               |
| IX               | Indexed I-O                 | -                          | 1            | 2               |
| IC               | Inter-program Communication | 1                          | 1            | 2               |
| ST               | Sort-Merge                  | -                          | 1            | 1               |
| SM               | Source Text Manipulation    | -                          | 1            | 2               |
| IF               | Intrinsic Functions         | -                          | -            | 1               |

#### Optional Modules

These modules are optional in Federal Standard COBOL and are not therefore part of a particular subset.

|    |                |               |
|----|----------------|---------------|
| RW | Report Writer  | Level 1       |
| CM | Communications | Level 1 and 2 |
| DB | Debug          | Level 1 and 2 |
| SG | Segmentation   | Level 1 and 2 |

Combination of Flagging Programs and Compiler Switches for each Implemented Subset or Implemented Optional Module.

| <u>IMPLEMENTATION LEVEL</u> | <u>COMPILER OPTION SWITCH</u> | <u>PROGRAMS</u>                                  |                                      |
|-----------------------------|-------------------------------|--------------------------------------------------|--------------------------------------|
| Minimum                     | Obsolete                      | NC302M                                           | SQ302M                               |
| Intermediate                | Above Minimum                 | IX301M<br>SM301M                                 | RL301M<br>ST301M                     |
|                             | Obsolete                      | NC302M<br>RL302M                                 | IX302M<br>SQ302M                     |
| High                        | Above Minimum                 | IX301M<br>SM301M                                 | RL301M<br>ST301M                     |
|                             | Above Intermediate            | IC401M<br>IF402M<br>IX401M<br>RL401M<br>SQ401M   | IF401M<br>IF403M<br>NC401M<br>SM401M |
|                             | Obsolete                      | IX302M<br>NC303M<br>SQ302M                       | NC302M<br>RL302M<br>SQ303M           |
| Communication Lev 1         |                               | (NO TESTS)                                       |                                      |
| Communication Lev 2         | Above 1 COM                   | CM401M                                           |                                      |
|                             | Obsolete                      | CM303M                                           |                                      |
| Debug Level 1               | 1 DEB                         | DB301M                                           |                                      |
|                             | Obsolete                      | DB302M                                           |                                      |
| Debug Level 2               | 1 DEB                         | DB301M                                           |                                      |
|                             | Obsolete                      | DB302M<br>DB304M (if 1COM implemented)<br>DB305M | DB303M                               |
| Report Writer               | RPW                           | RW301M                                           |                                      |
|                             | Obsolete                      | RW302M                                           |                                      |
| Segmentation Level 1        | Obsolete                      | SG302M                                           |                                      |
| Segmentation Level 2        | Above 1 SEG                   | SG401M                                           |                                      |
|                             | Obsolete                      | SG302M                                           | SG303M                               |

## APPENDIX B

### TABLE OF PROGRAM DEPENDENCIES

#### Main Program      Dependent Programs

##### Inter-Program Communications Module

|        |                        |
|--------|------------------------|
| IC101A | IC102A                 |
| IC103A | IC104A, IC105A         |
| IC106A | IC107A                 |
| IC108A | IC109A, IC110A, IC111A |
| IC112A | IC113A                 |
| IC114A | IC115A                 |
| IC116M | IC117M, IC118M         |
| IC201A | IC202A                 |
| IC203A | IC204A, IC205A, IC206A |
| IC207A | IC208A                 |
| IC209A | IC210A, IC211A, IC212A |
| IC213A | IC214A, IC215A         |
| IC216A | IC217A                 |
| OBIC1A | OBIC2A, OBIC3A         |

##### Indexed I-O Module

|        |                        |
|--------|------------------------|
| IX101A | IX102A, IX103A         |
| IX109A | IX110A, IX111A, IX112A |
| IX113A | IX114A..IX120A (inc)   |
| IX201A | IX202A, IX203A         |

##### Relative I-O Module

|        |                |
|--------|----------------|
| RL101A | RL102A, RL103A |
| RL108A | RL109A, RL110A |
| RL201A | RL202A, RL203A |
| RL206A | RL207A, RL208A |
| RL212A | RL213A         |

##### Source Text Manipulation Module

|        |        |
|--------|--------|
| SM101A | SM102A |
| SM103A | SM104A |
| SM201A | SM202A |
| SM203A | SM204A |

##### Sequential I-O Module

|        |                |
|--------|----------------|
| SQ202A | SQ203A         |
| OBSQ3A | OBSQ4A, OBSQ5A |

**Sort-Merge Module**

|        |                |
|--------|----------------|
| ST101A | ST102A, ST103A |
| ST104A | ST105A         |
| ST106A | ST107A         |
| ST109A | ST110A, ST111A |
| ST112M | ST113M, ST114M |
| ST115A | ST116A, ST117A |
| ST119A | ST120A, ST121A |
| ST122A | ST123A, ST124A |
| ST125A | ST126A         |

## APPENDIX C

### TEMPORARY PROGRAM FIXES

(V4.2-3)

#### 1. INTRODUCTION

This is a supplement to the 1985 CCVS X/Open Extension User Guide(Addendum to CCVS85 User Guide) Version 4.2, containing the updates for correcting the errors in that version of the X/Open Extension tests version 4.2. The updates, referred to as Temporary Program Fixes (TPFs) are presented in the format required for use by the EXEC85 program, and will be applied in conducting a formal validation. The TPFs are presented in this document in the same order as the programs appear on the CCVS85 Population File, to facilitate preparing the updates for use.



2. CORRECTIONS FOR X/OPEN EXTENSION TEST PROGRAMS VERSION 4.2

2.1 Program Name: XC008A

a. Description of Problem

This program does not conform to the standard or to the X/Open Extension specifications concerning the use of the RECORD VARYING clause. Also the program as currently coded would result in file attribute conflicts. The program is withdrawn until modified to conform to both the standard and X/Open Extension specifications.

2.2 Program Name: XE002A

a. Description of Problem

This program made reference to an incorrect program name.

b. Update

**\*START,XE002A  
020600 "XE002A".**

2.3 Program Name: XE003A

a. Description of Problem

This program made reference to an incorrect program name.

b. Update

**\*START,XE003A  
020800 "XE003A".**

2.4 Program Name: XH002A

a. Description of Problem

This program made reference to unnamed programs. The program now references test programs involved in testing file sharing and record locking.

b. Update

**\*START,XH002A  
047400- "XH002A".**



2.5 Program Name: XH027A

a. Description of Problem

This program made reference an incorrect program name.

b. Update

**\*START,XH027A  
024700 "XH027A".**

2.6 Program Name: XH057A

a. Description of Problem

This program made reference to unnamed programs. The program now references test programs involved in testing file sharing and record locking.

b. Update

**\*START,XH057A  
009400 DISPLAY "XH057A:Semaphore 1 in state 1, subject file  
010100 DISPLAY "XH057A:Wait until Semaphore 1 is in state 2".**

2.7 Program Name: XH058A

a. Description of Problem

This program made reference to unnamed programs. The program now references test programs involved in testing file sharing and record locking.

b. Update

**\*START,XH058A  
038000 DISPLAY "XH058A:Wait until Semaphore 1 is in state 1".  
039400 "XH058A : Semaphore 1 in state 1, XH057A".  
047200 DISPLAY "XH058A: Semaphore 1 in state 2".**

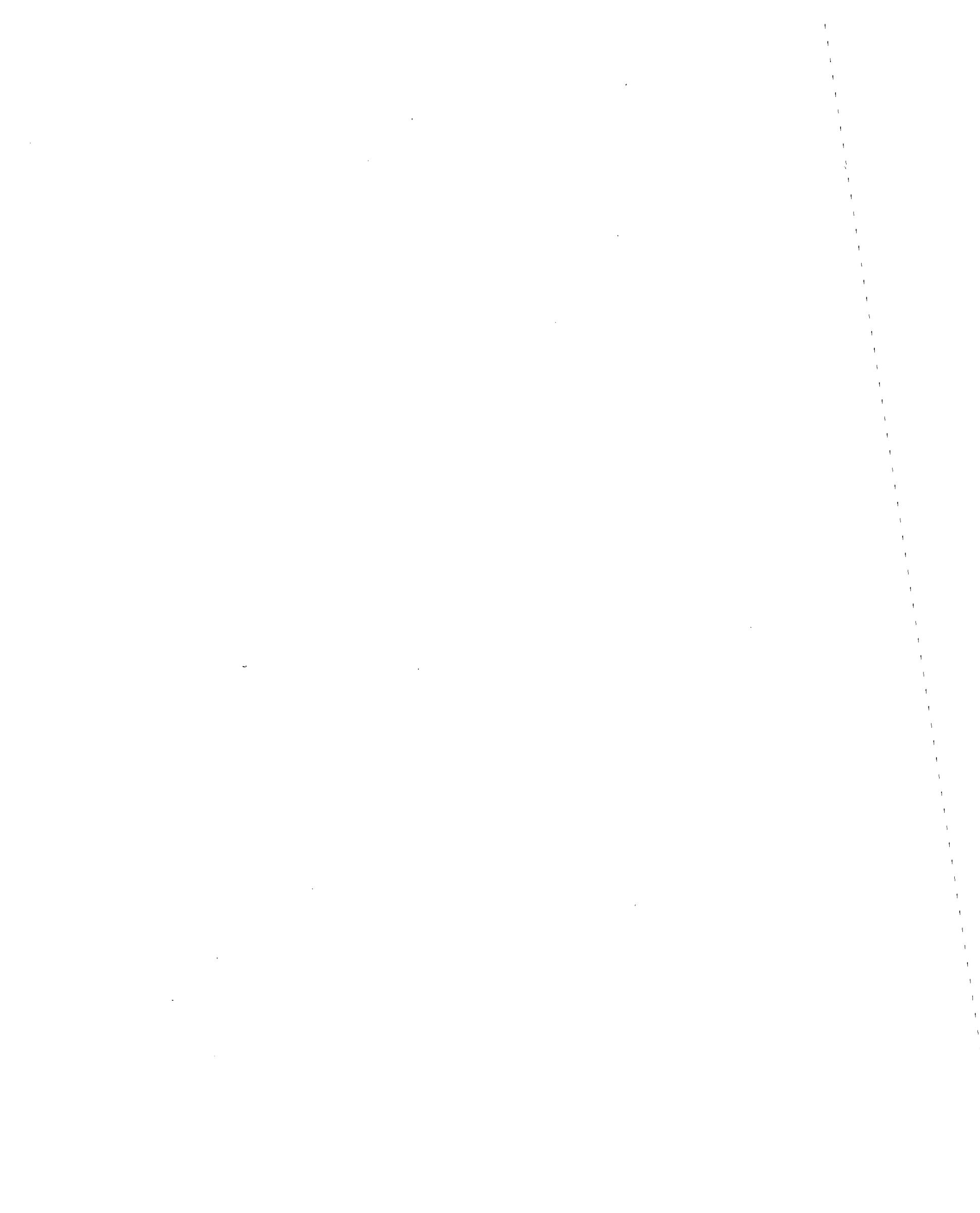
2.8 Program Name: XH059A

a. Description of Problem

This program made reference to unnamed programs. The program now references test programs involved in testing file sharing and record locking.

b. Update

**\*START,XH059A  
009100 DISPLAY "XH059A:Semaphore 1 in state 1, subject file  
009800 DISPLAY "XH059A : Wait until semaphore 1 is in state 2".**



2.9 Program Name: XH064A

a. Description of Problem

This program made reference to unnamed programs. The program now references test programs involved in testing file sharing and record locking.

b. Update

```
*START,XH064A
008600 DISPLAY "XH064A :Semaphore 1 in state 1".
008900 DISPLAY "XH064A : Wait until Semaphore 1 is in state 2".
010800 DISPLAY "XH064A: Set semaphore 1 to state 3".
011200 DISPLAY "XH064A: Wait until semaphore 1 is in state 4".
```

2.10 Program Name: XP007A

a. Description of Problem

Some program names referenced in this program were in lower case. They are now in upper case.

b. Update

```
*START,XP007A
002900 01 PROG-005-NAME PIC X(6) VALUE "XP005A".
003000 01 PROG-006-NAME PIC X(6) VALUE "XP006A".
```



The following is a composite of the source code represented by the TPFs contained in this document.

Card Columns:

```
0.....1.....2.....3.....4.....5.....6.....7.....8
*BEGIN-UPDATE
*START,XE002A
020600 "XE002A".
*START,XE003A
020800 "XE003A".
*START,XH002A
047400- "XH002A".
*START,XH027A
024700 "XH027A".
*START,XH057A
009400 DISPLAY "XH057A:Semaphore 1 in state 1, subject file
010100 DISPLAY "XH057A:Wait until Semaphore 1 is in state 2".
*START,XH058A
038000 DISPLAY "XH058A:Wait until Semaphore 1 is in state 1".
039400 "XH058A : Semaphore 1 in state 1, XH057A".
047200 DISPLAY "XH058A: Semaphore 1 in state 2 ".
*START,XH059A
009100 DISPLAY "XH059A:Semaphore 1 in state 1, subject file
009800 DISPLAY "XH059A : Wait until semaphore 1 is in state 2".
*START,XH064A
008600 DISPLAY "XH064A :Semaphore 1 in state 1".
008900 DISPLAY "XH064A : Wait until Semaphore 1 is in state 2".
010800 DISPLAY "XH064A: Set semaphore 1 to state 3".
011200 DISPLAY "XH064A: Wait until semaphore 1 is in state 4".
*START,XP007A
002900 01 PROG-005-NAME PIC X(6) VALUE "XP005A".
003000 01 PROG-006-NAME PIC X(6) VALUE "XP006A".
*END-UPDATE
```



## APPENDIX D

### X-CARDS

| X-card | Function                                | Explanation                                                                                                                                 |
|--------|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| X-01   | Tape name 1                             | Implementor-name in ASSIGN TO clause of SELECT statement<br>e.g. SELECT file-name ASSIGN TO XXXXX001.<br>X-01 UNISERVO TAPE-1               |
| X-02   | Tape name 2                             | " "                                                                                                                                         |
| X-03   | Tape name 3                             | " "                                                                                                                                         |
| X-04   | Tape name 4                             | " "                                                                                                                                         |
| X-05   | Tape name 5                             | " "                                                                                                                                         |
| X-06   | Multireel name 1                        | Implementor-name is ASSIGN TO clause, 3 reels.                                                                                              |
| X-07   | Multireel name 2                        | Implementor-name in ASSIGN TO clause, 2 reels.                                                                                              |
| X-08   | Multifile-reel name 1                   | Implementor-name in ASSIGN TO clause of SELECT statement; all files are assigned to the same physical reel of tape                          |
| X-09   | Multifile-reel name 2                   | " "                                                                                                                                         |
| X-10   | Multifile-reel name 3                   | " "                                                                                                                                         |
| X-11   | Multifile-reel name 4                   | " "                                                                                                                                         |
| X-12   | Multifile-reel name 5                   | " "                                                                                                                                         |
| X-13   | Multifile-reel name 6                   | " "                                                                                                                                         |
| X-14   | Sequential mass storage name 1          | Implementor-name in ASSIGN TO clause of SELECT statement; Organisation is sequential.<br>Assigned names must reference non reel/unit media. |
| X-15   | Sequential mass storage name 2          | " "                                                                                                                                         |
| X-16   | Sequential mass storage name 3          | " "                                                                                                                                         |
| X-17   | Optional multireel tape name 3          | Implementor-name in ASSIGN TO clause of SELECT OPTIONAL statement for a three reel file; this file should not be present at execution time. |
| X-18   | Optional sequential mass storage name 4 | Implementor-name in ASSIGN TO clause of SELECT OPTIONAL statement; this file should not be present at execution time.                       |
| X-19   | Multi-unit name 1                       | Implementor-name in ASSIGN TO clause of SELECT statement;                                                                                   |
| X-20   | Multi-unit name 2                       | 3 mass storage units each for X-19 and X-20; used with CLOSE UNIT statements.                                                               |
| X-21   | Relative file name 1                    | Implementor-name in ASSIGN TO clause of SELECT statement; organisation is relative.                                                         |
| X-22   | Relative file name 2                    | " "                                                                                                                                         |
| X-23   | Relative file name 3                    | " "                                                                                                                                         |
| X-24   | Indexed file name 1                     | Implementor-name in ASSIGN TO clause of SELECT statement; organisation is indexed. (See X-44, X-45 and X-46)                                |
| X-25   | Indexed file name 2                     | " "                                                                                                                                         |

