

COBOL and CICS Command Level
Conversion Aid
for OS/390 & MVS & VM



User's Guide

Version 2 Release 1

Note

Before using this information and the product it supports, be sure to read the general information under "Notices" on page 261.

Third Edition (September 2002)

This edition applies to COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM Version 2 Release 1 Modification 0 (CCCA, program number 5648-B05), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

This publication is available on the Web at:

www.ibm.com/s390/le/assist/tools/ccca.html

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation
H150/090
555 Bailey Avenue
San Jose, CA
95141-1003
U.S.A.

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 1982, 2013.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Tables	v
-------------------------	----------

Figures	vii
--------------------------	------------

About this book	ix
----------------------------------	-----------

How this book is organized	ix
--------------------------------------	----

How to read the syntax diagrams	x
---	---

Summary of changes	xiii
-------------------------------------	-------------

Fourth edition (July 2013)	xiii
--------------------------------------	------

Third edition (September 2002)	xiii
--	------

Second edition (October 1988)	xiv
---	-----

Chapter 1. Introduction	1
--	----------

What CCCA does.	1
-------------------------	---

Converting to COBOL 85 Standard Language	1
--	---

Converting using the Millennium Language	2
--	---

Extensions	2
----------------------	---

How CCCA works	4
--------------------------	---

BLL cell conversion	4
-------------------------------	---

Industry standards	5
------------------------------	---

Chapter 2. Getting started	7
---	----------

Dealing with source produced by earlier COBOL	7
---	---

compilers	7
---------------------	---

What to do before converting.	7
---------------------------------------	---

Accessing CCCA	8
--------------------------	---

Setting environment options (MVS only)	8
--	---

Navigating the menus and panels	10
---	----

Master menu	12
-----------------------	----

Converter menu	12
--------------------------	----

LCP Development Aid menu	14
------------------------------------	----

Options menu	15
------------------------	----

Chapter 3. Converting COBOL	17
--	-----------

programs	17
---------------------------	-----------

Setting source and target language levels	17
---	----

When the source and target language levels are	18
--	----

the same	18
--------------------	----

Setting conversion options	19
--------------------------------------	----

Submitting the conversion job under MVS	27
---	----

Running the conversion job under VM	32
---	----

Reading the Diagnostic listing	35
--	----

Conversion return codes	37
-----------------------------------	----

Chapter 4. DATE FORMAT Conversion	39
--	-----------

Option	39
-------------------------	-----------

Millennium Language Extensions (MLE) and Date	39
---	----

Fields	39
------------------	----

Definition of terms	39
-------------------------------	----

Date Field	40
----------------------	----

Century window	40
--------------------------	----

DATE FORMAT Clause	40
------------------------------	----

What you need to supply to CCCA	41
---	----

Date Identification file.	42
-----------------------------------	----

Selecting the DATE FORMAT Conversion Option.	46
--	----

How the DATE FORMAT Conversion Option works	46
---	----

Checking DATE FORMAT Clause syntax	46
--	----

Chapter 5. Conversion reports and the	49
--	-----------

conversion log	49
---------------------------------	-----------

Generating conversion reports	49
---	----

When there's nothing to report...	49
---	----

Program/File report	50
-------------------------------	----

File/Program report	53
-------------------------------	----

Copy/Program report	54
-------------------------------	----

Program/Copy report	54
-------------------------------	----

Call/Program report	55
-------------------------------	----

Program/Call report	56
-------------------------------	----

Using the conversion log	57
------------------------------------	----

Browsing or updating the conversion log	57
---	----

Erasing the conversion log	58
--------------------------------------	----

Chapter 6. Customizing CCCA	61
--	-----------

How CCCA invokes LCPs	63