|      |                                                                       |                                                                                                                                                |
|------|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| X-26 | Indexed file name 3                                                   | " "                                                                                                                                            |
| X-27 | Sort file name 1                                                      | implementor-name in ASSIGN TO clause of SELECT statement for sort files.                                                                       |
| X-28 | Sort file name 2                                                      | " "                                                                                                                                            |
| X-29 | Sort file name 3                                                      | " "                                                                                                                                            |
| X-30 | Input queue name for terminal 1<br>(up to 12 characters)              | The information in cards X-30 through X-43 is in the form of a non-numeric literal enclosed in quotes.                                         |
| X-31 | System password for input CD to terminal 1<br>(up to 10 characters)   | " "                                                                                                                                            |
| X-32 | Symbolic destination name for terminal 1<br>(up to 12 characters)     | " "                                                                                                                                            |
| X-33 | System password for output CD for terminal 1<br>(up to 10 characters) | " "                                                                                                                                            |
| X-34 | Input queue name for terminal 2<br>(up to 12 characters)              | " "                                                                                                                                            |
| X-35 | Symbolic destination name for terminal 2<br>(up to 12 characters)     | " "                                                                                                                                            |
| X-36 | System password for input CD for terminal 2<br>(up to 10 characters)  | " "                                                                                                                                            |
| X-37 | System password for output CD for terminal 2<br>(up to 10 characters) | " "                                                                                                                                            |
| X-38 | Queue name for PPPP messages (program CM105)<br>(48 characters)       | " "                                                                                                                                            |
| X-39 | Queue name for PPPS messages (program CM105)<br>(48 characters)       | " "                                                                                                                                            |
| X-40 | Queue name for PPSP messages (program CM105)<br>(48 characters)       | " "                                                                                                                                            |
| X-41 | Queue name for PSPP messages (program CM105)<br>(48 characters)       | " "                                                                                                                                            |
| X-42 | Name of first symbolic source<br>(up to 12 characters)                | The value of the literal must not equal "GARBAGE".                                                                                             |
| X-43 | Name of second symbolic source<br>(up to 12 characters)               |                                                                                                                                                |
| X-44 | System name for indexed file name 1                                   | Implementor-name for those systems which require an additional system interface for the indices in an indexed file (see *OPT10 J, Appendix C). |
| X-45 | System name for indexed file name 2                                   | " "                                                                                                                                            |
| X-46 | System name for indexed file name 3                                   | " "                                                                                                                                            |

|      |                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|------|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X-47 | Copy library name 1                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| X-48 | Copy library name 2                  | Referenced in COPY statement for alternate library.                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| X-49 | Sequential file<br>for a Report File | (Report Writer)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| X-50 | Sequential file<br>for a Report File | (Report Writer)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| X-51 | Switch name 1                        | Implementor-name in SPECIAL-NAMES paragraph; this switch should be ON at execution time.                                                                                                                                                                                                                                                                                                                                                                                                        |
| X-52 | Switch name 2                        | Implementor-name in SPECIAL-NAMES paragraph; this switch should be OFF at execution time.                                                                                                                                                                                                                                                                                                                                                                                                       |
| X-53 | RERUN statement                      | Complete entry in the I-O-CONTROL paragraph; eg,<br>RERUN ON SQ-FRR EVERY 10 RECORDS OF RR-FS1.<br><br>SQ-FRR is a file provided in each of the audit routines which uses the rerun facility. It is defined but not referenced in the Procedure Division. It is strictly for use in the rerun statement if a file-name option is required by the implementation.                                                                                                                                |
|      |                                      | RR-FS1 is the file which is accessed in the Procedure Division. It is the same name in each program which audits the rerun facility. Rerun information is written on SQ-FRR whenever 10 records of RR-FS1 have been processed.                                                                                                                                                                                                                                                                  |
| X-54 | System punch                         | Implementor-name is ASSIGN TO clause of SELECT statement.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| X-55 | System printer                       | Implementor-name for output listings in ASSIGN TO clause of SELECT statement.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| X-56 | DISPLAY mnemonic name                | Implementor-name in SPECIAL-NAMES paragraph.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| X-57 | ACCEPT mnemonic name                 | Implementor-name in SPECIAL-NAMES paragraph.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| X-58 | System input device                  | Implementor-name in ASSIGN TO clause of SELECT statement.                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| X-59 | Alphabet-name                        | Implementor-name to be used as an alphabet-name.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| X-60 | Sequential mass storage name 4       | Implementor-name in ASSIGN TO clause of SELECT                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| X-61 | Relative file name 4                 | Implementor-name in ASSIGN TO clause of SELECT statement; organisation is relative.                                                                                                                                                                                                                                                                                                                                                                                                             |
| X-62 | Relative file name 6                 | Implementor-name in ASSIGN TO clause of SELECT statement; organisation is relative.                                                                                                                                                                                                                                                                                                                                                                                                             |
| X-63 |                                      | A 51 character alphanumeric literal containing 50 of the characters of the COBOL character set. The characters are in ascending order according to the native collating sequence; the quotation mark ("") is excluded and the currency symbol (\$) is repeated twice. The literal is bounded by quotation marks and terminated by a period. For a system whose native collating sequence is ASCII, the following literal is used:<br><br>" \$\$(*+,-./0123456789;<=>ABCDEFGHIJKLMNPQRSTUVWXYZ". |

|      |                                  |                                                                                                                                                                                                                                                                                                                                                                                               |
|------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X-64 |                                  | A 51 character alphanumeric literal made up of any 51 characters. The characters are in descending order according to the native collating sequence. The literal is bounded by quotation marks and terminated by a period. The quote character is not to be used in the value for X-64 and the dollar sign character is to appear twice wherever it belongs in the native collating sequence. |
| X-65 |                                  | Number of records generated and sorted in ST115A-ST117A. For an official validation this four digit integer must be 9000.                                                                                                                                                                                                                                                                     |
| X-66 | Relative file name 5             | Implementor-name is ASSIGN TO clause of SELECT statement; organisation is relative.                                                                                                                                                                                                                                                                                                           |
| X-67 | Memory size in words             | Implementor supplied integer specified in the MEMORY SIZE clause of the OBJECT-COMPUTER paragraph.                                                                                                                                                                                                                                                                                            |
| X-68 | Memory size in characters        | " "                                                                                                                                                                                                                                                                                                                                                                                           |
| X-69 |                                  | Additional VALUE OF phrase for each FD containing a VALUE OF phrase. This image is optionally generated; see *OPT9 card, Appendix C.                                                                                                                                                                                                                                                          |
| X-70 | Record Delimiter                 | Implementor-name for RECORD DELIMITER clause in FD. The value of implementor-name must not be STANDARD-1.                                                                                                                                                                                                                                                                                     |
| X-71 | VALUE OF phrase 8                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-72 | VALUE OF phrase 7                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-73 | Implementor-name for top of page | Implementor-name defined in SPECIAL-NAMES paragraph, associated with mnemonic-name in WRITE ADVANCING statements.                                                                                                                                                                                                                                                                             |
| X-74 | VALUE OF implementor-name        | In VALUE OF clause for standard labels; eg,<br>VALUE OF<br>XXXXXX074<br>IS<br>XXXXX075 (or 076, 077, 078, 079, 080, 071,072).                                                                                                                                                                                                                                                                 |
| X-75 | VALUE OF phrase 1                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-76 | VALUE OF phrase 2                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-77 | VALUE OF phrase 3                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-78 | VALUE OF phrase 4                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-79 | VALUE OF phrase 5                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-80 | VALUE OF phrase 6                | (See X-74.)                                                                                                                                                                                                                                                                                                                                                                                   |
| X-81 |                                  | An eight character alphanumeric literal containing eight unique characters which are contained in the native character set but are not contained in the 51 character COBOL character set. The literal is bounded by quotation marks and terminated by a period.                                                                                                                               |
| X-82 | Source computer name             | Implementor-name defined in SOURCE-COMPUTER paragraph.                                                                                                                                                                                                                                                                                                                                        |
| X-83 | Object computer name             | Implementor-name defined in OBJECT-COMPUTER paragraph.                                                                                                                                                                                                                                                                                                                                        |

LABEL records option for the printer destined file used by each of the audit routines. If not specified, OMITTED will be assumed. If the word STANDARD appears, additional information may also be required in this entry (ie VALUE OF, CODE SET IS,...).

The following example is provided in order to explain X-85 through X-88:

|      | Before running executive routine                                                                                                                                                                       | After running executive routine                                                                                                                                                                                                                            |
|------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | FD      file-name-1<br>VALUE OF<br>XXXXX074<br>IS data-name-1<br>XXXXX085<br>IS data-name-2.                                                                                                           | FD      file-name-1<br>VALUE OF<br>FILE-ID<br>IS data-name-1<br>RETENTION-DATE<br>IS data-name-2.                                                                                                                                                          |
|      | FD      file-name-2<br>VALUE OF<br>XXXXX074<br>IS<br>XXXXX075<br>xxxxx085<br>IS data-name-3.<br>01      data-name-1<br>XXXXX086<br>01      data-name-2<br>XXXXX087<br>01      data-name 3<br>XXXXX088. | FD      file-name-2<br>VALUE OF<br>FILE-ID<br>IS<br>"FILE 2"<br>RETENTION DATE<br>IS data-name-3.<br>01      data-name-1<br>PIC X(6) VALUE "FILE 1".<br>01      data-name-2<br>PIC X(6) VALUE "870131".<br>01      data-name-3<br>PIC X(6) VALUE "870228". |
| X-85 | VALUE OF implementor-name                                                                                                                                                                              | Second implementor-name in the VALUE OF clause in File Description entry for standard labels.                                                                                                                                                              |
| X-86 | PICTURE description and VALUE clause                                                                                                                                                                   | Data Description entry for data-name-1 associated with VALUE OF clause in File Description entry.                                                                                                                                                          |
| X-87 | PICTURE description and VALUE clause                                                                                                                                                                   | Data Description entry for data-name-2 associated with VALUE OF clause in File Description entry.                                                                                                                                                          |
| X-88 | PICTURE description and VALUE clause                                                                                                                                                                   | Data Description entry for data-name-3 associated with VALUE OF clause in File Description entry.                                                                                                                                                          |
| X-90 | Ordinal number of character 'A' in native character set                                                                                                                                                | Used in CLASS clause in SPECIAL-NAMES paragraph.                                                                                                                                                                                                           |
| X-91 | Ordinal number of character 'D' in native character set                                                                                                                                                | Used in CLASS clause in SPECIAL-NAMES paragraph.                                                                                                                                                                                                           |
| X-92 | Non-existent test-file-name                                                                                                                                                                            | Implementor defined format for file which will not exist at run time (RL119A).                                                                                                                                                                             |



## APPENDIX E

### SAMPLE EXECUTIVE INPUT DATA (CONTROL-CARD-FILE)

```
*EXTRACT-ALL
*LIST X-CARDS
*LIST COMPACT
*OPT07 Z
*OPT08 C
*OPT11 U
X-001 TAPE1.
X-002 TAPE2.
X-003 TAPE3.
X-004 TAPE4.
X-005 TAPE5.
X-006 MULTR1.
X-007 MULTR2.
X-008 MULTF1.
X-009 MULTF2.
X-010 MULTF3.
X-014 MASS1.
X-015 MASS2.
X-016 MASS3.
X-017 MULTR3.
X-018 MASS4.
X-019 MULTU1.
X-020 MULTU2.
X-021 REL1.
X-022 REL2.
X-023 REL3.
X-024 INDEX1.
X-025 INDEX2.
X-026 INDEX3.
X-02706 SORT1XXXXXX.
X-02806 SORT2XXXXXX.
X-02906 SORT3XXXXXX.
X-047 COPYLIB.
X-051 SWITCH1.
X-052 SWITCH2.
X-055 PRINTER.
X-056 MYTERM.
X-057 MYTERM.
X-058 SYSIN.
X-063 "$$()*+,-./0123456789;<=>ABCDEFGHIJKLMNOPQRSTUVWXYZ".
X-064 "ZYXWVUTSRQPNMLKJIHGFEDCBA>=<;9876543210/-,+*)($$ ".
X-065 500.X-081 "@&%!{?}".
*END-MONITOR
*BEGIN-UPDATE
*END-UPDATE
```



## APPENDIX F

### SAMPLE PROGRAM FOR EXTRACTING THE AUDIT ROUTINES FROM THE POPULATION FILE

IDENTIFICATION DIVISION.

PROGRAM-ID. CREATECOB85.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

  SELECT COBFILES

    ASSIGN TO DISK.

  SELECT POPFILES

    ASSIGN TO "POPFILE.COBS".

DATA DIVISION.

FILE SECTION.

FD COBFILES

  VALUE OF "FILE-NAME".

01 SOURCEOUT           PIC X(80).

FD POPFILES.

01 SOURCEIN.

  03 FILLER           PIC X(6).

  03 HEADER   PIC X(7).

  03 FILLER           PIC X.

  03 FTYPE           PIC X(5).

  03 FILLER           PIC X.

  03 FILEIN           PIC X(6).

  03 FILLER           PIC X.

  03 SUB           PIC XXX.

  03 FILLER           PIC X(4)..

  03 SUBPRO           PIC X(6).

  03 FILLER           PIC X(40).

WORKING-STORAGE SECTION.

01 FILE-NAME.

  02 FILE-NAME       PIC X(6).

  02 FILEEXT   PIC X(4) VALUE ".COB".

PROCEDURE DIVISION.

0.

  OPEN INPUT POPFILES.

  MOVE ".COB" TO FILEEXT.

  MOVE "DUMMY" TO FILENAME.

  OPEN OUTPUT COBFILES.

1.

  MOVE SPACE TO SOURCEIN.

  READ POPFILES AT END GO TO DONE.

  IF HEADER = "\*HEADER" PERFORM OPEN-IT.

  MOVE SOURCEIN TO SOURCEOUT.

  WRITE SOURCEOUT.

  GO TO 1.

OPEN-IT.  
CLOSE COBFILES.  
MOVE ".COB" TO FILEEXT.  
IF FTYPE = "DATA"  
    MOVE ".DAT" TO FILEEXT.  
IF FTYPE = "CLBRY"  
    MOVE ".LIB" TO FILEEXT.  
MOVE FILEIN TO FILENAME.  
IF SUB = "SUB"  
    MOVE SUBPRO TO FILENAME.  
OPEN OUTPUT COBFILES.  
DISPLAY "Opening" FILENAME.  
DONE.  
STOP RUN.

## **APPENDIX G**

### **ERROR PROCEDURE**

If you should discover an error in the COBOL Compiler Validation Service software we would be grateful if you would complete the form overleaf and return it to one of the contacts listed below:

Dave Bamber  
NCC Open Systems  
The National Computing Centre Ltd  
Oxford Road  
Manchester  
M1 7ED  
United Kingdom

E-Mail:           **cobol@ncc.co.uk**

Judy B Kailey  
Software Standards Validation Group  
National Institute of Standards and Technology  
Building 225, Room A266  
Gaithersburg MD 20899  
U.S.A.

|                          |                     |
|--------------------------|---------------------|
| SYSTEM NAME: CCVS85 V4.2 | PROGRAM NAME: _____ |
| ORGANISATION: _____      | DATE: _____         |
| ORIGINATOR NAME: _____   |                     |
| TELEPHONE NO: _____      |                     |

## SEVERITY OF ERROR:

- HIGH: SYSTEM DOES NOT FUNCTION  
 MEDIUM: SYSTEM FUNCTIONS BUT NOT SATISFACTORILY  
 LOW: SYSTEM FUNCTIONS BUT WITH MINOR IRREGULARITIES

## ERROR NARRATIVE:

|                                      |                         |
|--------------------------------------|-------------------------|
| FOR INTERNAL USE ONLY:               | ERROR REPORT NO: _____  |
| ERROR REPORT RECEIVED: _____         | RESULT: DISPUTED/UPHELD |
| ERROR TESTED: _____                  |                         |
| TPF ISSUED: _____                    |                         |
| INCORPORATED INTO CCVS85 DATE: _____ | VERSION: _____          |

SYSTEM NAME: CCVS85 V4.2

PROGRAM NAME: \_\_\_\_\_

ERROR REPORT NUMBER: \_\_\_\_\_

DATE: \_\_\_\_\_

DETAILS OF TESTING UNDERTAKEN:

FINDINGS:

ACTION TO BE TAKEN: .



## APPENDIX H

### AMENDMENT SHEET

Issued by NCC      Date: .....

SYSTEM NAME: CCVS85 V4.2

#### TITLE OF SECTION: AMENDMENT SHEET

This sheet will be updated and issued with each set of revised/new pages of the COBOL 85 Compiler Validation System User Guide to all recipients of the CCVS 85.

| AMENDMENT |      |         | DISCARD  |         |          | INSERT |
|-----------|------|---------|----------|---------|----------|--------|
| No.       | Date | Section | Issue No | Section | Issue No |        |



## APPENDIX I

### TEMPORARY PROGRAM FIXES

(V4.2-3)

#### 1. INTRODUCTION

This is a supplement to the 1985 CCVS User Guide Version 4.2, containing the updates for correcting the errors in that version of the CCVS85. The updates, referred to as Temporary Program Fixes (TPFs) are presented in the format required for use by the EXEC85 program, and will be applied in conducting a formal validation. The TPFs are presented in this document in the same order as the programs appear on the CCVS85 Population File, to facilitate preparing the updates for use.



2. CORRECTIONS FOR CCVS85 V4.2

2.1 Program Name: IX110A.

a. Description of Problem

Line 012250 states that STATUS-TEST-10 is PIC P. It should be PIC 9.

b. Update

**\*START,IX110A**

**012250        01     STATUS-TEST-10                    PIC 9 VALUE ZERO.**

2.2 Program Name: SM104A.

a. Description of Problem

When the program-id is part of the replacement by the X-card during extraction (See CCVS85 User Guide Version 3.0, Page 11), line 003800 in the extracted program requires modification to correct the expected filename to that created by SM103A. The entry on line 003800 must correspond to the entry for the X-01 card with a manual substitution of "K3FCA" for the program-id.

2.3 Program Name: SM204A.

a. Description of Problem

When the program-id is part of the replacement by the X-card during extraction (See CCVS85 User Guide Version 3.0, Page 11), line 003800 in the extracted program requires modification to correct the expected filename to that created by SM203A. The entry on line 003800 must correspond to the entry for the X-01 card with a manual substitution of "K3FCB" for the program-id.

2.4 Program Name: SQ303M.

a. Description of Problem

The OPEN...REVERSED statement in line 004300 should reference a file assigned to tape instead of mass storage.

b. Update

**\*START,SQ303M**

**001600                XXXXX001**



2.5 Program Name: SQ401M.

a. Description of Problem

OPEN OUTPUT of an OPTIONAL is illegal (line 013410)

WRITE...AT END-OF-PAGE should not be used with a non-print file.

b. Update

```
*START,SQ401M
003901 SELECT PRINTFILE ASSIGN
003902 XXXXX055.
003903
007601 FD PRINTFILE
007602 LINAGE IS 20 LINES.
007603*Message expected for above statement: NON-CONFORMING STANDARD
007604 01 FREC4 PIC X(80).
007605
013100 OPEN OUTPUT PRINTFILE.
013200 WRITE FREC4 AT END-OF-PAGE DISPLAY "HELLO".
013700*TOTAL NUMBER OF FLAGS EXPECTED = 19.
001600 XXXXX001
```



The following is a composite of the source code represented by the TPFs contained in this document.

Card Columns:

0.....1.....2.....3.....4.....5.....6.....7.....8  
\*BEGIN-UPDATE  
\*START, IX110A  
012250 01 STATUS-TEST-10 PIC 9 VALUE ZERO.  
\*START, SQ303M  
001600 XXXXX001  
\*START, SQ401M  
003901 SELECT PRINTFILE ASSIGN  
003902 XXXXX055.  
003903  
007601 FD PRINTFILE  
007602 LINAGE IS 20 LINES.  
007603\*Message expected for above statement: NON-CONFORMING STANDARD  
007604 01 FREC4 PIC X(80).  
007605  
013100 OPEN OUTPUT PRINTFILE.  
013200 WRITE FREC4 AT END-OF-PAGE DISPLAY "HELLO".  
013700\*TOTAL NUMBER OF FLAGS EXPECTED = 19.  
001600 XXXXX001  
\*END-UPDATE



1 March 1993

CCVS85 V4.2

X/OPEN EXTENSION CHECKSHEETS

COMPANY: .....

COMPILER: .....

O/S: .....

HARDWARE: .....

VSR NO: .....

DATE: .....

TESTING SPECIALIST: .....

$x^*$

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XA001M       | *1    | 1         |     |     |     |     |
| XA002M       | *1    | 1         |     |     |     |     |
| XA003M       | *1    | 1         |     |     |     |     |
| XA004M       | *1    | 1         |     |     |     |     |
| XA005M       | *1    | 1         |     |     |     |     |
| XA006M       | *1    | 1         |     |     |     |     |
| XA007M       | *1    | 1         |     |     |     |     |
| XA008M       | *1    | 1         |     |     |     |     |
| XA009M       | *1    | 1         |     |     |     |     |
| XA010M       | *1    | 1         |     |     |     |     |
| XA011M       | *1    | 1         |     |     |     |     |
| XA012M       | *1    | 1         |     |     |     |     |
| XA013M       | *1    | 1         |     |     |     |     |
| XA014M       | *1    | 1         |     |     |     |     |
| XA015M       | *1    | 1         |     |     |     |     |
| XA016M       | *1    | 1         |     |     |     |     |
| XA017M       | *1    | 1         |     |     |     |     |
| XA018M       | *1    | 1         |     |     |     |     |
| XA019M       | *1    | 1         |     |     |     |     |
| XA020M       | *1    | 1         |     |     |     |     |
| <b>TOTAL</b> |       | <b>20</b> |     |     |     |     |

\*1 This test fails if the Program can be executed.



1 March 1993

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE |
|--------------|-------|------------|-----|-----|-----|-----|
| XB001A       |       | 128        |     |     |     |     |
| XB002A       |       | 18         |     |     |     |     |
| XB003M       |       | 6          |     |     |     |     |
| XB004A       |       | 88         |     |     |     |     |
| XB005A       |       | 8          |     |     |     |     |
| XB006A       |       | 10         |     |     |     |     |
| XB007A       |       | 24         |     |     |     |     |
| XB008A       |       | 17         |     |     |     |     |
| <b>TOTAL</b> |       | <b>299</b> |     |     |     |     |

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XC008A       |       | 14        |     |     |     |     |
| <b>TOTAL</b> |       | <b>14</b> |     |     |     |     |

1 March 1993

| PROG ID      | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|--------------|-------|----------|-----|-----|-----|-----|
| XD001A       |       | 4        |     |     |     |     |
| XD002A       |       | 3        |     |     |     |     |
| <b>TOTAL</b> |       | <b>7</b> |     |     |     |     |

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XE001A       |       | 18        |     |     |     |     |
| XE002A       |       | 2         |     |     |     |     |
| XE003A       | *1    | 2         |     |     |     |     |
| XE004A       |       | 0         |     |     |     |     |
| XE005A       |       | 0         |     |     |     |     |
| <b>TOTAL</b> |       | <b>22</b> |     |     |     |     |

\*1 Calls XE004A and XE005A

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XF013M       |       | 2         |     |     |     |     |
| XF014M       |       | 1         |     |     |     |     |
| XF015M       |       | 1         |     |     |     |     |
| XF016M       |       | 3         |     |     |     |     |
| XF017M       |       | 1         |     |     |     |     |
| XF018M       |       | 1         |     |     |     |     |
| XF019M       |       | 2         |     |     |     |     |
| XF020M       |       | 3         |     |     |     |     |
| XF021M       |       | 6         |     |     |     |     |
| XF022M       |       | 1         |     |     |     |     |
| XF023M       |       | 2         |     |     |     |     |
| XF024M       |       | 4         |     |     |     |     |
| XF025M       |       | 4         |     |     |     |     |
| XF026M       |       | 3         |     |     |     |     |
| XF027M       |       | 2         |     |     |     |     |
| XF028M       |       | 2         |     |     |     |     |
| <b>TOTAL</b> |       | <b>38</b> |     |     |     |     |

1 March 1993

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| XH001A  | *1    | 0        |     |     |     |     |
| XH002A  | *2    | 4        |     |     |     |     |
| XH003A  | *1    | 0        |     |     |     |     |
| XH004A  | *2    | 4        |     |     |     |     |
| XH005A  | *1    | 0        |     |     |     |     |
| XH006A  | *2    | 4        |     |     |     |     |
| XH007A  | *1    | 0        |     |     |     |     |
| XH008A  | *2    | 8        |     |     |     |     |
| XH009A  | *1    | 0        |     |     |     |     |
| XH010A  | *2    | 8        |     |     |     |     |
| XH011A  | *1    | 0        |     |     |     |     |
| XH012A  | *2    | 8        |     |     |     |     |
| XH013A  | *1    | 0        |     |     |     |     |
| XH014A  | *2    | 16       |     |     |     |     |
| XH015A  | *1    | 0        |     |     |     |     |
| XH016A  | *2    | 16       |     |     |     |     |
| XH017A  | *1    | 0        |     |     |     |     |
| XH018A  | *2    | 16       |     |     |     |     |
| XH019A  | *1    | 0        |     |     |     |     |
| XH020A  | *2    | 1        |     |     |     |     |
| TOTAL   |       | 85       |     |     |     |     |

\*1 Execute Concurrently with the test below

\*2 Remember to delete the SYNC files before running the next two tests

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XH021A       | *1    | 0         |     |     |     |     |
| XH022A       | *2    | 1         |     |     |     |     |
| XH023A       | *1    | 0         |     |     |     |     |
| XH024A       | *2    | 1         |     |     |     |     |
| XH025A       | *1    | 3         |     |     |     |     |
| XH026A       | *2    | 10        |     |     |     |     |
| XH027A       | *1    | 1         |     |     |     |     |
| XH028A       | *2    | 10        |     |     |     |     |
| XH029A       | *1    | 1         |     |     |     |     |
| XH030A       | *2    | 10        |     |     |     |     |
| XH031A       | *1    | 0         |     |     |     |     |
| XH032A       | *2    | 8         |     |     |     |     |
| XH033A       | *1    | 0         |     |     |     |     |
| XH034A       | *2    | 8         |     |     |     |     |
| XH035A       | *1    | 0         |     |     |     |     |
| XH036A       | *2    | 3         |     |     |     |     |
| XH037A       | *1    | 0         |     |     |     |     |
| XH038A       | *2    | 3         |     |     |     |     |
| XH039A       | *1    | 0         |     |     |     |     |
| XH040A       | *2    | 3         |     |     |     |     |
| XH041A       | *1    | 0         |     |     |     |     |
| XH042A       | *2    | 9         |     |     |     |     |
| <b>TOTAL</b> |       | <b>71</b> |     |     |     |     |

\*1 Execute Concurrently with the test below

\*2 Remember to delete the SYNC files before running the next two tests

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XH043A       | *1    | 0         |     |     |     |     |
| XH044A       | *2    | 9         |     |     |     |     |
| XH045A       | *1    | 0         |     |     |     |     |
| XH046A       | *2    | 9         |     |     |     |     |
| XH047A       | *1    | 0         |     |     |     |     |
| XH048A       | *2    | 3         |     |     |     |     |
| XH049A       | *1    | 0         |     |     |     |     |
| XH050A       | *2    | 3         |     |     |     |     |
| XH051A       | *1    | 0         |     |     |     |     |
| XH052A       | *2    | 2         |     |     |     |     |
| XH053A       | *1    | 0         |     |     |     |     |
| XH054A       | *2    | 2         |     |     |     |     |
| XH055A       |       | 5         |     |     |     |     |
| XH056A       |       | 5         |     |     |     |     |
| XH057A       | *1    | 0         |     |     |     |     |
| XA058A       | *2    | 4         |     |     |     |     |
| XH059A       | *1    | 0         |     |     |     |     |
| XH060A       | *2    | 4         |     |     |     |     |
| XH061A       |       | 5         |     |     |     |     |
| XH062A       | *1    | 0         |     |     |     |     |
| XH063A       | *2    | 4         |     |     |     |     |
| <b>TOTAL</b> |       | <b>55</b> |     |     |     |     |

\*1 Execute Concurrently with the test below

\*2 Remember to delete the SYNC files before running the next two tests

1 March 1993

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| XH064A  | *1    | 0        |     |     |     |     |
| XH065A  | *2    | 7        |     |     |     |     |
| XH066A  | *1    | 0        |     |     |     |     |
| XH067A  | *2    | 7        |     |     |     |     |
| XH068A  | *1    | 0        |     |     |     |     |
| XH069A  | *2    | 5        |     |     |     |     |
| XH070A  | *1    | 0        |     |     |     |     |
| XH071A  | *2    | 4        |     |     |     |     |
| TOTAL   |       | 23       |     |     |     |     |

\*1 Execute Concurrently with the test below

\*2 Remember to delete the SYNC files before running the next two tests

1 March 1993

| PROG ID      | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|--------------|-------|----------|-----|-----|-----|-----|
| XI001A       |       | 8        |     |     |     |     |
| <b>TOTAL</b> |       | <b>8</b> |     |     |     |     |

1 March 1993

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| XM002A  |       | 1        |     |     |     |     |
| TOTAL   |       | 1        |     |     |     |     |

1 March 1993

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| XP001A       |       | 5         |     |     |     |     |
| XP002A       |       | 9         |     |     |     |     |
| XP003A       |       | 12        |     |     |     |     |
| XP004A       |       | 5         |     |     |     |     |
| XP005A       |       | 6         |     |     |     |     |
| XP006A       |       | 2         |     |     |     |     |
| XP007A       |       | 0         |     |     |     |     |
| <b>TOTAL</b> |       | <b>39</b> |     |     |     |     |

13 March 1993

CCVS85 V4.2

CHECKSHEETS

COMPANY: .....

COMPILER: .....

O/S: .....

HARDWARE: .....

VSR NO: .....

DATE: .....

TESTING SPECIALIST: .....



13 March 1993

CCVS85 V4.2

\*1 Run twice

\*2 Displays only

\*3 Flagging test

Page 1 of 31

Signed .....  
Date .....

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM   | EXE |
|---------|-------|----------|-----|-----|-------|-----|
| DB101A  | *1    | 34       |     |     |       |     |
| DB102A  |       | 14       |     |     |       |     |
| DB103M  | *2    | 14       |     |     |       |     |
| DB104A  | *3    | 12       |     |     |       |     |
| DB105A  |       | 227      |     |     |       |     |
| DB301M  | *4    | 1        |     |     | XXXXX |     |
| DB302M  | *4    | 1        |     |     | XXXXX |     |
|         |       |          |     |     | XXXXX |     |
| DB201A  | *5    | 64       | 4   |     |       |     |
| DB202A  | *6    | 24       |     |     |       |     |
| DB203A  | *7    | 20       |     |     |       |     |
| DB204A  | *8    | 4        |     |     |       |     |
| DB205A  | *9    | 15       |     |     |       |     |
| DB303M  | *4    | 2        |     |     | XXXXX |     |
| DB304M  | *4    | 1        |     |     | XXXXX |     |
| DB305M  | *4    | 1        |     |     | XXXXX |     |
|         |       |          |     |     | XXXXX |     |
|         |       |          |     |     |       |     |
|         |       |          |     |     |       |     |
| TOTAL   |       | 434      |     |     |       |     |

\*1 Expect 264 408 447 488 535 576 617 670 721 on report

\*2 Run twice

\*3 Expect 448 509 448 on report

\*4 Flagging test

\*5 Expect 407 407 571 770 883 966 1133 1307 on report

\*6 Expect 395 502 668 749 on report

\*7 Expect 501 600 on report

\*8 Expect 454 on report

\*9 Only run if CM supported - Expect 466 535 on report

Signed .....

Page 2 of 31

Date .....

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| IC101A       |       | 5         |     |     |     |     |
| IC102A       | *1    |           |     |     |     |     |
| IC103A       |       | 10        |     |     |     |     |
| IC104A       | *1    |           |     |     |     |     |
| IC105A       | *1    |           |     |     |     |     |
| IC106A       |       | 14        |     |     |     |     |
| IC107A       | *1    |           |     |     |     |     |
| IC108A       |       | 9         |     |     |     |     |
| IC109A       | *1    |           |     |     |     |     |
| IC110A       | *1    |           |     |     |     |     |
| IC111A       | *1    |           |     |     |     |     |
| IC112A       | *2    | 3         |     |     |     |     |
| IC113A       | *1    |           |     |     |     |     |
| IC114A       | *2    | 3         |     |     |     |     |
| IC115A       | *1    |           |     |     |     |     |
| IC116M       | *3    | 1         |     |     |     |     |
| IC117M       | *1    |           |     |     |     |     |
| IC118M       | *1    |           |     |     |     |     |
|              |       |           |     |     |     |     |
| IC201A       |       | 11        |     |     |     |     |
| IC202A       | *1    |           |     |     |     |     |
|              |       |           |     |     |     |     |
| <b>TOTAL</b> |       | <b>56</b> |     |     |     |     |

\*1 No report

\*2 Check SQ104A passes - see User Guide if not

\*3 3 display messages

Signed .....

Page 3 of 31

Date .....

13 March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE |
|--------------|-------|------------|-----|-----|-----|-----|
| IC203A       |       | 21         |     |     |     |     |
| IC204A       | *1    |            |     |     |     |     |
| IC205A       | *1    |            |     |     |     |     |
| IC206A       | *1    |            |     |     |     |     |
| IC207A       |       | 11         |     |     |     |     |
| IC208A       | *1    |            |     |     |     |     |
| IC209A       |       | 4          |     |     |     |     |
| IC210A       | *1    |            |     |     |     |     |
| IC211A       | *1    |            |     |     |     |     |
| IC212A       | *1    |            |     |     |     |     |
| IC213A       |       | 3          |     |     |     |     |
| IC214A       | *1    |            |     |     |     |     |
| IC215A       | *1    |            |     |     |     |     |
| IC216A       |       | 2          |     |     |     |     |
| IC217A       | *1    |            |     |     |     |     |
| IC222A       | *2    | 16         |     |     |     |     |
| IC223A       | *2    | 11         |     |     |     |     |
| IC224A       | *2    | 44         |     |     |     |     |
| IC225A       | *2    | 36         |     |     |     |     |
| IC226A       | *2    | 4          |     |     |     |     |
| IC227A       | *2    | 19         | 4   |     |     |     |
| IC228A       | *3    | 4          |     |     |     |     |
| <b>TOTAL</b> |       | <b>175</b> |     |     |     |     |

\*1 No report      \*2 Must produce 2 separate compilation units  
 \*3 Must produce a single compilation unit

Signed .....

Page 4 of 31

Date .....

13 March 1993

CCVS85 V4.2

\*1 No report \*2 Must produce 2 separate compilation units  
\*3 Must produce a single compilation unit  
\*4 Flaqging Test

Page 5 of 31

Signed .....  
Date .....