---------------------------------	----

Customizing the way CCCA converts a language	64
--	----

element.	64
------------------	----

Customizing CCCA to convert an additional	64
---	----

language element	64
----------------------------	----

Updating the COBOL reserved word Data Set.	66
--	----

Compiling LCPs under MVS	68
------------------------------------	----

Compiling LCPs under VM	70
-----------------------------------	----

Deleting LCPs and activating/deactivating	71
---	----

debugging for LCPs	71
------------------------------	----

Generating a directory of the LCP library	72
---	----

Updating the message file	73
-------------------------------------	----

Chapter 7. Developing Language	77
---	-----------

Conversion Programs	77
--------------------------------------	-----------

What is an LCP?.	77
--------------------------	----

What LCPs do	77
------------------------	----

LCP structure.	77
------------------------	----

LCP divisions	78
-------------------------	----

LCP source line format	78
----------------------------------	----

Characters.	78
---------------------	----

Data item identifiers and paragraph names.	79
--	----

Reserved words	79
--------------------------	----

Literals.	80
-------------------	----

Comment lines	80
-------------------------	----

Punctuation	80
-----------------------	----

LCP statement summary	81
---------------------------------	----

Identification Division.	81
----------------------------------	----

CONVER statement	81
----------------------------	----

Data Division (Optional)	83
------------------------------------	----

Procedure Division	84
------------------------------	----

ADD statement	84
-------------------------	----

Conditions.	84
---------------------	----

EXIT statement	86
--------------------------	----

GO TO statement	86
IF statement	86
MOVE statement	88
Paragraph names	88
PERFORM statement	89
SUBTRACT statement	90
LCP functions	91
Using LCP functions	91
Retrieving tokenized source	92
Bypassing token identifiers	93
Removing tokenized source	94
Modifying tokenized source and inserting tokens	94
Editing tokens	96
Constructing tokens	96
Manipulating files	101
Control file	102
Work file	104
Using LCPs	106
Controlling LCP invocation	106
Processing LCPs	107
Tokenization	108
Debugging LCPs	110
Processing differences between tokens and elements	111

Appendix A. Converted COBOL language elements 117

Appendix B. Converted CICS commands 141

Linkage section	141
Working-Storage Section	141

Appendix C. Messages 145

Converter error messages	145
LCP compiler error messages	148
Tokenization diagnostics	150
Conversion diagnostics from LCPs	151

Appendix D. LCP reserved words . . . 167

Appendix E. Predefined data items 175

Appendix F. List of LCP functions . . 187

Appendix G. LCP directory 191

Converted CICS commands	191
-----------------------------------	-----

Completely converted COBOL statements	191
COBOL statements converted with warning	194
COBOL statements flagged	195
LCPs corresponding to information	197

Appendix H. Sample output 199

Program/File report	199
File/Program report	200
Copy/Program report	200
Call/Program report	201
LCP directory	201
Compilation of an LCP	205
COBOL conversion	211
COBOL conversion with COPY	217
COBOL conversion with CICS commands	222
Tokenization	238
LCP debugging	251

Appendix I. Maintaining CCCA under MVS 257

Re-installing CCCA	257
Applying service updates	257
What you receive	257
Checklist for applying service	257
Step 1. Prepare to install service	257
Step 2. Receive the service	258
Step 3. Accept applied service (optional)	258
Step 4. Apply the service	258
Step 5. Test the service	258
Step 6. Accept the service	258
Removing CCCA	258
Reporting a problem with CCCA	259
Obtaining service information	259

Notices 261

Programming interface information	262
Trademarks	262

Bibliography 263

IBM COBOL for OS/390 & VM	263
IBM COBOL for MVS & VM	263
Enterprise COBOL for z/OS & OS/390	263
Other publications	263

Index 265

Tables

1.	Source language levels	1	6.	Language elements converted to specified target language	117
2.	Target language levels	1	7.	Converted CICS commands.	142
3.	Valid combinations of source and target language levels	18	8.	Summary of steps for installing service on CCCA	257
4.	LCP characters—their meanings and uses	78	9.	Component IDs.	259
5.	LCP statement summary	81			