13 March 1993

CCVS85 V4.2

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| IF101A  |       | 26       |     |     |     |     |
| IF102A  |       | 13       |     |     |     |     |
| IF103A  |       | 23       |     |     |     |     |
| IF104A  |       | 27       |     |     |     |     |
| IF105A  |       | 8        |     |     |     |     |
| IF106A  |       | 30       |     |     |     |     |
| IF107A  |       | 2        |     |     |     |     |
| IF108A  |       | 10       |     |     |     |     |
| IF109A  |       | 8        |     |     |     |     |
| IF110A  |       | 9        |     |     |     |     |
| IF111A  |       | 23       |     |     |     |     |
| IF112A  |       | 8        |     |     |     |     |
| IF113A  |       | 8        |     |     |     |     |
| IF114A  |       | 23       |     |     |     |     |
| IF115A  |       | 8        |     |     |     |     |
| IF116A  |       | 25       |     |     |     |     |
| IF117A  |       | 32       |     |     |     |     |
| IF118A  |       | 13       |     |     |     |     |
| IF119A  |       | 23       |     |     |     |     |
| IF120A  |       | 17       |     |     |     |     |
| IF121A  |       | 17       |     |     |     |     |
| IF122A  |       | 17       |     |     |     |     |
| IF123A  |       | 23       |     |     |     |     |
| TOTAL   |       | 393      |     |     |     |     |

13 March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE    |
|--------------|-------|------------|-----|-----|-----|--------|
| IF124A       |       | 21         |     |     |     |        |
| IF125A       |       | 20         |     |     |     |        |
| IF126A       |       | 30         |     |     |     |        |
| IF127A       |       | 9          |     |     |     |        |
| IF128A       |       | 16         |     |     |     |        |
| IF129A       |       | 17         |     |     |     |        |
| IF130A       |       | 21         |     |     |     |        |
| IF131A       |       | 8          |     |     |     |        |
| IF132A       |       | 15         |     |     |     |        |
| IF133A       |       | 17         |     |     |     |        |
| IF134A       |       | 13         |     |     |     |        |
| IF135A       |       | 31         |     |     |     |        |
| IF136A       |       | 26         |     |     |     |        |
| IF137A       |       | 17         |     |     |     |        |
| IF138A       |       | 16         |     |     |     |        |
| IF139A       |       | 30         |     |     |     |        |
| IF140A       |       | 13         |     |     |     |        |
| IF141A       |       | 16         |     |     |     |        |
| IF142A       |       | 2          |     |     |     |        |
| IF401M       | *1    | 14         |     |     |     | XXXXXX |
| IF402M       | *1    | 17         |     |     |     | XXXXXX |
| IF403M       | *1    | 13         |     |     |     | XXXXXX |
|              |       |            |     |     |     | XXXXXX |
| <b>TOTAL</b> |       | <b>382</b> |     |     |     |        |

\*1 Flagging Test

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE |
|--------------|-------|------------|-----|-----|-----|-----|
| IX101A       |       | 2          |     |     |     |     |
| IX102A       |       | 11         |     |     |     |     |
| IX103A       |       | 12         |     |     |     |     |
| IX104A       |       | 13         |     |     |     |     |
| IX105A       | *1    | 9          |     |     |     |     |
| IX106A       |       | 10         |     |     |     |     |
| IX107A       |       | 14         |     |     |     |     |
| IX108A       |       | 32         |     |     |     |     |
| IX109A       |       | 13         |     |     |     |     |
| IX110A       |       | 4          |     |     |     |     |
| <b>TOTAL</b> |       | <b>120</b> |     |     |     |     |

\*1 Run only when implementation provides variable-length records for RECORD CONTAINS integer TO integer.

13<sup>th</sup> March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE           |
|--------------|-------|-----------|-----|-----|-----|---------------|
| IX111A       |       | 1         |     |     |     |               |
| IX112A       |       | 7         |     |     |     |               |
| IX113A       |       | 4         |     |     |     |               |
| IX114A       |       | 3         |     |     |     |               |
| IX115A       |       | 3         |     |     |     |               |
| IX116A       |       | 3         |     |     |     |               |
| IX117A       |       | 3         |     |     |     |               |
| IX118A       |       | 3         |     |     |     |               |
| IX119A       |       | 3         |     |     |     |               |
| IX120A       |       | 2         |     |     |     |               |
| IX121A       |       | 3         |     |     |     |               |
| IX301M       | *1    | 7         |     |     |     | XXXXXX        |
| IX302M       | *1    | 4         |     |     |     | XXXXXX        |
| <b>TOTAL</b> |       | <b>46</b> |     |     |     | <u>XXXXXX</u> |

\*1 Flagging test

13 March 1993

CCVS85 V4.2

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM            | EXE |
|---------|-------|----------|-----|-----|----------------|-----|
| IX201A  |       | 2        |     |     |                |     |
| IX202A  |       | 11       |     |     |                |     |
| IX203A  |       | 12       |     |     |                |     |
| IX204A  |       | 13       |     |     |                |     |
| IX205A  |       | 12       |     |     |                |     |
| IX206A  |       | 10       |     |     |                |     |
| IX207A  |       | 8        |     |     |                |     |
| IX208A  |       | 29       |     |     |                |     |
| IX209A  |       | 56       |     |     |                |     |
| IX210A  |       | 39       |     |     |                |     |
| IX211A  |       | 17       |     |     |                |     |
| IX212A  |       | 24       |     |     |                |     |
| IX213A  |       | 21       |     |     |                |     |
| IX214A  |       | 39       |     |     |                |     |
| IX215A  |       | 33       |     |     |                |     |
| IX216A  |       | 14       | 1   |     |                |     |
| IX217A  |       | 6        |     |     |                |     |
| IX218A  |       | 6        |     |     |                |     |
| IX401M  | *1    | 10       |     |     | XXXXX<br>XXXXX |     |
| TOTAL   |       | 362      |     |     |                |     |

\*1 Flagging test

| PROG ID      | NOTES | NO TESTS    | DEL | EXT | COM | EXE |
|--------------|-------|-------------|-----|-----|-----|-----|
| NC101A       |       | 93          |     |     |     |     |
| NC102A       |       | 42          |     |     |     |     |
| NC103A       |       | 102         |     |     |     |     |
| NC104A       |       | 141         |     |     |     |     |
| NC105A       |       | 129         | 3   |     |     |     |
| NC106A       |       | 126         |     |     |     |     |
| NC107A       | *1    | 177         |     |     |     |     |
| NC108M       | *1 *4 | 14          |     |     |     |     |
| NC109M       | *2    | 11          |     |     |     |     |
| NC110M       | *3    | 1           |     |     |     |     |
| NC111A       |       | 7           |     |     |     |     |
| NC112A       |       | 32          |     |     |     |     |
| NC113M       | *1    | 15          |     |     |     |     |
| NC114M       | *4    | 6           |     |     |     |     |
| NC115A       |       | 31          |     |     |     |     |
| NC116A       |       | 66          |     |     |     |     |
| NC117A       |       | 40          |     |     |     |     |
| NC118A       |       | 29          |     |     |     |     |
| NC119A       |       | 36          |     |     |     |     |
| NC120A       |       | 39          |     |     |     |     |
| NC121M       | *5    | 41          |     |     |     |     |
| <b>TOTAL</b> |       | <b>1178</b> |     |     |     |     |

\*1 Visual check

\*5 Visual check, 2 displays

\*2 Displays, op input

\*3 Displays, no report

\*4 Check listing /

Signed .....

Page 11 of 31

Date .....

| PRCG ID      | NOTES | NO TESTS    | DEL | EXT | COM | EXE |
|--------------|-------|-------------|-----|-----|-----|-----|
| NC122A       |       | 24          |     |     |     |     |
| NC123A       |       | 34          |     |     |     |     |
| NC124A       |       | 169         |     |     |     |     |
| NC125A       |       | 110         |     |     |     |     |
| NC126A       |       | 145         |     |     |     |     |
| NC127A       |       | 2           |     |     |     |     |
| NC131A       |       | 10          |     |     |     |     |
| NC132A       |       | 25          |     |     |     |     |
| NC133A       |       | 25          |     |     |     |     |
| NC134A       |       | 20          |     |     |     |     |
| NC135A       | *1    | 8           |     |     |     |     |
| NC136A       |       | 8           |     |     |     |     |
| NC137A       |       | 8           |     |     |     |     |
| NC138A       |       | 36          |     |     |     |     |
| NC139A       |       | 41          |     |     |     |     |
| NC140A       |       | 70          |     |     |     |     |
| NC141A       |       | 9           |     |     |     |     |
| NC170A       |       | 96          |     |     |     |     |
| NC171A       |       | 108         |     |     |     |     |
| NC172A       |       | 101         |     |     |     |     |
| NC173A       |       | 102         |     |     |     |     |
| NC174A       |       | 73          | 4   |     |     |     |
| <b>TOTAL</b> |       | <b>1224</b> |     |     |     |     |

\*1 Visual check

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE              |
|--------------|-------|------------|-----|-----|-----|------------------|
| NC175A       |       | 97         |     |     |     |                  |
| NC176A       |       | 124        |     |     |     |                  |
| NC177A       |       | 108        |     |     |     |                  |
| OBNC1M       | *3    | 6          |     |     |     |                  |
| NC302M       | *1    | 7          |     |     |     | XXXXXX<br>XXXXXX |
|              |       |            |     |     |     |                  |
|              |       |            |     |     |     |                  |
| NC201A       |       | 59         |     |     |     |                  |
| NC202A       |       | 77         |     |     |     |                  |
| NC203A       |       | 57         |     |     |     |                  |
| NC204M       | *3    | 15         |     |     |     |                  |
| NC205A       |       | 10         |     |     |     |                  |
| NC206A       |       | 53         |     |     |     |                  |
| NC207A       |       | 85         |     |     |     |                  |
| NC208A       |       | 24         |     |     |     |                  |
| NC209A       |       | 32         |     |     |     |                  |
| NC210A       |       | 85         |     |     |     |                  |
| NC211A       |       | 51         |     |     |     |                  |
| NC214M       | *4    | 1          |     |     |     |                  |
| NC215A       |       | 7          |     |     |     |                  |
| NC216A       |       | 57         |     |     |     |                  |
|              |       |            |     |     |     |                  |
| <b>TOTAL</b> |       | <b>955</b> |     |     |     |                  |

\*1 Flagging test

\*2 Check output

\*3 Op input, visual check

\*4 Check Julian Date

Signed .....

Page 13 of 31

Date .....

13 March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE |
|--------------|-------|------------|-----|-----|-----|-----|
| NC217A       |       | 80         | 1   |     |     |     |
| NC218A       |       | 125        |     |     |     |     |
| NC219A       |       | 9          |     |     |     |     |
| NC220M       | *5    | 25         |     |     |     |     |
| NC221A       |       | 17         |     |     |     |     |
| NC222A       |       | 8          |     |     |     |     |
| NC223A       |       | 94         |     |     |     |     |
| NC224A       |       | 14         |     |     |     |     |
| NC225A       |       | 63         |     |     |     |     |
| NC231A       |       | 24         |     |     |     |     |
| NC232A       |       | 17         |     |     |     |     |
| NC233A       |       | 14         |     |     |     |     |
| NC234A       |       | 17         |     |     |     |     |
| NC235A       |       | 13         |     |     |     |     |
| NC236A       |       | 10         |     |     |     |     |
| NC237A       |       | 13         |     |     |     |     |
| NC238A       |       | 10         |     |     |     |     |
| NC239A       |       | 8          |     |     |     |     |
| NC240A       |       | 11         |     |     |     |     |
| NC241A       |       | 11         |     |     |     |     |
| NC242A       |       | 12         |     |     |     |     |
| <b>TOTAL</b> |       | <b>595</b> |     |     |     |     |

\*5 Visual check, 2 displays

Signed .....

Page 14 of 31

Date .....

13 March 1993

CCVS85 V4.2

\*1 Switch test

\*2 Date compiled: check source

### \*3 Flagging test

Signed .....

Date \_\_\_\_\_

13 March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE   |
|--------------|-------|------------|-----|-----|-----|-------|
| RL101A       |       | 1          |     |     |     |       |
| RL102A       |       | 11         |     |     |     |       |
| RL103A       |       | 11         |     |     |     |       |
| RL104A       |       | 12         |     |     |     |       |
| RL105A       |       | 4          |     |     |     |       |
| RL106A       |       | 4          |     |     |     |       |
| RL107A       |       | 19         |     |     |     |       |
| RL108A       |       | 1          |     |     |     |       |
| RL109A       |       | 11         |     |     |     |       |
| RL110A       |       | 10         |     |     |     |       |
| RL111A       |       | 24         |     |     |     |       |
| RL112A       |       | 12         |     |     |     |       |
| RL113A       |       | 11         |     |     |     |       |
| RL114A       |       | 13         |     |     |     |       |
| RL115A       |       | 13         |     |     |     |       |
| RL116A       |       | 3          |     |     |     |       |
| RL117A       |       | 6          | 2   |     |     |       |
| RL118A       |       | 2          | 2   |     |     |       |
| RL119A       |       | 1          |     |     |     |       |
| RL301M       | *1    | 6          |     |     |     | XXXXX |
| RL302M       | *1    | 4          |     |     |     | XXXXX |
|              |       |            |     |     |     | XXXXX |
|              |       |            |     |     |     | XXXXX |
|              |       |            |     |     |     |       |
| <b>TOTAL</b> |       | <b>179</b> |     |     |     |       |

\*1 Flagging test

13 March 1993

CCVS85 V4.2

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM    | EXE    |
|---------|-------|----------|-----|-----|--------|--------|
| RL201A  |       | 1        |     |     |        |        |
| RL202A  |       | 11       |     |     |        |        |
| RL203A  |       | 11       |     |     |        |        |
| RL204A  |       | 12       |     |     |        |        |
| RL205A  |       | 66       | 1   |     |        |        |
| RL206A  |       | 501      |     |     |        |        |
| RL207A  |       | 20       |     |     |        |        |
| RL208A  |       | 11       |     |     |        |        |
| RL209A  |       | 1        |     |     |        |        |
| RL210A  |       | 1        |     |     |        |        |
| RL211A  |       | 501      |     |     |        |        |
| RL212A  |       | 1        |     |     |        |        |
| RL213A  |       | 521      |     |     |        |        |
| RL401M  | *1    | 9        |     |     | XXXXXX | XXXXXX |
| TOTAL   |       | 1667     |     |     |        |        |

\*1 Flagging test

Page 17 of 31

Signed .....  
Date .....

13 March 1993

CCVS85 V4.2

\*1 1 page of 20 lines expected

\*1 1 page of 20 lines expected  
\*2 3 pages of 20 lines expected

\*3 3 pages of 20 lines plus footer expected

\*3 3 pages of 20  
\*4 Flagging test

Signed .....

Date : .....

13 March 1993

CCVS85 V4.2

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE              |
|---------|-------|----------|-----|-----|-----|------------------|
| SG101A  |       | 151      |     |     |     |                  |
| SG102A  |       | 8        |     |     |     |                  |
| SG103A  |       | 7        |     |     |     |                  |
| SG104A  | *1    | 9        |     |     |     |                  |
| SG105A  | *1    | 9        |     |     |     |                  |
| SG106A  | *1    | 9        |     |     |     |                  |
| SG302M  | *2    | 1        |     |     |     | XXXXXX<br>XXXXXX |
| SG201A  |       | 79       |     |     |     |                  |
| SG202A  |       | 5        |     |     |     |                  |
| SG203A  |       | 18       |     |     |     |                  |
| SG204A  |       | 15       |     |     |     |                  |
| SG303M  | *2    | 4        |     |     |     | XXXXXX<br>XXXXXX |
| SG401M  | *2    | 2        |     |     |     | XXXXXX<br>XXXXXX |
| TOTAL   |       | 317      |     |     |     |                  |

\*1 Contains SORT

\*2 Flagging test

Page 19 of 31

Signed .....  
Date .....

13 March 1993

CCVS85 V4.2

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM    | EXE    |
|---------|-------|----------|-----|-----|--------|--------|
| SM101A  |       | 8        |     |     |        |        |
| SM102A  |       | 4        |     |     |        |        |
| SM103A  |       | 6        |     |     |        |        |
| SM104A  |       | 7        |     |     |        |        |
| SM105A  |       | 9        |     |     |        |        |
| SM106A  | *1    | 1        |     |     |        |        |
| SM107A  |       | 200      |     |     |        |        |
| SM301M  | *2    | 1        |     |     | XXXXXX | XXXXXX |
| SM201A  |       | 11       |     |     |        |        |
| SM202A  |       | 7        |     |     |        |        |
| SM203A  |       | 1        |     |     |        |        |
| SM204A  |       | 4        |     |     |        |        |
| SM205A  |       | 9        |     |     |        |        |
| SM206A  |       | 16       | 2   |     |        |        |
| SM207A  |       | 2        |     |     |        |        |
| SM208A  |       | 9        | 1   |     |        |        |
| SM401M  | *2    | 2        |     |     | XXXXXX | XXXXXX |
| TOTAL   |       | 297      |     |     |        |        |

\*1 Report is K6SCA

\*2 Flagging test

Signed .....

Date .....

| PROG ID      | NOTES | NO TESTS   | DEL | EXT | COM | EXE |
|--------------|-------|------------|-----|-----|-----|-----|
| SQ101M       | *1    | 57         |     |     |     |     |
| SQ102A       | *4    | 11         |     |     |     |     |
| SQ103A       | *4    | 30         |     |     |     |     |
| SQ104A       |       | 11         |     |     |     |     |
| SQ105A       |       | 22         |     |     |     |     |
| SQ106A       | *4    | 69         | 6   |     |     |     |
| SQ107A       |       | 6          |     |     |     |     |
| SQ108A       |       | 8          |     |     |     |     |
| SQ109M       | *2 *4 | 6          |     |     |     |     |
| SQ110M       | *3    | 6          |     |     |     |     |
| SQ111A       | *4    | 1          |     |     |     |     |
| SQ112A       | *4    | 7          |     |     |     |     |
| SQ113A       | *4    | 22         |     |     |     |     |
| SQ114A       | *4    | 15         |     |     |     |     |
| SQ115A       |       | 3          |     |     |     |     |
| SQ116A       |       | 10         |     |     |     |     |
| SQ117A       |       | 8          |     |     |     |     |
| SQ121A       |       | 3          |     |     |     |     |
| SQ122A       | *5    | 7          |     |     |     |     |
| SQ123A       |       | 9          |     |     |     |     |
| <b>TOTAL</b> |       | <b>311</b> |     |     |     |     |

\*1 Visual Check

\*2 Close Reel

\*3 Close Unit

\*4 Run on both magnetic tape and mass storage if both media are supported.

\*5 Abnormal Termination may occur after test 7 is reported

Signed .....

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| SQ124A  |       | 19       |     |     |     |     |
| SQ125A  | *1 *3 | 2        |     |     |     |     |
| SQ126A  | *1    | 7        |     |     |     |     |
| SQ127A  |       | 6        |     |     |     |     |
| SQ128A  | *1    | 9        |     |     |     |     |
| SQ129A  | *1    | 1        |     |     |     |     |
| SQ130A  |       | 1        |     |     |     |     |
| SQ131A  |       | 2        |     |     |     |     |
| SQ132A  | *4    | 1        |     |     |     |     |
| SQ133A  |       | 15       |     |     |     |     |
| SQ134A  | *2    | 15       |     |     |     |     |
| TOTAL   |       | 78       |     |     |     |     |

\*1 Run on both magnetic tape and mass storage if both media are supported

\*2 Run only when implementation provides variable-length records for RECORD CONTAINS integer TO integer.

\*3 Abnormal termination may occur after test 2 is reported

\*4 Abnormal termination may occur after test 1 is reported. However, if close of an unopened file is detected at compile time, the test should be considered passed.

Signed .....

Page 22 of 31

Date .....

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| SQ135A       |       | 1         |     |     |     |     |
| SQ136A       |       | 1         |     |     |     |     |
| SQ137A       |       | 1         |     |     |     |     |
| SQ138A       |       | 1         |     |     |     |     |
| SQ139A       | *1    | 1         |     |     |     |     |
| SQ140A       | *1    | 1         |     |     |     |     |
| SQ141A       | *1    | 1         |     |     |     |     |
| SQ142A       | *1    | 1         |     |     |     |     |
| SQ143A       | *1 *2 | 1         |     |     |     |     |
| SQ144A       |       | 1         |     |     |     |     |
| <b>TOTAL</b> |       | <b>10</b> |     |     |     |     |

\*1 Run on both magnetic tape and mass storage if both media are supported

\*2 Abnormal termination may occur prior to reporting test 1. Check execution messages to see that I-O status 42 occurred. However, if close of an unopened file is detected at compile time, the test should be considered passed.

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM   | EXE   |
|---------|-------|----------|-----|-----|-------|-------|
| SQ146A  | *1 *4 | 1        |     |     |       |       |
| SQ147A  | *5    | 1        |     |     |       |       |
| SQ148A  | *6    | 2        |     |     |       |       |
| SQ149A  | *1 *7 | 1        |     |     |       |       |
| SQ150A  | *1 *7 | 1        |     |     |       |       |
| SQ151A  | *1 *5 | 1        |     |     |       |       |
| SQ152A  | *5    | 1        |     |     |       |       |
| SQ153A  | *5    | 1        |     |     |       |       |
| SQ154A  | *1 *8 | 1        |     |     |       |       |
| SQ155A  | *1 *8 | 1        |     |     |       |       |
| SQ156A  | *1 *8 | 1        |     |     |       |       |
| OBSQ1A  | *1    | 6        |     |     |       |       |
| SQ302M  | *3    | 4        |     |     | XXXXX | XXXXX |
| TOTAL   |       | 22       |     |     |       |       |

- \*1 Run on both magnetic tape and mass storage if both media are supported
- \*2 Run only when implementation provides variable-length records for RECORD CONTAINS integer TO integer.
- \*3 Flagging Test
- \*4 Abnormal termination may occur prior to reporting test 1. Check execution messages to see that I-O status 42 occurred. However, if close of an unopened file is detected at compile time, the test should be considered passed.
- \*5 Abnormal termination may occur after test 1 is reported.
- \*6 Abnormal termination may occur after test 2 is reported.
- \*7 Abnormal termination may occur prior to reporting test 1. Check execution messages to see that I-O status 47 occurred.
- \*8 Abnormal termination may occur prior to reporting test 1. Check execution messages to see that I-O status 48 occurred.

Signed .....

Page 24 of 31

Date .....

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| SQ201M       | *2    | 23        |     |     |     |     |
| SQ202A       | *4    | 1         |     |     |     |     |
| SQ203A       | *4    | 4         |     |     |     |     |
| SQ204A       | *4    | 2         |     |     |     |     |
| SQ205A       | *4    | 2         |     |     |     |     |
| SQ206A       | *4    | 4         |     |     |     |     |
| SQ207M       | *1 *2 | 8         |     |     |     |     |
| SQ208M       | *2    | 7         |     |     |     |     |
| SQ209M       | *2 *3 | 3         |     |     |     |     |
| SQ210M       | *2    | 3         |     |     |     |     |
| <b>TOTAL</b> |       | <b>57</b> |     |     |     |     |

\*1 Mnemonic-names (for EOP)

\*2 Visual Check

\*3 Uses files from OBSQ3A

\*4 Run on both magnetic tape and mass storage if both media are supported

Signed .....  
Date .....

| PROG ID | NOTES | NO TESTS | DEL | EXT | COM | EXE |
|---------|-------|----------|-----|-----|-----|-----|
| SQ211A  | *3 *4 | 4        |     |     |     |     |
| SQ212A  |       | 1        |     |     |     |     |
| SQ213A  | *3    | 7        |     |     |     |     |
| SQ214A  |       | 5        |     |     |     |     |
| SQ215A  | *5    | 4        |     |     |     |     |
| SQ216A  | *3    | 7        |     |     |     |     |
| SQ217A  | *3    | 7        |     |     |     |     |
| SQ218A  | *1    | 6        |     |     |     |     |
| SQ219A  | *1 *2 | 6        |     |     |     |     |
| TOTAL   |       | 47       |     |     |     |     |

\*1 Tape files

\*2 This program is not required to be run if RECORD DELIMITER implementor-name is not implemented

\*3 Run on both magnetic tape and mass storage if both media are supported

\*4 Abnormal termination may occur prior to reporting test 4. Check execution messages to see that I-O status 38 occurred.

\*5 Abnormal termination may occur after test 4 is reported.

Signed .....

13 March 1993

CCVS85 V4.2

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| SQ220A       |       | 6         |     |     |     |     |
| SQ221A       |       | 6         |     |     |     |     |
| SQ222A       |       | 6         |     |     |     |     |
| SQ223A       |       | 6         |     |     |     |     |
| SQ224A       |       | 3         |     |     |     |     |
| SQ225A       |       | 3         |     |     |     |     |
| SQ226A       |       | 37        |     |     |     |     |
| SQ227A       |       | 16        |     |     |     |     |
| SQ228A       |       | 1         |     |     |     |     |
| <b>TOTAL</b> |       | <b>84</b> |     |     |     |     |

| PROG ID | NOTES  | NO TESTS | DEL | EXT | COM    | EXE    |
|---------|--------|----------|-----|-----|--------|--------|
| SQ229A  | *6 *7  | 1        |     |     |        |        |
| SQ230A  | *7 *8  | 1        |     |     |        |        |
| OBSQ3A  | *5     |          |     |     |        |        |
| OBSQ4A  | *5     | 4        |     |     |        |        |
| OBSQ5A  | *5     | 5        |     |     |        |        |
| SQ303M  | *1 *2  | 2        |     |     | XXXXXX | XXXXXX |
| SQ401M  | *1,3,4 | 18       |     |     | XXXXXX | XXXXXX |
| TOTAL   |        | 31       |     |     |        |        |

\*1 Flagging tests

\*2 OPEN..REVERSED may not be implemented; run this program only if REVERSED is implemented.

\*3 The values in the BLOCK CONTAINS .. CHARACTERS clause may be changed by the vendor to conform to the physical record size required for the file TFIL.

\*4 If OPEN..REVERSED is not implemented, line 010900 may be commented out and the total number of flags reduced to 17.

\*5 Run on magnetic tape media only.

\*6 Abnormal termination may occur after test 1 is reported.

\*7 Run on both magnetic tape and mass storage if both media are supported

\*8 Abnormal termination may occur prior to reporting test 1. Check execution messages to see that I-O status 47 occurred.

Signed .....

Page 28 of 31

Date .....

| PROG ID      | NOTES | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|-------|-----------|-----|-----|-----|-----|
| ST101A       | *2    | 9         |     |     |     |     |
| ST102A       | *1 *2 | -         |     |     |     |     |
| ST103A       | *2    | 9         |     |     |     |     |
| ST104A       | *2    | 1         |     |     |     |     |
| ST105A       | *2    | 2         |     |     |     |     |
| ST106A       | *2    | 1         |     |     |     |     |
| ST107A       | *2    | 6         |     |     |     |     |
| ST108A       |       | 9         |     |     |     |     |
| ST109A       | *2    | -         |     |     |     |     |
| ST110A       | *1 *2 | -         |     |     |     |     |
| <b>TOTAL</b> |       | <b>37</b> |     |     |     |     |

\*1 No report

\*2 Run on both magnetic tape and mass storage if both media are supported.

| PROG ID      | NOTES  | NO TESTS  | DEL | EXT | COM | EXE |
|--------------|--------|-----------|-----|-----|-----|-----|
| ST111A       | *4     | 7         |     |     |     |     |
| ST112M       | *2 *4  | -         |     |     |     |     |
| ST113M       | *1,2,4 | -         |     |     |     |     |
| ST114M       | *2 *4  | 10        |     |     |     |     |
| ST115A       | *2 *4  | -         |     |     |     |     |
| ST116A       | *1 *4  | -         |     |     |     |     |
| ST117A       | *4     | 1         |     |     |     |     |
| ST118A       |        | 9         |     |     |     |     |
| ST119A       | *4     | 27        |     |     |     |     |
| ST120A       | *1 *4  | -         |     |     |     |     |
| ST121A       | *4     | 9         |     |     |     |     |
| ST122A       | *3,4,5 | -         |     |     |     |     |
| <b>TOTAL</b> |        | <b>63</b> |     |     |     |     |

\*1 No report

\*2 Reel swap

\*3 High Level Only

\*4 Run on both magnetic tape and mass storage if both media are supported.

\*5 This program should only be run when variable length records are provided for RECORD CONTAINS integer TO integer clause

Signed .....

Date .....

| PROG ID | NOTES  | NO TESTS | DEL | EXT | COM | EXE              |
|---------|--------|----------|-----|-----|-----|------------------|
| ST123A  | *2,4,5 | -        |     |     |     |                  |
| ST124A  | *2 *4  | 7        |     |     |     |                  |
| ST125A  | *4     | 1        |     |     |     |                  |
| ST126A  | *4     | 18       |     |     |     |                  |
| ST127A  | *4     | 27       |     |     |     |                  |
| ST131A  | *4     | 15       |     |     |     |                  |
| ST132A  | *1 *4  | 6        |     |     |     |                  |
| ST133A  | *4     | 18       |     |     |     |                  |
| ST134A  | *4     | 4        |     |     |     |                  |
| ST135A  | *2 *4  | 9        |     |     |     |                  |
| ST136A  | *4     | 5        |     |     |     |                  |
| ST137A  | *2     | 6        |     |     |     |                  |
| ST139A  | *2     | 10       |     |     |     |                  |
| ST140A  | *2 *4  | 11       |     |     |     |                  |
| ST144A  | *2 *4  | 11       |     |     |     |                  |
| ST146A  | *2     | 4        |     |     |     |                  |
| ST147A  | *2     | 26       |     |     |     |                  |
| ST301M  | *3     | 6        |     |     |     | XXXXXX<br>XXXXXX |
| TOTAL   |        | 184      |     |     |     |                  |

\*1 Reel swap \*5 No Report

\*2 High Level Only

\*2 High Level On

\*3 Flagging Test

\*4 Run on both magnetic tape and mass storage if both media are supported.

Signed .....

Page 31 of 31

Date .....



**CCVS 85**

**X/OPEN EXTENSION USER GUIDE**

**(Addendum to CCVS85 User Guide)**

**Version 4.2**

**Copyright 1993  
The National Computing Centre Limited  
All rights reserved.**

**Prepared by: The National Computing Centre Limited  
Oxford Road  
Manchester  
M1 7ED  
United Kingdom**

**01 March 1993**



| 1. CONTENTS                                 | Page |
|---------------------------------------------|------|
| 1 Introduction                              | 1    |
| 2 Background                                | 1    |
| 3 Naming conventions for the audit routines | 1    |
| 4 Required resources                        | 2    |
| 5 Using the Tests                           | 2    |
| 6 X-Cards                                   | 2    |
| 7 List of Audit Routines                    | 2    |
| 8 X/Open COBOL Extension Test Module        | X-1  |
| XA001M                                      | X-1  |
| XA002M                                      | X-1  |
| XA003M                                      | X-1  |
| XA004M                                      | X-2  |
| XA005M                                      | X-2  |
| XA006M                                      | X-2  |
| XA007M                                      | X-2  |
| XA008M                                      | X-3  |
| XA009M                                      | X-3  |
| XA010M                                      | X-3  |
| XA011M                                      | X-3  |
| XA012M                                      | X-4  |
| XA013M                                      | X-4  |
| XA014M                                      | X-4  |
| XA015M                                      | X-4  |
| XA016M                                      | X-5  |
| XA017M                                      | X-5  |
| XA018M                                      | X-5  |
| XA019M                                      | X-5  |
| XA020M                                      | X-6  |
| XB001A                                      | X-6  |
| XB002A                                      | X-6  |
| XB003A                                      | X-6  |
| XB004A                                      | X-7  |
| XB005A                                      | X-7  |
| XB006A                                      | X-7  |
| XB007A                                      | X-7  |
| XB008A                                      | X-8  |
| XC008A                                      | X-8  |
| XD001A                                      | X-8  |
| XD002A                                      | X-9  |
| XE001A                                      | X-9  |
| XE002A                                      | X-9  |



|        |      |
|--------|------|
| XE003A | X-10 |
| XE004A | X-10 |
| XE005A | X-10 |
| XF013M | X-11 |
| XF014M | X-11 |
| XF015M | X-11 |
| XF016M | X-12 |
| XF017M | X-12 |
| XF018M | X-12 |
| XF019M | X-12 |
| XF020M | X-13 |
| XF021M | X-13 |
| XF022M | X-13 |
| XF023M | X-13 |
| XF024M | X-14 |
| XF025M | X-14 |
| XF026M | X-14 |
| XF027M | X-14 |
| XF028M | X-15 |
| XH001A | X-15 |
| XH002A | X-15 |
| XH003A | X-16 |
| XH004A | X-16 |
| XH005A | X-16 |
| XH006A | X-17 |
| XH007A | X-17 |
| XH008A | X-17 |
| XH009A | X-18 |
| XH010A | X-18 |
| XH011A | X-18 |
| XH012A | X-19 |
| XH013A | X-19 |
| XH014A | X-19 |
| XH015A | X-20 |
| XH016A | X-20 |
| XH017A | X-20 |
| XH018A | X-21 |
| XH019A | X-21 |
| XH020A | X-21 |
| XH021A | X-21 |
| XH022A | X-22 |
| XH023A | X-22 |
| XH024A | X-22 |
| XH025A | X-22 |
| XH026A | X-23 |
| XH027A | X-23 |
| XH028A | X-23 |
| XH029A | X-24 |
| XH030A | X-24 |
| XH031A | X-24 |
| XH032A | X-25 |
| XH033A | X-25 |
| XH034A | X-25 |



|                                      |      |
|--------------------------------------|------|
| XH035A                               | X-25 |
| XH036A                               | X-26 |
| XH037A                               | X-26 |
| XH038A                               | X-26 |
| XH039A                               | X-26 |
| XH040A                               | X-27 |
| XH041A                               | X-27 |
| XH042A                               | X-27 |
| XH043A                               | X-27 |
| XH044A                               | X-28 |
| XH045A                               | X-28 |
| XH046A                               | X-28 |
| XH047A                               | X-28 |
| XH048A                               | X-29 |
| XH049A                               | X-29 |
| XH050A                               | X-29 |
| XH051A                               | X-29 |
| XH052A                               | X-30 |
| XH053A                               | X-30 |
| XH054A                               | X-30 |
| XH055A                               | X-30 |
| XH056A                               | X-31 |
| XH057A                               | X-31 |
| XH058A                               | X-31 |
| XH059A                               | X-31 |
| XH060A                               | X-32 |
| XH061A                               | X-32 |
| XH062A                               | X-32 |
| XH063A                               | X-32 |
| XH064A                               | X-33 |
| XH065A                               | X-33 |
| XH066A                               | X-33 |
| XH067A                               | X-33 |
| XH068A                               | X-34 |
| XH069A                               | X-34 |
| XH070A                               | X-34 |
| XH071A                               | X-34 |
| XI001A                               | X-35 |
| XM002A                               | X-35 |
| XP001A                               | X-35 |
| XP002A                               | X-35 |
| XP003A                               | X-36 |
| XP004A                               | X-36 |
| XP005A                               | X-36 |
| XP006A                               | X-37 |
| XP007A                               | X-37 |
| <br>                                 |      |
| APPENDIX A - X-Cards                 | A-1  |
| APPENDIX B - CCVS85 Error Procedure  | B-1  |
| APPENDIX C - Temporary Program Fixes | C-1  |



## 1 INTRODUCTION

This User Guide covers the X/Open Extension COBOL tests which are included with the CCVS85 Test Suite.

These tests have been produced for the purpose of testing conformance to the X/Open Common Application Environment (CAE) COBOL language specification. To conform to X/Open COBOL requirements, a vendor must be able to execute all of these tests successfully as well as meet the requirements of ISO/ANSI testing in using the main body of the CCVS.

THESE TESTS ARE NOT REQUIRED FOR FIPS TESTING OR ISO/ANSI TESTING.  
THEY ARE REQUIRED ONLY FOR X/OPEN TESTING.