Figures

1. The three conversion phases	3	28. LCP compiler job statement information panel (MVS)	68
2. Environment Options panel	8	29. LCP Compiler Selection panel (MVS)	69
3. Map of CCCA menus and panels	11	30. LCP Compiler Submission panel (MVS)	69
4. Master menu	12	31. LCP Compiler Selection panel (VM)	70
5. Converter menu	13	32. LCP compiler member selection panel (VM)	71
6. LCP Development Aid menu.	14	33. Delete/Debug LCP panel	72
7. Options menu.	15	34. Extract from a directory of the LCP library	73
8. Language LEVEL panel	17	35. Messages panel	74
9. Conversion Options panel 1	19	36. Saving and repositioning TOKEN-POINTER	93
10. Conversion Options panel 2	23	37. Control and work file record relationships	102
11. Conversion Job Statement Information panel (MVS)	27	38. OTHERWISE LCP source code	107
12. Conversion Selection panel (MVS)	28	39. OTHERWISE LCP compilation listing	108
13. Conversion Member List panel (MVS)	28	40. Section of a tokenized COBOL source program	109
14. Conversion Submission panel (MVS)	31	41. Trace of OTHERWISE LCP execution	110
15. Conversion Selection panel (VM)	32	42. Section of the Diagnostic listing showing result of OTHERWISE conversion	111
16. Conversion Member Selection panel (VM)	33	43. TKNTEST LCP compilation listing	112
17. Extract from a diagnostic listing.	36	44. Source of program to be converted	112
18. Program/File report.	50	45. Tokenization of the COBOL source program containing tokens and elements (Part 1 of 2)	113
19. File/Program report.	53	46. Tokenization of the COBOL source program containing tokens and elements (Part 2 of 2)	114
20. Copy/Program report	54	47. Trace of TKNTEST LCP execution (Part 1 of 2)	115
21. Program/Copy report	55	48. Trace of TKNTEST LCP execution (Part 2 of 2)	116
22. Call/Program report	56		
23. Program/Call report	56		
24. Conversion Log panel	57		
25. Confirm Erase Log panel	59		
26. How CCCA invokes LCPs	62		
27. Reserved Words panel	66		

About this book

This book describes COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM (CCCA, program number 5648-B05).

CCCA helps you convert old COBOL 68 Standard and COBOL 74 Standard language in source programs and copy books to COBOL 85 Standard language. For a definition of these COBOL standards, see “Industry standards” on page 5.

CCCA can also help you to solve your Year 2000 problems by converting your programs to make use of the millennium language extensions (MLE).

How this book is organized

This book is divided into these chapters and appendixes:

Chapter 1, “Introduction,” on page 1

Summarizes what CCCA does, and how it works.

Chapter 2, “Getting started,” on page 7

Describes:

- What to do before converting
- Accessing CCCA
- Setting CCCA environment options
- Navigating CCCA menus and panels

Chapter 3, “Converting COBOL programs,” on page 17

Describes the procedure for converting COBOL programs:

- Setting source and target language levels
- Setting conversion options
- Submitting the conversion job
- Reading the Diagnostic listing

Chapter 4, “DATE FORMAT Conversion Option,” on page 39

Describes:

- Millennium language extensions and date fields
- MLE terms
- The DATE FORMAT clause
- What you need to supply to CCCA for the DATE FORMAT conversion option
- How to select the DATE FORMAT conversion option
- How the DATE FORMAT conversion option works

Chapter 5, “Conversion reports and the conversion log,” on page 49

Describes how to:

- Generate conversion reports
- Browse, update, and erase the conversion log

Chapter 6, “Customizing CCCA,” on page 61

Describes how to:

- Customize CCCA
- Update the COBOL Reserved Word data set
- Compile Language Conversion Programs (LCPs)
- Delete LCPs from the LCP library
- Activate and deactivate debugging for each LCP
- Print a directory of the LCP library

- Update messages

Chapter 7, “Developing Language Conversion Programs,” on page 77

Describes the language and functions you use to develop LCPs.

Appendix A, “Converted COBOL language elements,” on page 117

Lists COBOL language elements converted by CCCA.

Appendix B, “Converted CICS commands,” on page 141

Lists CICS® commands converted by CCCA.

Appendix C, “Messages,” on page 145

Lists CCCA messages.

Appendix D, “LCP reserved words,” on page 167

Lists words that have a special meaning to the LCP compiler (you cannot use these words for your LCP data item identifiers or LCP paragraph names).

Appendix E, “Predefined data items,” on page 175

Lists data items that are predefined by the LCP compiler.

Appendix F, “List of LCP functions,” on page 187

Lists functions you can use in LCPs.

Appendix G, “LCP directory,” on page 191

Lists the supplied LCPs.

Appendix H, “Sample output,” on page 199

Lists sample CCCA output:

- Reports
- LCP directory
- LCP compilation
- COBOL conversions
- Tokenization
- LCP trace (for debugging)

Appendix I, “Maintaining CCCA under MVS,” on page 257

Describes:

- Re-installing CCCA
- Applying Service Updates
- Removing CCCA
- Reporting a Problem with CCCA
- Obtaining Service Information

How to read the syntax diagrams

Throughout this book, syntax descriptions use the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line as indicated by the following symbols:

▶▶— The beginning of a statement.

—▶ The statement syntax is continued on the next line.

▶— The statement is continued from the previous line.

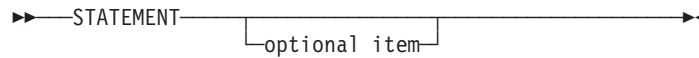
—▶▶ The end of a statement.

Diagrams of syntactical units other than complete statements start with the ▶— symbol and end with the —▶ symbol.

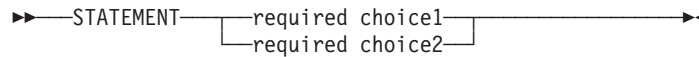
- Required items appear on the horizontal line (the main path).

▶▶—STATEMENT——required item————▶▶

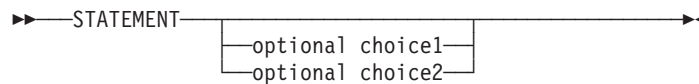
- Optional items appear below the main path.



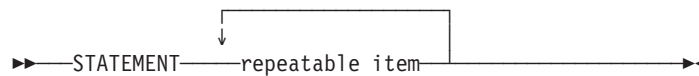
- When you can choose from two or more items, they appear vertically, in a stack. If you **must** choose one of the items, one item of the stack appears on the main path.



If choosing one of the items is optional, the entire stack appears below the main path.



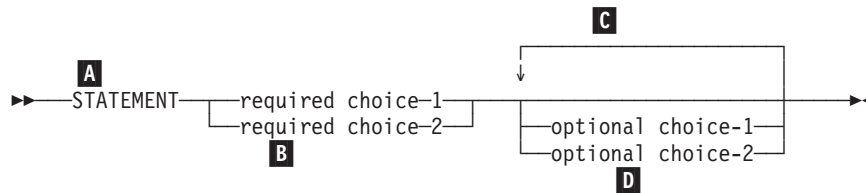
- An arrow returning to the left above the main line indicates an item that can be repeated.



A repeat arrow above a stack indicates that you can make more than one choice from the stacked items, or repeat a single choice.

- Language Conversion Program (LCP) key words appear in uppercase letters. They must be spelled exactly as shown. Variables appear in all lowercase letters. They represent user-supplied names or values.
- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, they must be entered as part of the syntax.

The following example shows how the syntax is used.



- A** The key word must be specified and coded as shown.
- B** One of these two options is required.
- C** This is a repeatable item.
- D** You can select one or more of these options.

Summary of changes

Fourth edition (July 2013)

The following major enhancements and changes have been made to this manual since the previous edition. All changes are marked in the text by a change bar in the left margin.

COBOL Version 5 changes:

- Removal of MLE functionality
- Removal of USE ... AFTER ... LABEL PROCEDURE ...

Third edition (September 2002)

The following major enhancements and changes have been made to this manual since the previous edition. All changes are marked in the text by a change bar in the left margin.

- Modifications to messages (Appendix C, "Messages," on page 145).
- Additional DATE FORMAT clauses (Chapter 4, "DATE FORMAT Conversion Option," on page 39).
- Modifications to the following language elements (Appendix A, "Converted COBOL language elements," on page 117):
 - ASSIGN
 - CBL
 - CURRENT-DATE
 - ERROR declaratives
 - IF
 - PERFORM
 - PROCESS
 - TIME-OF-DAY
 - TRANSFORM
 - UNSTRING
 - UPSI
 - VALUE
 - WHEN-COMPILED
- Process for deleting or debugging LCPs modified ("Deleting LCPs and activating/deactivating debugging for LCPs" on page 71).
- DLI option added to the Conversion Selection panel ("Submitting the conversion job under MVS" on page 27) to support the recognition of DLI processing.
- Process for maintaining the COBOL Reserved Word file modified ("Updating the COBOL reserved word Data Set" on page 66).
- Modifications to the Conversion Options Panel 2 ("Setting conversion options" on page 19).
- Additional predefined data items (Appendix E, "Predefined data items," on page 175).
- Enterprise COBOL for z/OS[®] and OS/390[®] added as a target language. COBOL for MVS[™] & VM and COBOL for OS/390 & VM combined into a single target language (IBM[®] COBOL) ("Setting source and target language levels" on page 17).

Second edition (October 1988)

The following major enhancements and changes have been made to this manual since the previous edition:

- Where it differs to MVS, the procedure for converting a COBOL program under VM has been added to Chapter 3, "Converting COBOL programs," on page 17. This new section appears under "Running the conversion job under VM" on page 32.
- Where it differs to MVS, the procedure for compiling an LCP under VM has been added to Chapter 6, "Customizing CCCA," on page 61. This new section appears under "Compiling LCPs under VM" on page 70.