## 2 BACKGROUND

The tests were produced by

NCC (The National Computing Centre Limited)  
Oxford Road  
Manchester  
M1 7ED  
United Kingdom

## 3 NAMING CONVENTIONS FOR THE AUDIT ROUTINES

An audit routine name consists of six characters, the first character 'X' designates the test as being of X/Open COBOL. The second character represents the area that is to be tested:

|    |   |                                                  |
|----|---|--------------------------------------------------|
| XA | - | Reserved Words                                   |
| XB | - | Binary Usage                                     |
| XC | - | Line Sequential Files, Record Termination        |
| XD | - | USING BY VALUE                                   |
| XE | - | Special Register, RETURN-CODE                    |
| XF | - | Screen Handling                                  |
| XH | - | File Sharing and Record Locking                  |
| XI | - | Concatenation Expressions                        |
| XM | - | ASSIGN TO PRINTER                                |
| XP | - | Command Line Arguments and Environment Variables |

The third character, which is numeric, represents the level of the functional module. As the X/Open tests do not have levels this character is 0 (zero).

The fourth and fifth characters, also numeric, provide each audit routine with a unique name and also indicate the sequence in which programs need to be run, eg XB007A is the seventh in a series of several programs which test Binary Usage in X/Open COBOL. The "A" in the sixth character denotes that the test results are Automatically checked by the program, "M" denotes that some Manual checking is required.

#### **4 REQUIRED RESOURCES**

As well as those resources stated in section 2.3 of the CCVS User Guide, there should also be facilities to run two COBOL programs at the same time (to test X/Open COBOL File Locking).

The X/Open Tests alone take approximately 6 Megabytes of storage.

#### **5 USING THE TESTS**

The main considerations when carrying out X/Open testing regard XH tests (File Sharing and Record Locking).

Firstly, on a non-conforming X/Open implementation, these tests may wait indefinitely for a response from the program that is being run concurrently. Therefore these tests should be run under supervision.

Secondly, users should remember to delete files that are created by XH tests after the test is complete.

See section 3 of the CCVS85 User Guide for details on how to use all other tests that are part of the CCVS85.

#### **6 X-CARDS**

Appendix A gives details of the X-card contents for the X/Open Tests.

#### **7. LIST OF AUDIT ROUTINES**

Section 8 gives a brief description of each of the X/Open COBOL audit routines, the features tested, the number of individual tests and the X-numbers used.

Document Reference: X/Open CAE Specification, COBOL Language, X/CAE/91/200

Test Purpose Reference: NCC Specifications for CCVS X/Open Extensions, XOPCOB/12/3.7.1

The tests described in this section address those aspects of the X/Open COBOL Language Specification which define extensions to ISO 1989:1985 (ANSI X3/23-1985), or provide definitions for elements left as implementor-defined by the standard.

XA001M Program XA001M tests the ability of the implementation to detect use of the reserved word 'AUTO' as a user-defined name (a class-name).

X-numbers: 82, 83

Test Purpose: 1.1

Reference: CAECL/3.1/76/X/0

This program contains one test. The test fails if the program can be executed.

XA002M Program XA002M tests the ability of the implementation to detect use of the reserved word 'AUTOMATIC' as a user-defined name (a condition-name).

X-numbers: 82, 83

Test Purpose: 1.1

Reference: CAECL/3.1/76/X/0

This program contains one test. The test fails if the program can be executed.

XA003M Program XA003M tests the ability of the implementation to detect the use of the reserved word 'BACKGROUND-COLOR' as a user-defined name (a record-name).

X-numbers: 82, 83

Test Purpose: 1.1

Reference: CAECL/3.1/76/X/0

This program contains one test. The test fails if the program can be executed.

- XA004M** Program XA004M tests the ability of the implementation to detect the use of the reserved word 'BELL' as a user-defined name (a file name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA005M** Program XA005M tests the ability of the implementation to detect the use of the reserved word 'BLINK' as a user-defined name (an index-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA006M** Program XA006M tests the ability of the implementation to detect the use of the reserved word 'COMP-3' as a user-defined name (a paragraph-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA007M** Program XA007M tests the ability of the implementation to detect the use of the reserved word 'COMP-5' as a user-defined name (a data-name, in the description of an elementary item and in a reference to that elementary item in an arithmetic statement).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.

- XA008M** Program XA008M tests the ability of the implementation to detect the use of the reserved word 'COMPUTATIONAL-3' as a user-defined name (a section-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA009M** Program XA009M tests the ability of the implementation to detect the use of the reserved word 'COMPUTATIONAL-5' as a user-defined name (the name of a symbolic-character).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA010M** Program XA010M tests the ability of the implementation to detect the use of the reserved word 'CRT' as a user-defined name (a program-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA011M** Program XA011M tests the ability of the implementation to detect the use of the reserved word 'CURSOR' as a user-defined name (a record-name subordinate to an FD).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.

- XA012M Program XA012M tests the ability of the implementation to detect the use of the reserved word 'END-ACCEPT' as a user-defined name (an alphabet-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA013M Program XA013M tests the ability of the implementation to detect the use of the reserved word 'END-DISPLAY' as a user-defined name (a data-name, in the description of an elementary item and in a reference to that elementary item in an arithmetic statement).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA014M Program XA014M tests the ability of the implementation to detect the use of the reserved word 'EOL' as a user-defined name (a paragraph-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA015M Program XA015M tests the ability of the implementation to detect the use of the reserved word 'EOS' as a user-defined name (a condition-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.

- XA016M** Program XA016M tests the ability of the implementation to detect the use of the reserved word 'ERASE' as a user-defined name (a program-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA017M** Program XA017M tests the ability of the implementation to detect the use of the reserved word 'EXCLUSIVE' as a user-defined name (a data-name, in the description of an elementary item and in a reference to that elementary item in an arithmetic statement).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA018M** Program XA018M tests the ability of the implementation to detect the use of the reserved word 'FOREGROUND-COLOR' as a user-defined name (a paragraph-name).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XA019M** Program XA019M tests the ability of the implementation to detect the use of the reserved word 'FULL' as a user-defined name (a data name, in the description of an elementary item and in a reference to that elementary item in an arithmetic statement).
- X-numbers: 82, 83
- Test Purpose: 1.1
- Reference: CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.

- XA020M      Program XA020M tests the ability of the implementation to detect the use of the reserved word 'HIGHLIGHT' as a user-defined name (a file-name).
- X-numbers:      82, 83
- Test Purpose: 1.1
- Reference:      CAECL/3.1/76/X/0
- This program contains one test. The test fails if the program can be executed.
- XB001A      Program XB001A determines that data items with USAGE BINARY are aligned on character boundaries, and may be aligned on both odd and even character boundaries.
- X-numbers:      55, 82, 83
- Test Purpose: 2.1
- Reference:      CAECL/3.2/77/G/3-1
- This program contains 128 tests.
- XB002A      Program XB002A tests that the implementation stores the least significant portion of a data item with USAGE BINARY at the high-address end of its container.
- X-numbers:      55, 82, 83
- Test Purpose: 2.2
- Reference:      CAECL/3.2/77/G/2-2
- This program contains 18 tests.
- XB003A      Program XB003A tests that the implementation stores both signed and unsigned data items with USAGE BINARY and lengths 4, 9, and 18 decimal digits in fields of lengths 2, 4 and 8 bytes respectively.
- X-numbers:      55, 82, 83
- Test Purpose: 2.3
- Reference:      CAECL/3.2/77/G/3
- This program contains 6 tests.

- XB004A** Program XB004A tests that the implementation stores packed decimal data items with two digits per byte, and that in all cases space is allocated for a sign indication.
- X-numbers: 55, 82, 83
- Test Purpose: 2.4
- Reference: CAECL/3.2/78/G/6,7
- This program contains 88 tests.
- XB005A** Program XB005A tests that the implementation stores packed decimal data items with the sign indication in the highest addressed byte, and that the sign representations are different for positive, negative and unsigned values.
- X-numbers: 55, 82, 83
- Test Purpose: 2.5
- Reference: CAECL/3.2/78/G/6,7
- This program contains 8 tests.
- XB006A** Program XB006A tests that the implementation stores packed decimal data items with the sign indication in the highest addressed byte, and that the sign representations are x'C', x'D' and x'F' for positive, negative and unsigned values, respectively.
- X-numbers: 55, 82, 83
- Test Purpose: 2.6
- Reference: CAECL/3.2/78/G/6,7
- This program contains 10 tests.
- XB007A** Program XB007A tests that the implementation stores signed data items with USAGE COMPUTATIONAL-5 and COMP-5, and with lengths 4, 9 and 18 decimal digits in fields of lengths 2, 4 and 8 bytes, respectively.
- X-numbers: 55, 82, 83
- Test Purpose: 2.7
- Reference: CAECL/3.2/78/G/5
- This program contains 24 tests.

- XB008A      Program XB008A tests that the implementation permits data items with USAGE COMPUTATIONAL-5 and COMP-5 to be specified as sending and receiving items in arithmetic statements and that such data items produce or store, as appropriate, the correct values.
- X-numbers:      55, 82, 83
- Test Purpose: 2.8
- Reference:      CAECL/3.2/78/G/5
- This program contains 17 tests.
- XC008A      Program XC008A tests that for files with Line Sequential organisation, the implementation does not count the record terminator in the length of a record and discards the terminator when the record is read. It also tests that Line-Sequential files may be opened in the INPUT, OUTPUT and EXTEND modes, that if a record is read that is shorter than the input record area then it is extended to the length of the buffer with spaces, and that when a record is longer than the input record area then the overflowing portion is treated as the next record.
- Note: The substitution for X-numbers 93 and 94 should be such that they reference the same physical file. X-93 should be the name of the file. X-94 should be the same name presented as a nonnumeric literal.
- X-numbers:      55, 82, 83, 93, 94
- Test Purpose: 3.8, also  
3.3, 3.4, 3.5, 3.6, 3.7, 13.1, 13.3, 13.4
- Reference:      CAECL/3.3/80/G/1,3  
CAECL/3.3/80/5  
CAECL/8.3/164/G/1,3,4
- This program contains 14 tests.
- XD001A      Program XD001A tests that the implementation accepts the syntax for parameter passing by value in a CALL statement with the OVERFLOW phrase, with the parameter specified as any of a 2-byte data item with USAGE COMP-5, a 4-byte data item with USAGE COMP-5, or a 1-byte data item.
- X-numbers:      55, 82, 83
- Test Purpose: 4.1
- Reference:      CAECL/3.4/81/G/3
- This program contains 4 test statements, none of which is executed. The program reports success if it can be run without modification.

- XD002A** Program XD002A tests that the implementation accepts the syntax for parameter passing by value in a CALL statement with the EXCEPTION phrase, with the parameter specified as any of a 2-byte data item with USAGE COMP-5, a 4-byte data item with USAGE COMP-5, or a 1-byte data item.
- X-numbers: 55, 82, 83
- Test Purpose: 4.2
- Reference: CAECL/3.4/81/G/3
- This program contains 3 test statements, none of which is executed. The program reports success if it can be run without modification.
- XE001A** Program XE001A determines that the special register RETURN-CODE is present in a COBOL run unit consisting of a single COBOL source program containing no nested programs, and the special register behaves as a signed integer data item capable of holding values less than 100000.
- X-numbers: 55, 82, 83
- Test Purpose: 5.1
- Reference: CAECL/3.5/83/X/0,8,9/171/X/0
- This program contains 18 tests.
- XE002A** Program XE002A determines that a value placed in the Special Register RETURN-CODE in a called COBOL program which is nested inside the COBOL program which calls it is available to the calling COBOL program on exit from the called program.
- X-numbers: 55, 82, 83
- Test Purpose: 5.2
- Reference: CAECL/3.5/83/X/1,8,9/171/X/1
- This program contains 2 tests.

**XE003A** Program XE003A determines that a value placed in the Special Register RETURN-CODE in a called COBOL program which has been compiled separately from the COBOL program which calls it is available on exit from the called program.

Program XE003A calls separate programs XE004A and XE005A.

X-numbers: 55, 82, 83

Test Purpose: 5.3

Reference: CAECL/3.5/83/X/1.8.9/171/X/2

This program contains 2 tests.

**XE004A** Program XE004A is called by XE003A. It places a specific value in RETURN-CODE and then either returns control to the calling program immediately or after calling and receiving control back from program XE005A, depending on parameters passed by XE003A..

X-numbers: None

Test Purpose: 5.3

Reference: See XE003A

This program is used only in support of program XE003A and contains no tests.

**XE005A** Program XE005A is called by XE003A and XE004A. It places a specific value in special register RETURN-CODE and returns control to the calling program.

X-numbers: None

Test Purpose: 5.3

Reference: See XE003A

This program is used only in support of program XE003A and contains no tests.

**XF013M** Program XF013M tests that when, during the execution of an ACCEPT statement , the number of characters required to fill a field described with the AUTO clause has been entered into that field, and there is at least one more field into which characters should be entered, the cursor moves to the next field. It also tests that the correct value has been stored in the CRT STATUS data item after execution of an ACCEPT statement.

X-numbers: 55, 82, 83

Test Purpose: 6.13, also  
6.5

Reference: CAECL/4.3.3/98/F  
CAECL/4.2.1/90/S/3

This program contains 2 tests.

**XF014M** Program XF014M determines that if the AUTO clause is specified at a group level, it applies to each input and update item subordinate to that group item. It also tests that the correct value has been stored in the CRT STATUS data item after execution of an ACCEPT statement.

X-numbers: 55, 82, 83

Test Purpose: 6.14, also  
6.5

Reference: CAECL/4.3.3/98/G/1  
CAECL/4.2.1/90/S/3

This program contains 1 test.

**XF015M** Program XF015M tests that the values 0 to 7 produce background colours of black, blue, green, cyan, red, magenta, brown and white, respectively. The program is fully applicable only when a colour display is supported by an implementation under test, but may also be used with a multi-level monochrome display.

X-numbers: 55, 82, 83

Test Purpose: 6.15

Reference: CAECL/4.3.4/99/G/2

This program contains 1 test.

|        |                                                                                                                                                                                                                                                                   |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XF016M | Program XF016M tests the capability of the implementation to generate an audio tone when a screen data element for which BELL is specified is processed during the execution of a DISPLAY statement.                                                              |
|        | X-numbers: 55, 82, 83                                                                                                                                                                                                                                             |
|        | Test Purpose: 6.16                                                                                                                                                                                                                                                |
|        | Reference: CAECL/4.3.5/100/G/1                                                                                                                                                                                                                                    |
|        | This program contains 3 tests.                                                                                                                                                                                                                                    |
| XF017M | Program XF017M tests that when the BLANK SCREEN clause is specified in a Screen Description entry, the screen is cleared before presentation of the data item during the execution of a DISPLAY statement, and the cursor is positioned at line 1, column 1.      |
|        | X-numbers: 55, 82, 83                                                                                                                                                                                                                                             |
|        | Test Purpose: 6.17                                                                                                                                                                                                                                                |
|        | Reference: CAECL/4.3.6/101/G/1                                                                                                                                                                                                                                    |
|        | This program contains 1 test.                                                                                                                                                                                                                                     |
| XF018M | Program XF018M determines that when the BLANK LINE clause is specified in a Screen Description Entry, the screen line on which the subject of the entry is displayed is cleared before presentation of the data item during the execution of a DISPLAY statement. |
|        | X-numbers: 55, 82, 83                                                                                                                                                                                                                                             |
|        | Test Purpose: 6.18                                                                                                                                                                                                                                                |
|        | Reference: CAECL/4.3.6/101/G/2                                                                                                                                                                                                                                    |
|        | This program contains 1 test.                                                                                                                                                                                                                                     |
| XF019M | Program XF019M determines that when the BLANK SCREEN clause is specified in a Screen Description entry, it has no effect on the execution of an ACCEPT statement.                                                                                                 |
|        | X-numbers: 55, 82, 83                                                                                                                                                                                                                                             |
|        | Test Purpose: 6.19                                                                                                                                                                                                                                                |
|        | Reference: CAECL/4.3.6/101/G/6                                                                                                                                                                                                                                    |
|        | This program contains 2 tests.                                                                                                                                                                                                                                    |

- XF020M      Program XF020M tests that when the BLINK clause is specified in a Screen Description Entry, the representation of the data item on the display to blink.
- X-numbers:      55, 82, 83
- Test Purpose: 6.20
- Reference:      CAECL/4.3.8/103/F
- This program contains 3 tests.
- XF021M      Program XF021M tests that when PLUS is not specified in the COLUMN clause for a screen data element, the column specification is taken as relative to the column specified in the applicable ACCEPT or DISPLAY statement.
- X-numbers:      55, 82, 83
- Test Purpose: 6.21
- Reference:      CAECL/4.3.9/104/G/2
- This program contains 6 tests.
- XF022M      Program XF022M tests that the values 0 to 7 produce foreground colours of black, blue, green, cyan, red, magenta, brown and white, respectively. The program is fully applicable only when a colour display is supported by an implementation under test, but may also be used with a multi-level monochrome display.
- X-numbers:      55, 82, 83
- Test Purpose: 6.22
- Reference:      CAECL/4.3.11/106/G/2
- This program contains 1 test.
- XF023M      Program XF023M tests that when a FOREGROUND-COLOR clause applies to a screen item, and the description of that screen item also contains a BLANK SCREEN clause, then when the screen item is displayed, the specified colour becomes the default foreground colour, and remains so until changed by display of another screen item that specifies a default foreground colour, in the same or a subsequent DISPLAY statement.
- X-numbers:      55, 82, 83
- Test Purposes: 6.23
- Reference:      CAECL/4.3.11/101/G/5
- This program contains 2 tests.