Chapter 1. Introduction

This chapter summarizes:

- What CCCA does
- How CCCA works

What CCCA does

As supplied, COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM (CCCA) helps you convert COBOL source:

- To COBOL 85 Standard language
- To make use of the millennium language extensions (MLE)

Converting to COBOL 85 Standard Language

Using CCCA, you can convert COBOL source from the source language levels listed in Table 1 to any of the target language levels listed in Table 2.

Table 1. Source language levels

Source language	Version	Release	Program number
DOS/VS COBOL	1	3	5746-CB1
OS/VS COBOL	1	2	5740-CB1
VS COBOL II	1	1, 2, or 3	5668-958
COBOL for VSE/ESA	1	1	5686-068
COBOL for MVS & VM	1	2	5688-197
COBOL for OS/390 & VM	2	2	5648-A25
Enterprise COBOL	4	2	5655-G53

Table 2. Target language levels

COBOL 85 Standard language	Version	Release	Program number
VS COBOL II ¹	1	4	5668-958
COBOL for VSE/ESA	1	1	5686-068
COBOL for MVS & VM	1	2	5688-197
COBOL for OS/390 & VM	2	1	5648-A25
Enterprise COBOL for z/OS and OS/390	3	1	5655-G53
Enterprise COBOL for z/OS	5	0	5655-G53

Note:

1. DATE FORMAT conversion option cannot be used (see "Converting using the Millennium Language Extensions" on page 2).

CCCA identifies COBOL language elements and CICS commands in the input source programs that are:

- Not supported by the target language
- Supported in a different manner

then does one of the following:

- Converts them to the equivalent in the target language

Introduction

- Removes them
- Flags them

For details on how CCCA converts specific COBOL language elements and CICS commands, see Appendix A, “Converted COBOL language elements,” on page 117 and Appendix B, “Converted CICS commands,” on page 141.

Converting using the Millennium Language Extensions

If you plan to make use of the millennium language extensions to help solve your Year 2000 problem, an option within CCCA will help reduce the workload associated with converting your programs.

If you select this option, CCCA adds the DATE FORMAT clause to the data description entries of the data items that have been identified as containing dates. In the remainder of this document, this is referred to as the *DATE FORMAT conversion option*.

CCCA performs the DATE FORMAT conversion in addition to any other conversion required for converting to a different level of COBOL. The level of COBOL to which you are converting must support the DATE FORMAT clause.

If your program has been written using a level of COBOL that supports the DATE FORMAT clause but the program source does not include the DATE FORMAT clause, you can use CCCA to perform the DATE FORMAT conversion only. This applies to the following levels of COBOL:

- COBOL for VSE/ESA
- COBOL for MVS & VM
- COBOL for OS/390 & VM
- Enterprise COBOL for z/OS and OS/390 (pre Version 5)

In this case, you:

- Specify the same level of COBOL for both the source and target languages (see “Setting source and target language levels” on page 17)
- Select the DATE FORMAT conversion option (see Chapter 4, “DATE FORMAT Conversion Option,” on page 39)
- Set the conversion option *Remove obsolete elements* to N (see “Setting conversion options” on page 19)

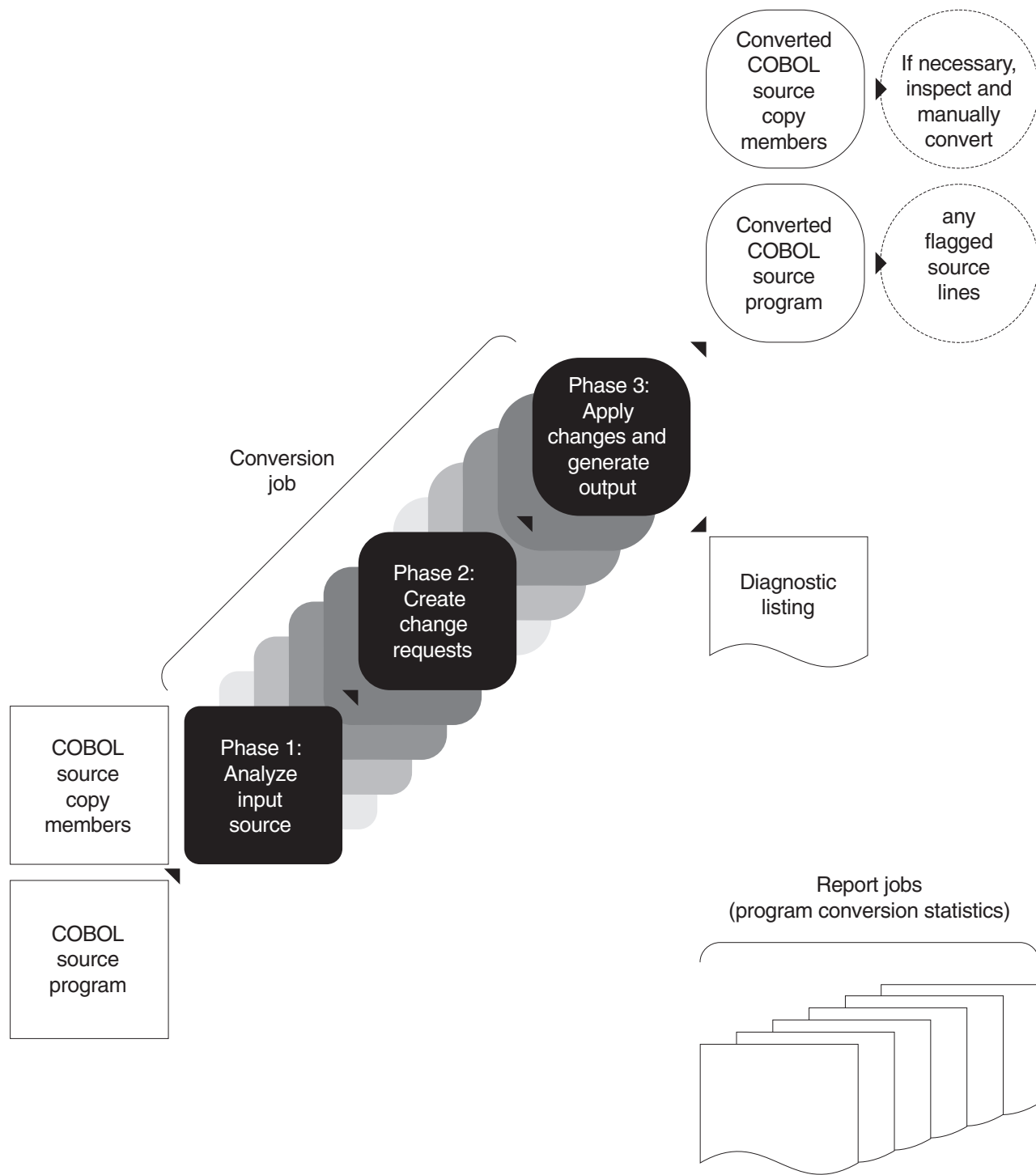


Figure 1. The three conversion phases

How CCCA works

CCCA is an interactive system comprising ISPF panels that enable you to access a batch (MVS) or foreground (VM) conversion application. You use CCCA online ISPF panels to:

- Define the type of conversion you want
- Submit a batch job (MVS), or run CCCA in foreground (VM), to convert your programs

Figure 1 on page 3 shows the three phases of a conversion job.

Phase 1: Analyze input source

At the start of a conversion job, phase 1:

- Extracts copy members from the appropriate copy libraries and merges them with the source program
- Translates the original source program and copy books into a set of character strings known as *tokenized source*
- For each language element in the tokenized source, identifies whether conversion is required, and if so, which Language Conversion Program (LCP) to use

Phase 2: Create change requests

For each item that needs converting, phase 2:

- Loads an LCP
- Runs the LCP
- Generates change requests

Phase 3: Apply changes and generate output

Finally, phase 3:

- Applies the change requests from phase 2, creating new source programs and, if required, new copy members
- Generates the Diagnostic listing

BLL cell conversion

CICS programs written in DOS/VS COBOL and OS/VS COBOL have to maintain addressability to storage not contained within the WORKING-STORAGE SECTION. In order to satisfy program requests, these programs must keep track of the storage area addresses allocated by CICS. This requires the manipulation of BLL cells within the application program.

For CICS programs written in any of the target languages, this is no longer required. The manipulation of BLL cells is no longer supported, so conversion of the source code is necessary. CCCA performs much of the required BLL cell conversion.

CCCA uses the CICS translator and the OS/VS COBOL compiler to perform the BLL cell conversion.