- XF024M      Program XF024M determines that when PLUS is not specified in the LINE clause for a screen data element, the line specification is taken as relative to the line specified in the applicable ACCEPT or DISPLAY statement.
- X-numbers:      55, 82, 83
- Test Purpose: 6.24
- Reference:      CAECL/4.3.15/110/G/2
- This program contains 4 tests.
- XF025M      Program XF025M determines that when PLUS is specified in the LINE clause for a screen data element, the line specification is taken as relative to the line at which the preceding item ends.
- X-numbers:      55, 82, 83
- Test Purpose: 6.25
- Reference:      CAECL/4.3.15/110/G/4
- This program contains 4 tests.
- XF026M      Program XF026M determines that when the REVERSE-VIDEO clause is specified in the description of a screen data item, the default or specified foreground and background colours for that item are interchanged when it is presented on the display.
- X-numbers:      55, 82, 83
- Test Purpose: 6.26
- Reference:      CAECL/4.3.19/115/F
- This program contains 3 tests.
- XF027M      Program XF027M tests that when the SECURE clause applies to a screen data item, characters entered to that item during the execution of an ACCEPT statement do not appear on the screen.
- X-numbers:      55, 82, 83
- Test Purpose: 6.27
- Reference:      CAECL/4.3.20/116/G/2
- This program contains 2 tests.

**XF028M** Program XF028M determines that when the UNDERLINE clause is specified in the description of a screen data item, each character of the field is underlined when it is presented on the display.

X-numbers: 55, 82, 83

Test Purpose: 6.28

Reference: CAECL/4.3.22/118/F

This program contains 2 tests.

**XH001A** Program XH001A is executed concurrently with XH002A to test that when a program has a sequential file open in exclusive lock mode, an attempt by another program to open that file in shared mode receives a 'File Locked' error and is denied access. Program XH001A establishes and releases locks on a file. Program XH002A contains the tests.

X-numbers: 55, 82, 83, 95, 96, 97, 98

Test Purpose: 8.1 also  
8.5, 8.6, 8.9, 8.10, 8.14

Reference: CAECL/5.2/126/X, 5.7.3/136/G/2  
CAECL/5.5/X  
CAECL/5.6.1/133/G/2,3  
CAECL/5.7.1/134/G/1  
CAECL/5.7.3/136/G/1

This program contains no tests.

**XH002A** See XH001A

X-numbers: 55, 82, 83, 95, 96, 97, 98, 99

Test Purpose: 8.1, also  
8.5, 8.6, 8.9, 8.10

Reference: CAECL/5.2/126/X, 5.7.3/136/G/2  
CAECL/5.5/X  
CAECL/5.6.1/133/G/2,3  
CAECL/5.7.1/134/G/1

This program contains 4 tests.

XH003A      Program XH003A is executed concurrently with XH004A to test that when a program has a relative file open in exclusive lock mode, an attempt by another program to open that file in shared mode receives a "File Locked" error and is denied access. Program XH003A establishes and releases locks on a file. Program XH004A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose: 8.1, also  
                  8.5, 8.7, 8.10, 8.14

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1

This program contains no tests.

XH004A      See XH003A

X-numbers:      55, 82, 83, 95, 96, 97, 98, 99

Test Purpose: 8.1, also  
                  8.5, 8.7, 8.10

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1

This program contains 4 tests.

XH005A      Program XH005A is executed concurrently with XH006A to test that when a program has an indexed file open in exclusive lock mode, an attempt by another program to open that file in shared mode receives a "File Locked" error and is denied access. Program XH005A establishes and releases locks on a file. Program XH006A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose: 8.1, also  
                  8.5, 8.8, 8.10, 8.14

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1

This program contains no tests.

XH006A      See XH005A

X-numbers:      55, 82, 83, 95, 96, 97, 98, 99

Test Purpose:  8.1, also  
                  8.5, 8.8, 8.10

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1

This program contains 4 tests.

XH007A      Program XH007A is executed concurrently with XH008A to test that when a program has a sequential file open in exclusive lock mode, an attempt by another program to open that file in exclusive mode receives a 'File Locked' error and is denied access. Program XH007A establishes and releases locks on a file. Program XH008A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose:  8.2, also  
                  8.5, 8.6, 8.9, 8.10, 8.14

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2,3  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1

This program contains no tests.

XH008A      See XH007A

X-numbers:      55, 82, 83, 95, 96, 97, 98, 99

Test Purpose:  8.2, also  
                  8.5, 8.6, 8.9, 8.10

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2,3  
                  CAECL/5.7.1/134/G/1

This program contains 8 tests.

XH009A      Program XH009A is executed concurrently with XH010A to test that when a program has a relative file open in exclusive lock mode, an attempt by another program to open that file in exclusive mode receives a "File Locked" error and is denied access. Program XH009A establishes and releases locks on a file. Program XH010A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose:  8.2, also  
                  8.5, 8.7, 8.10, 8.14

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1

This program contains no tests.

XH010A      See XH009A

X-numbers:      55, 82, 83, 95, 96, 97, 98, 99

Test Purpose:  8.2, also  
                  8.5, 8.7, 8.10

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1

This program contains 8 tests.

XH011A      Program XH011A is executed concurrently with XH012A to test that when a program has an indexed file open in exclusive mode, an attempt by another program to open that file in shared mode receives a "File Locked" error and is denied access. Program XH011A establishes and releases locks on a file. Program XH012A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose:  8.2, also  
                  8.5, 8.8, 8.10, 8.14

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/2  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1

This program contains no tests.

|        |                                                                                                                                                                                                                                                                                                                                                    |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XH012A | See XH011A                                                                                                                                                                                                                                                                                                                                         |
|        | X-numbers: 55, 82, 83, 9, 96, 97, 98, 99                                                                                                                                                                                                                                                                                                           |
|        | Test Purpose: 8.2, also<br>8.5, 8.8, 8.10                                                                                                                                                                                                                                                                                                          |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/2<br>CAECL/5.5/X<br>CAECL/5.6.1/133/G/2<br>CAECL/5.7.1/134/G/1                                                                                                                                                                                                                                             |
|        | This program contains 8 tests.                                                                                                                                                                                                                                                                                                                     |
| XH013A | Program XH013A is executed concurrently with XH014A to test that when a program has a sequential file open in shared lock mode, an attempt by another program to open that file in exclusive mode receives a "File Locked" error and is denied access. Program XH013A establishes and releases locks on a file. Program XH014A contains the tests. |
|        | X-numbers: 55, 82, 83, 95, 96, 97, 98                                                                                                                                                                                                                                                                                                              |
|        | Test Purpose: 8.3, also<br>8.5, 8.6, 8.9, 8.11, 8.15, 8.16                                                                                                                                                                                                                                                                                         |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/3<br>CAECL/5.5/X<br>CAECL/5.6.1/133/G/2,3<br>CAECL/5.7.1/134/G/1<br>CAECL/5.7.3/136/G/3                                                                                                                                                                                                                    |
|        | This program contains no tests.                                                                                                                                                                                                                                                                                                                    |
| XH014A | See XH013A                                                                                                                                                                                                                                                                                                                                         |
|        | X-numbers: 55, 82, 83, 95, 96, 97, 98, 99                                                                                                                                                                                                                                                                                                          |
|        | Test Purpose: 8.3, also<br>8.5, 8.6, 8.9, 8.11, 8.14, 8.15, 8.16                                                                                                                                                                                                                                                                                   |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/3<br>CAECL/5.5/X<br>CAECL/5.6.1/133/G/2,3<br>CAECL/5.7.1/134/G/1<br>CAECL/5.7.3/136/G/1,3                                                                                                                                                                                                                  |
|        | This program contains 16 tests.                                                                                                                                                                                                                                                                                                                    |

XH015A      Program XH015A is executed concurrently with XH016A to test that when a program has a relative file open in shared lock mode, an attempt by another program to open that file in exclusive mode receives a "File Locked" error and is denied access. Program XH015A establishes and releases locks on a file. Program XH016A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose:  8.3, also  
                  8.5, 8.7, 8.11, 8.15, 8.16

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1, 3

This program contains no tests.

XH016A      See XH015A

X-numbers:      55, 82, 83, 95, 96, 97, 98, 99

Test Purpose:  8.3, also  
                  8.5, 8.7, 8.11, 8.14, 8.15, 8.16

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1,3

This program contains 16 tests.

XH017A      Program XH017A is executed concurrently with XH018A to test that when a program has an indexed file open in shared lock mode, an attempt by another program to open that file in exclusive mode receives a "File Locked" error and is denied access. Program XH017A establishes and releases locks on a file. Program XH018A contains the tests.

X-numbers:      55, 82, 83, 95, 96, 97, 98

Test Purpose:  8.3, also  
                  8.5, 8.8, 8.11, 8.15, 8.16

Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/3

This program contains no tests.

- XH018A      See XH017A
- X-numbers:      55, 82, 83, 95, 96, 97, 98, 99
- Test Purpose: 8.3, also  
                  8.5, 8.8, 8.11, 8.14, 8.15, 8.16
- Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3  
                  CAECL/5.5/X  
                  CAECL/5.6.1/133/G/2  
                  CAECL/5.7.1/134/G/1  
                  CAECL/5.7.3/136/G/1, 3
- This program contains 16 tests.
- XH019A      Program XH019A is executed concurrently with XH020A to test that when a program has a sequential file open in shared lock mode, another program may also open that file in shared mode. Program XH019A opens the subject file first. Program XH020A then tests that it can also open the file.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.4
- Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3
- This program contains no tests.
- XH020A      See XH019A
- X-numbers:      55, 82, 83, 95, 97, 99
- Test Purpose: 8.4
- Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3
- This program contains 1 test.
- XH021A      Program XH021A is executed concurrently with XH022A to test that when a program has a relative file open in shared lock mode, another program may also open that file in shared mode. Program XH021A opens the subject file first. Program XH022A then tests that it can also open the file.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.4
- Reference:      CAECL/5.2/126/X, 5.7.3/136/G/3
- This program contains no tests.

|        |                                                                                                                                                                                                                                                                                                                                                                               |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XH022A | See XH021A                                                                                                                                                                                                                                                                                                                                                                    |
|        | X-numbers: 55, 82, 83, 95, 97, 99                                                                                                                                                                                                                                                                                                                                             |
|        | Test Purpose: 8.4                                                                                                                                                                                                                                                                                                                                                             |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/3                                                                                                                                                                                                                                                                                                                                     |
|        | This program contains 1 test.                                                                                                                                                                                                                                                                                                                                                 |
| XH023A | Program XH023A is executed concurrently with XH024A to test that when a program has an indexed file open in shared lock mode, another program may also open that file in shared mode. Program XH023A opens the subject file first. Program XH024A then tests that it can also open the file.                                                                                  |
|        | X-numbers: 82, 83, 95, 97                                                                                                                                                                                                                                                                                                                                                     |
|        | Test Purpose: 8.4                                                                                                                                                                                                                                                                                                                                                             |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/3                                                                                                                                                                                                                                                                                                                                     |
|        | This program contains no tests.                                                                                                                                                                                                                                                                                                                                               |
| XH024A | See XH023A                                                                                                                                                                                                                                                                                                                                                                    |
|        | X-numbers: 55, 82, 83, 95, 97, 99                                                                                                                                                                                                                                                                                                                                             |
|        | Test Purpose: 8.4                                                                                                                                                                                                                                                                                                                                                             |
|        | Reference: CAECL/5.2/126/X, 5.7.3/136/G/3                                                                                                                                                                                                                                                                                                                                     |
|        | This program contains 1 test.                                                                                                                                                                                                                                                                                                                                                 |
| XH025A | Program XH025A is executed concurrently with program XH026A to test that a file status code of "9x" is returned for a sequential file when the execution of a READ or REWRITE statement fails because the record is locked via another file connector, where "x" is a character defined by the implementor to identify the particular situation. Both programs contain tests. |
|        | X-numbers: 82, 83, 95, 97, 100, 101, 102                                                                                                                                                                                                                                                                                                                                      |
|        | Test Purpose: 8.5, also<br>8.17, 8.22                                                                                                                                                                                                                                                                                                                                         |
|        | Reference: CAECL/5.5/X<br>CAECL/5.7.4/137/G/2<br>CAECL/5.7.5/139/G/1                                                                                                                                                                                                                                                                                                          |
|        | This program contains 3 tests.                                                                                                                                                                                                                                                                                                                                                |

|        |                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XH026A | See XH026A                                                                                                                                                                                                                                                                                                                                                                          |
|        | X-numbers: 55, 82, 83, 95, 101, 102                                                                                                                                                                                                                                                                                                                                                 |
|        | Test Purpose: 8.5, also<br>8.17, 8.22                                                                                                                                                                                                                                                                                                                                               |
|        | Reference: CAECL/5.5/X<br>CAECL/5.7.4/137/G/2<br>CAECL/5.7.5/139/G/1                                                                                                                                                                                                                                                                                                                |
|        | This program contains 10 tests.                                                                                                                                                                                                                                                                                                                                                     |
| XH027A | Program XH027A is executed concurrently with program XH028A to test that a file status code of "9x" is returned for a relative file when the execution of a DELETE, READ or REWRITE statement fails because the record is locked via another file connector, where "x" is a character defined by the implementor to identify the particular situation. Both programs contain tests. |
|        | X-numbers: 82, 83, 95, 97, 100, 101, 102                                                                                                                                                                                                                                                                                                                                            |
|        | Test Purpose: 8.5, also<br>8.12, 8.17, 8.22                                                                                                                                                                                                                                                                                                                                         |
|        | Reference: CAECL/5.5/X<br>CAECL/5.7.2/135/G/1<br>CAECL/5.7.4/137/G/2<br>CAECL/5.7.5/139/G/1                                                                                                                                                                                                                                                                                         |
|        | This program contains 1 test.                                                                                                                                                                                                                                                                                                                                                       |
| XH028A | See XH027A                                                                                                                                                                                                                                                                                                                                                                          |
|        | X-numbers: 55, 82, 83, 95, 97, 101, 102                                                                                                                                                                                                                                                                                                                                             |
|        | Test Purpose: 8.5, also<br>8.12, 8.17, 8.22                                                                                                                                                                                                                                                                                                                                         |
|        | Reference: CAECL/5.5/X<br>CAECL/5.7.2/135/G/1<br>CAECL/5.7.4/137/G/2<br>CAECL/5.7.5/139/G/1                                                                                                                                                                                                                                                                                         |
|        | This program contains 10 tests.                                                                                                                                                                                                                                                                                                                                                     |

XH029A      Program XH029A is executed concurrently with program XH030A to test that a file status code of "9x" is returned for an indexed file when the execution of a DELETE, READ or REWRITE statement fails because the record is locked via another file connector, where "x" is a character defined by the implementor to identify the particular situation. Both programs contain tests.

X-numbers:      82, 83, 95, 97, 100, 101, 102

Test Purpose:  8.5, also  
                  8.12, 8.17, 8.22

Reference:      CAECL/5.5/X  
                  CAECL/5.7.2/135/G/1  
                  CAECL/5.7.4/137/G/2  
                  CAECL/5.7.5/139/G/1

This program contains 1 test.

XH030A      See XHO29A

X-numbers:      55, 82, 83, 95, 97, 101, 102

Test Purpose:  8.5, also  
                  8.12, 8.17, 8.22

Reference:      CAECL/5.5/X  
                  CAECL/5.7.2/135/G/1  
                  CAECL/5.7.4/137/G/2  
                  CAECL/5.7.5/139/G/1

This program contains 10 tests.

XH031A      Program XH031A is executed concurrently with XH032A to test that execution of a DELETE statement referencing a record in a relative file may be successful when the file connector referenced in the DELETE statement holds a lock on the record to be deleted and no other file connector holds a lock on that record.

X-numbers:      82, 83, 95, 97, 101, 102

Test Purpose:  8.13

Reference:      CAECL/5.7.2/135/G/1

This program contains no tests.

- XH032A      See XH031A
- X-numbers:      55, 82, 83, 95, 97, 101, 102
- Test Purpose: 8.13
- Reference:      CAECL/5.7.2/135/G/1
- This program contains 8 tests.
- XH033A      Program XH033A is executed concurrently with XH034A to test that execution of a DELETE statement referencing a record in an indexed file may be successful when the file connector referenced in the DELETE statement holds a lock on the record to be deleted and no other file connector holds a lock on that record.
- X-numbers:      82, 83, 95, 101, 102
- Test Purpose: 8.13
- Reference:      CAECL/5.7.2/135/G/1
- This program contains no tests.
- XH034A      See XH033A
- X-numbers:      55, 82, 83, 95, 97, 101, 102
- Test Purpose: 8.13
- Reference:      CAECL/5.7.2/135/G/1
- This program contains 8 tests.
- XH035A      Program XH035A is executed concurrently with XH036A to test that, for sequential files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then execution of a READ statement with the WITH NO LOCK phrase does not acquire a lock on the record that is read.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains no tests.

- XH036A      See XH035A
- X-numbers:      55, 82, 83, 95, 97, 101
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains 3 tests.
- XH037A      Program XH037A is executed concurrently with XH038A to test that, for relative files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then execution of a READ statement with the WITH NO LOCK phrase does not acquire a lock on the record that is read.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains no tests.
- XH038A      See XH037A
- X-numbers:      55, 82, 83, 95, 97, 101
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains 3 tests.
- XH039A      Program XH039A is executed concurrently with XH040A to test that, for indexed files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then execution of a READ statement with the WITH NO LOCK phrase does not acquire a lock on the record that is read.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains no tests.