CCCA only performs BLL cell conversion if:

- - Under MVS, you have set the **CICS** option to Y on the Conversion (Selection) panel (see Figure 12 on page 28)
 - Under VM, you have set the **CICS** option to Y on the Conversion Selection panel (see Figure 15 on page 32),

- You have set the **Source language level** to 1, 2, 3, or 4 (DOS/VS or OS/VS COBOL) on the Language Level panel (see Figure 8 on page 17), and
- CCCA determines that there are BLL cells in the Linkage section of the source program to be converted.

To perform BLL cell conversion, CCCA:

- In phase 1, reduces the source program to a Linkage area
- In a number of intermediate steps between phase 1 and phase 2:
 - Translates and compiles the reduced program
 - Analyzes the compiler's glossary output
- Passes the compiler's glossary output to phase 2

Industry standards

The term “COBOL 68 Standard” is used in this document to refer to the following standards:

- X3.23-1968, American National Standard for Programming Language COBOL
- ISO International Standard 1989-1972 COBOL

The term “COBOL 74 Standard” is used in this document to refer to the following standards:

- X3.23-1974, American National Standard for Programming Language COBOL
- ISO International Standard 1989-1978 COBOL

The term “COBOL 85 Standard” is used in this document to refer to the following standards:

- X3.23-1985, American National Standard for Information Systems - Programming Language - COBOL
- X3.23a-1989, American National Standard for Information Systems - Programming Language - Intrinsic Function Module for COBOL
- ISO 1989:1985, Programming languages - COBOL
- ISO 1989/Amendment 1, Programming languages - COBOL - Amendment 1: Intrinsic function module

Introduction

Chapter 2. Getting started

This chapter describes:

- Dealing with source produced by earlier COBOL compilers
- What to do before converting
- Accessing CCCA
- Setting CCCA environment options
- Navigating CCCA menus and panels

Dealing with source produced by earlier COBOL compilers

The earlier OS/VS COBOL compilers contained a number of undocumented extensions. Where possible, CCCA attempts to handle these extensions. However, CCCA will not always correctly convert OS/VS COBOL code that compiles with warning-level or error-level diagnostics using the OS/VS COBOL 2.4 compiler.

If you have any OS/VS COBOL programs that you want CCCA to convert, which have not been compiled with the OS/VS COBOL 2.4 compiler, it is recommended that *before* you input these programs to CCCA you:

- Recompile each program using OS/VS COBOL 2.4
- Check for, and correct, any compiler-related warning-level or error-level diagnostics that result

Note: One notable undocumented extension of the pre-OS/VS COBOL 2.4 compiler that CCCA does *not* handle are COPY statements which are not terminated with a period (".").

Therefore, at the very least, you should ensure that all COPY statements, in any of your programs that you intend to convert using CCCA, are terminated with a period.

What to do before converting

Before using CCCA to convert your programs:

Decide whether to customize CCCA

Before converting any programs, you must decide on one of these courses of action:

- Use CCCA as supplied
- Customize CCCA

Most users will opt to use CCCA as supplied.

If, however, you require:

- Additional (possibly non-COBOL) language elements to be converted, flagged, or removed
- Particular language elements converted differently

then you can customize CCCA so that it meets your conversion requirements.

If you are interested in customizing CCCA, read Chapter 6, "Customizing CCCA," on page 61 and Chapter 7, "Developing Language Conversion Programs," on page 77.

Getting started

For a list of the COBOL language elements converted, removed, or flagged by CCCA as supplied, see Appendix A, “Converted COBOL language elements,” on page 117.

Ensure your source programs are error-free

Ensure your source programs compile and execute without errors.

To enhance conversion performance...

Setting the **Check procedure names** option to N reduces conversion time. For details, see page 23.

Restrictions

CCCA does not support certain COBOL 85 Standard language elements and certain IBM extensions in source code. Unsupported language elements include:

- Nested programs
- Program names that do not conform to the COBOL 85 Standard
- Object-oriented class and method definitions

Accessing CCCA

To access CCCA:

1. Log on to TSO (under MVS) or CMS (under VM)
2. Invoke ISPF
3. Select CCCA from your system's application menu—the Master menu appears (see Figure 4 on page 12)
4. **(MVS only)** If you have not already done so, you must set the environment options before you do anything else.