- XH040A      See XH039A
- X-numbers:      55, 82, 83, 95, 97, 101
- Test Purpose: 8.18
- Reference:      CAECL/5.7.4/137/G/4
- This program contains 3 tests.
- XH041A      Program XH041A is executed concurrently with XH042A to test that, for sequential files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then successful execution of a READ statement without the WITH NO LOCK phrase acquires a lock on the record that is read, and the lock is released by the next execution of an I-O statement, other than a START statement, referencing that file connector.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.19
- Reference:      CAECL/5.7.4/137/G/4
- This program contains no tests.
- XH042A      See XH041A
- X-numbers:      55, 82, 83, 95, 97, 103
- Test Purpose: 8.19
- Reference:      CAECL/5.7.4/137/G/4
- This program contains 9 tests.
- XH043A      Program XH043A is executed concurrently with XH044A to test that, for relative files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then successful execution of a READ statement without the WITH NO LOCK phrase acquires a lock on the record that is read, and the lock is released by the next execution of an I-O statement, other than a START statement, referencing that file connector.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.19
- Reference:      CAECL/5.7.4/137/G/4
- This program contains no tests.

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XH044A | See XH043A                                                                                                                                                                                                                                                                                                                                                                                                                               |
|        | X-numbers: 55, 82, 83, 95, 97, 103                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Test Purpose: 8.19                                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Reference: CAECL/5.7.4/137/G/4                                                                                                                                                                                                                                                                                                                                                                                                           |
|        | This program contains 9 tests.                                                                                                                                                                                                                                                                                                                                                                                                           |
| XH045A | Program XH045A is executed concurrently with XH046A to test that, for indexed files, when the lock mode for a file connector is AUTOMATIC and the file is open in I-O mode, then successful execution of a READ statement without the WITH NO LOCK phrase acquires a lock on the record that is read, and the lock is released by the next execution of an I-O statement, other than a START statement, referencing that file connector. |
|        | X-numbers: 82, 83, 95, 97                                                                                                                                                                                                                                                                                                                                                                                                                |
|        | Test Purpose: 8.19                                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Reference: CAECL/5.7.4/137/G/4                                                                                                                                                                                                                                                                                                                                                                                                           |
|        | This program contains no tests.                                                                                                                                                                                                                                                                                                                                                                                                          |
| XH046A | See XH045A                                                                                                                                                                                                                                                                                                                                                                                                                               |
|        | X-numbers: 55, 82, 83, 95, 97, 103                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Test Purpose: 8.19                                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Reference: CAECL/5.7.4/137/G/4                                                                                                                                                                                                                                                                                                                                                                                                           |
|        | This program contains 9 tests.                                                                                                                                                                                                                                                                                                                                                                                                           |
| XH047A | Program XH047A is executed concurrently with XH048A to test that, for relative files, when the lock mode for a file connector is MANUAL and the file is open in I-O mode, then successful execution of a READ statement without the WITH NO LOCK phrase does not acquire a lock on the record that is read.                                                                                                                              |
|        | X-numbers: 82, 83, 95, 97                                                                                                                                                                                                                                                                                                                                                                                                                |
|        | Test Purpose: 8.20                                                                                                                                                                                                                                                                                                                                                                                                                       |
|        | Reference: CAECL/5.7.4/137/G/5                                                                                                                                                                                                                                                                                                                                                                                                           |
|        | This program contains no tests.                                                                                                                                                                                                                                                                                                                                                                                                          |

XH048A      See XH047A

X-numbers:      55, 82, 83, 95, 97, 99

Test Purpose: 8.20

Reference:      CAECL/5.7.4/137/G/5

This program contains 3 tests.

XH049A      Program XH050A is executed concurrently with XH049A to test that, for indexed files, when the lock mode for a file connector is MANUAL and the file is open in I-O mode, then successful execution of a READ statement without the WITH NO LOCK phrase does not acquire a lock on the record that is read.

X-numbers:      82, 83, 95, 97

Test Purpose: 8.20

Reference:      CAECL/5.7.4/137/G/5

This program contains no tests.

XH050A      See XH049A

X-numbers:      55, 82, 83, 95, 97, 99

Test Purpose: 8.20

Reference:      CAECL/5.7.4/137/G/5

This program contains 3 tests.

XH051A      Program XH051A is executed concurrently with XH052A to test that, for relative files, when the lock mode for a file connector is MANUAL and the file is open in I-O mode, then successful execution of a READ statement with the WITH LOCK phrase acquires a lock on the record that is read.

X-numbers:      82, 83, 95, 97

Test Purpose: 8.21

Reference:      CAECL/5.7.4/137/G/5

This program contains no tests.

- XH052A      See XH051A
- X-numbers:      55, 82, 83, 95, 97, 103
- Test Purpose: 8.21
- Reference:      CAECL/5.7.4/137/G/5
- This program contains 2 tests.
- XH053A      Program XH053A is executed concurrently with XH054A to test that, for indexed files, when the lock mode for a file connector is MANUAL and the file is open in I-O mode, then successful execution of a READ statement with the WITH LOCK phrase acquires a lock on the record that is read.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.21
- Reference:      CAECL/5.7.4/137/G/5
- This program contains no tests.
- XH054A      See XH053A
- X-numbers:      55, 82, 83, 95, 97, 103
- Test Purpose: 8.21
- Reference:      CAECL/5.7.4/137/G/5
- This program contains 2 tests.
- XH055A      Program XH055A tests that execution of an UNLOCK statement for a relative file can be successful when the referenced file connector does not hold any locks.
- X-numbers:      55, 82, 83, 95
- Test Purpose: 8.24
- Reference:      CAECL/5.7.7/141/G/2
- This program contains 5 tests.

- XH056A      Program XH056A tests that execution of an UNLOCK statement for an indexed file can be successful when the referenced file connector does not hold any locks.
- X-numbers:      55, 82, 83, 95
- Test Purpose: 8.24
- Reference:      CAECL/5.7.7/141/G/2
- This program contains 5 tests.
- XH057A      Program XH057A is executed concurrently with XH058A to test that, for a relative file, successful execution of a START statement does not acquire a record lock on the referenced record.
- X-numbers:      82, 83, 95, 97, 101, 102
- Test Purpose: 8.23
- Reference:      CAECL/5.7.6/140/G/1
- This program contains no tests.
- XH058A      See XH057A
- X-numbers:      55, 82, 83, 95, 97, 101, 102
- Test Purpose: 8.23
- Reference:      CAECL/5.7.6/140/G/5
- This program contains 4 tests.
- XH059A      Program XH059A is executed concurrently with XH060A to test that, for an indexed file, successful execution of a START statement does not acquire a record lock on the referenced record.
- X-numbers:      82, 83, 95, 97, 101, 102
- Test Purpose: 8.23
- Reference:      CAECL/5.7.6/140/G/1
- This program contains no tests.

|        |                                                                                                                                                                                              |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XH060A | See XH059A                                                                                                                                                                                   |
|        | X-numbers: 55, 82, 83, 95, 97, 101, 102                                                                                                                                                      |
|        | Test Purpose: 8.23                                                                                                                                                                           |
|        | Reference: CAECL/5.7.6/140/G/5                                                                                                                                                               |
|        | This program contains 4 tests.                                                                                                                                                               |
| XH061A | Program XH061A determines that execution of an UNLOCK statement for a sequential file can be successful when the referenced file connector does not hold any locks.                          |
|        | X-numbers: 55, 82, 83, 95                                                                                                                                                                    |
|        | Test Purpose: 8.24                                                                                                                                                                           |
|        | Reference: CAECL/5.7.7/141/6/2                                                                                                                                                               |
|        | This program contains 5 tests.                                                                                                                                                               |
| XH062A | Program XH062A is executed concurrently with XH063A to test that successful execution of an UNLOCK statement for a sequential file releases any locks held by the referenced file connector. |
|        | X-numbers: 82, 83, 95, 97, 103                                                                                                                                                               |
|        | Test Purpose: 8.25                                                                                                                                                                           |
|        | Reference: CAECL/5.7.7/141/G/2                                                                                                                                                               |
|        | This program contains no tests.                                                                                                                                                              |
| XH063A | See XH062A                                                                                                                                                                                   |
|        | X-numbers: 55, 82, 83, 95, 97, 103                                                                                                                                                           |
|        | Test Purpose: 8.25                                                                                                                                                                           |
|        | Reference: CAECL/5.7.7/141/G/2                                                                                                                                                               |
|        | This program contains 4 tests.                                                                                                                                                               |

XH064A Program XH064A is executed concurrently with XH065A to test that successful execution of an UNLOCK statement for a relative file releases any locks held by the referenced file connector.

X-numbers: 82, 83, 95, 97, 103

Test Purpose: 8.25

Reference: CAECL/5.7.7/141/G/2

This program contains no tests.

XH065A See XH064A

X-numbers: 55, 82, 83, 95, 97, 103

Test Purpose: 8.25

Reference: CAECL/5.7.7/141/G/2

This program contains 7 tests.

XH066A Program XH066A is executed concurrently with XH067A to test that successful execution of an UNLOCK statement for an indexed file releases any locks held by the referenced file connector.

X-numbers: 82, 83, 95, 97, 103

Test Purpose: 8.25

Reference: CAECL/5.7.7/141/G/2

This program contains no tests.

XH067A See XH066A

X-numbers: 55, 82, 83, 95, 97, 103

Test Purpose: 8.25

Reference: CAECL/5.7.7/141/G/2

This program contains 7 tests.

- XH068A      Program XH068A is executed concurrently with XH069A to test that when, for a relative file, the locking mode is AUTOMATIC, successful execution of a WRITE statement releases any existing record lock held by the referenced file connector.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.26
- Reference:      CAECL/5.7.8/142/G/4
- This program contains no tests.
- XH069A      See XH068A
- X-numbers:      55, 82, 83, 95, 97, 103
- Test Purpose: 8.26
- Reference:      CAECL/5.7.8/142/G/4
- This program contains 5 tests.
- XH070A      Program XH070A is executed concurrently with XH071A to test that when, for an indexed file, the locking mode is AUTOMATIC, successful execution of a WRITE statement releases any existing record lock held by the referenced file connector.
- X-numbers:      82, 83, 95, 97
- Test Purpose: 8.26
- Reference:      CAECL/5.7.8/142/G/4
- This program contains no tests.
- XH071A      See XH070A
- X-numbers:      55, 82, 83, 95, 97, 103
- Test Purpose: 8.26
- Reference:      CAECL/5.7.8/142/G/4
- This program contains 4 tests.

- XI001A      Program XI001A tests that a concatenation of non-numeric literals is treated as a single non-numeric literal consisting of the concatenation of the separate literals.
- X-numbers:        55, 82, 83
- Test Purpose: 9.1
- Reference:        CAECL/3.7/85/G/1
- This program contains 8 tests.
- XM002A      Program XM002A tests that when PRINTER is specified as implementor-name in the ASSIGN clause of a file control entry, then the associated physical file is a print file.
- X-numbers:        82, 83
- Test Purpose: 13.2
- Reference:        CAECL/8.3/164/G/2
- This program contains 1 test.
- XP001A      Program XP001A tests that the number of command line arguments is made available to a run unit through the implementor-name ARGUMENT-NUMBER and the ACCEPT statement.
- This particular program should be run with no arguments on the command line.
- X-numbers:        55, 82, 83
- Test Purpose: 16.1
- Reference:        CAECL/8.6/166/G/1
- This program contains 5 tests.
- XP002A      Program XP002A tests that the number of command line arguments is made available to a run unit through the implementor-name ARGUMENT-NUMBER and the ACCEPT statement.
- This particular program should be run with one command line argument.
- X-numbers:        55, 82, 83
- Test Purpose: 16.1, also  
                        16.2, 16.3
- Reference:        CAECL/8.6/166/G/1
- This program contains 9 tests.

- XP003A** Program XP003A tests that the number of command line arguments is made available to a run unit through the implementor-name ARGUMENT-NUMBER and the ACCEPT statement.
- This particular program should be run with 13 arguments on the command line.
- X-numbers: 55, 82, 83
- Test Purpose: 16.1, also  
16.2, 16.3
- Reference: CAECL/8.6/166/G/1
- This program contains 12 tests.
- XP004A** Program XP004A tests that the number of command line arguments is made available to a run unit through the implementor-name ARGUMENT-NUMBER and the ACCEPT statement.
- This particular program should be run with 99 arguments on the command line.
- X-numbers: 55, 82, 83
- Test Purpose: 16.1, also  
16.2, 16.3
- Reference: CAECL/8.6/166/G/1
- This program contains 5 tests.
- XP005A** Program XP005A tests that the value of an environment variable can be set through successive execution of a DISPLAY statement that references the system-name ENVIRONMENT-NAME and the required name, and of a DISPLAY statement that references the system-name ENVIRONMENT-VALUE and the required value.
- Program XP005A is called by program XP007A, which then calls program XP006A to check the values set in the environment variables by program XP005A.
- X-numbers: 55, 82, 83
- Test Purpose: 16.4
- Reference: CAECL/8.6/166/G/5,6
- This program contains 6 tests.

- XP006A** Program XP006A tests that the value of an environment variable can be retrieved through successive execution of a DISPLAY statement that references the system-name ENVIRONMENT-NAME and the required name, and of an ACCEPT statement that references the system-name ENVIRONMENT-VALUE.
- Program XP006A is called by program XP007A and checks that values set in environment variables by program XP005A can be retrieved.
- X-numbers: 82, 83, 100
- Test Purpose: 16.5, also  
16.4
- Reference: CAECL/8.6/166/G/5,6
- This program contains 2 tests.
- XP007A** Program XP007A calls programs XP005A and XP006A in succession in order to check that values set in environment variables by one program can be retrieved by another program.
- X-numbers: 82, 83
- Test Purpose: 16.5, also  
16.4
- Reference: CAECL/8.6/166/G/5,6
- This program contains no tests.
- XP001A.SH** This is a data file which contains the argument values expected by programs XP001A, XP002A, XP003A and XP004A.
- The file contains four lines. Each line begins with the name of a program, which is followed by the arguments required by that program.



## APPENDIX A

### X-Cards Used for X/Open Tests

| X-card | Function                                                             | Explanation                                                                   |
|--------|----------------------------------------------------------------------|-------------------------------------------------------------------------------|
| X-55   | System printer                                                       | Implementor-name for output listings in ASSIGN TO clause of SELECT statement. |
| X-82   | Source computer name                                                 | Implementor-name defined in SOURCE-COMPUTER paragraph.                        |
| X-83   | Object computer name                                                 | Implementor-name defined in OBJECT-COMPUTER paragraph.                        |
| X-93   | Sequential Mass Storage name                                         |                                                                               |
| X-94   | Nonnumeric literal containing Sequential Mass Storage name           | Note: X-93 and X-94 must reference the same physical file name                |
| X-95   | Filename-1 for File Sharing and Record Locking Tests                 |                                                                               |
| X-96   | Filename-2 for File Sharing and Record Locking Tests                 |                                                                               |
| X-97   | Synchronisation Filename-1 for File Sharing and Record Locking Tests |                                                                               |
| X-98   | Synchronisation Filename-2 for File Sharing and Record Locking Tests |                                                                               |
| X-99   | File Locked Status Code for File Sharing and Record Locking Tests    | 2 Character Code (e.g. "9A")                                                  |
| X-100  | Secondary Print file filename                                        |                                                                               |

|       |                                                                           |                                                                     |
|-------|---------------------------------------------------------------------------|---------------------------------------------------------------------|
| X-101 | File Locked Status Code<br>for File Sharing<br>and Record Locking Tests   | 1 Character Code. 2nd Character of<br>the value of X-99 (e.g. "A")  |
| X-102 | Record Locked Status Code<br>for File Sharing<br>and Record Locking Tests | 1 Character Code. 2nd Character of<br>the value of X-103 (e.g. "D") |
| X-103 | Record Locked Status Code<br>for File Sharing<br>and Record Locking Tests | 2 Character Code (e.g. "9D")                                        |

## **APPENDIX B**

### **ERROR PROCEDURE**

If you should discover an error in the CCVS85 X/Open Extension Tests we would be grateful if you would complete the form overleaf and return it to the contact listed below:

Dave Bamber  
NCC Open Systems  
The National Computing Centre Ltd  
Oxford Road  
Manchester  
M1 7ED  
United Kingdom

E-Mail:        [cobol@ncc.co.uk](mailto:cobol@ncc.co.uk)



SYSTEM NAME: CCVS85 V4.2

PROGRAM NAME: \_\_\_\_\_

ERROR REPORT NUMBER: \_\_\_\_\_

DATE: \_\_\_\_\_

DETAILS OF TESTING UNDERTAKEN:

FINDINGS:

ACTION TO BE TAKEN:



## **APPENDIX C**

### **TEMPORARY PROGRAM FIXES**

#### **1. INTRODUCTION**

This is a supplement to the 1985 CCVS X/Open Extension User Guide Version 4.2, containing the updates for correcting the errors in X/Open Extension tests that version of the CCVS85. The updates, referred to as Temporary Program Fixes (TPFs) are presented in the format required for use by the EXEC85 program, and will be applied in conducting a formal validation. The TPFs are presented in this document in the same order as the programs appear on the CCVS85 Population File, to facilitate preparing the updates for use.

1058

## 2. CORRECTIONS FOR CCVS85 X/OPEN EXTENSION TESTS V4.2

The following is a composite of the source code represented by the TPFs contained in this document.  
Card Columns:

0.....1.....2.....3.....4.....5.....6..  
\*BEGIN-UPDATE  
\*END-UPDATE