Setting environment options (MVS only)

To set the environment options:

- Go to panel **O.1** to display the Environment Options panel, shown in Figure 2.

```
----- CCCA Environment Options -----
COMMAND ==>

  High level qualifiers:
  Non-VSAM Shared Data Sets.. ==>
  Non-VSAM Private Data Sets. ==>
  VSAM Shared Data Sets..... ==>
  VSAM Private Data Sets..... ==>

  UNIT for Work Files ..... ==>
  CLIST debugging ..... ==>   Y/N

Job statement information:      (Verify before proceeding)
==>
==>
==>
==>

SYSOUT CLASS ==>

PF1 Help  PF3 Exit  PF4 Return  ENTER Save options
```

Figure 2. Environment Options panel

- Enter values for:

High Level Qualifiers

The data sets used by CCCA are divided into two categories, “Shared” and “Private”. Shared data sets are available to all users, and were created as part of the installation process. Each user requires a unique set of Private data sets which are used in read/write mode during conversion.

Non-VSAM Shared Data Sets

The high level qualifier name that has been assigned to the Non-VSAM Shared data sets. This name will be available from the system programmer who installed CCCA.

(Dialog variable ABJNVSH)

Non-VSAM Private Data Sets

The high level qualifier name to be assigned to the Non-VSAM Private data sets. The default is Userid.

(Dialog variable ABJNVPR)

VSAM Shared Data Sets

The high level qualifier name that has been assigned to the VSAM Shared data sets. This name will be available from the system programmer who installed CCCA.

(Dialog variable ABJVSSH)

VSAM Private Data Sets

The high level qualifier name to be assigned to the VSAM Private data sets. The default is Userid.

(Dialog variable ABJVSPR)

UNIT for Work Files

The unit on which the CCCA work files are allocated.

(Dialog variable ABJUNIT)

CLIST debugging

Y CCCA provides you with a statement-by-statement CLIST screen display to assist with error determination. Use this option if you are experiencing CLIST problems.

N No CLIST screen is displayed.

The default is N.

(Dialog variable ABJBUG)

Job statement information

The job card information for the batch job that CCCA submits. These lines are submitted as part of batch jobs exactly as they are entered (except for entirely blank lines, which are ignored). All of the rules of JCL must be followed. CCCA does not validate this information.

(Dialog variables BJC1, BJC2, BJC3, and BJC4)

See following **Note**.

SYSOUT CLASS

The output class to which you want your CCCA batch job output sent.

The output class can be:

- An asterisk (*)— indicating the default value for your environment.

Getting started

- Any letter (A through Z) or any numeral (0 through 9), indicating a specific output class.

Note: The output class that you enter on the Environment Options panel becomes the *default output class* for all subsequent jobs that you submit during the current session of CCCA. You can, however, assign a different output class for an individual job at the time of submitting that particular conversion job—see “Submitting the conversion job under MVS” on page 27 and “Compiling LCPs under MVS” on page 68.

- Press Enter.
JCL is generated to create any private data sets required by CCCA that currently do not exist.
The generated job consists of JCL to:
 - Define Sequential data sets required for installation verification.
 - Define required VSAM clusters, and load files from members in the sample library.CCCA creates an edit session for the generated JCL.
- You must modify this JCL (using the editor) to provide installation-specific information. When you have done this, use the TSO SUBMIT command to submit the job for batch processing.
- Once submission is completed, press Enter.
CCCA exits the edit session and returns to the Environment Options panel.

Navigating the menus and panels

To exit CCCA

If you are not at the Master Menu, press PF4.

From the Master Menu, press PF3 or PF4.

To select an option from a menu

In the Option ==> field, type the highlighted option number or letter then press Enter.

To go to any menu or panel from any other menu or panel

Type an equal sign (=) followed by the options you would enter to get there from the Master Menu. Separate the options with periods.

For example: to go to the Environment Options panel, type =O.1 in the Option ==> or Command ==> field, then press Enter.

To go to any menu or panel from the Master Menu

Type the options separated by periods (as above), without an equal sign.

For example: to go to the Conversion Log panel from the Master Menu, type 1.L in the Option ==> field, then press Enter.

The following keys have standard functions in CCCA:

- PF1** Displays Help for the current menu or panel
- PF3** Exits the current menu or panel, and returns to the previous menu
- PF4** Returns to the Master Menu from any menu or panel (except Help)
- Enter** Saves changes you have made to the current panel

Within Help, PF3 and PF4 exit the current Help panel, and take you back to the menu or panel you were at when you pressed PF1.

Figure 3 shows a map of CCCA menus and panels.

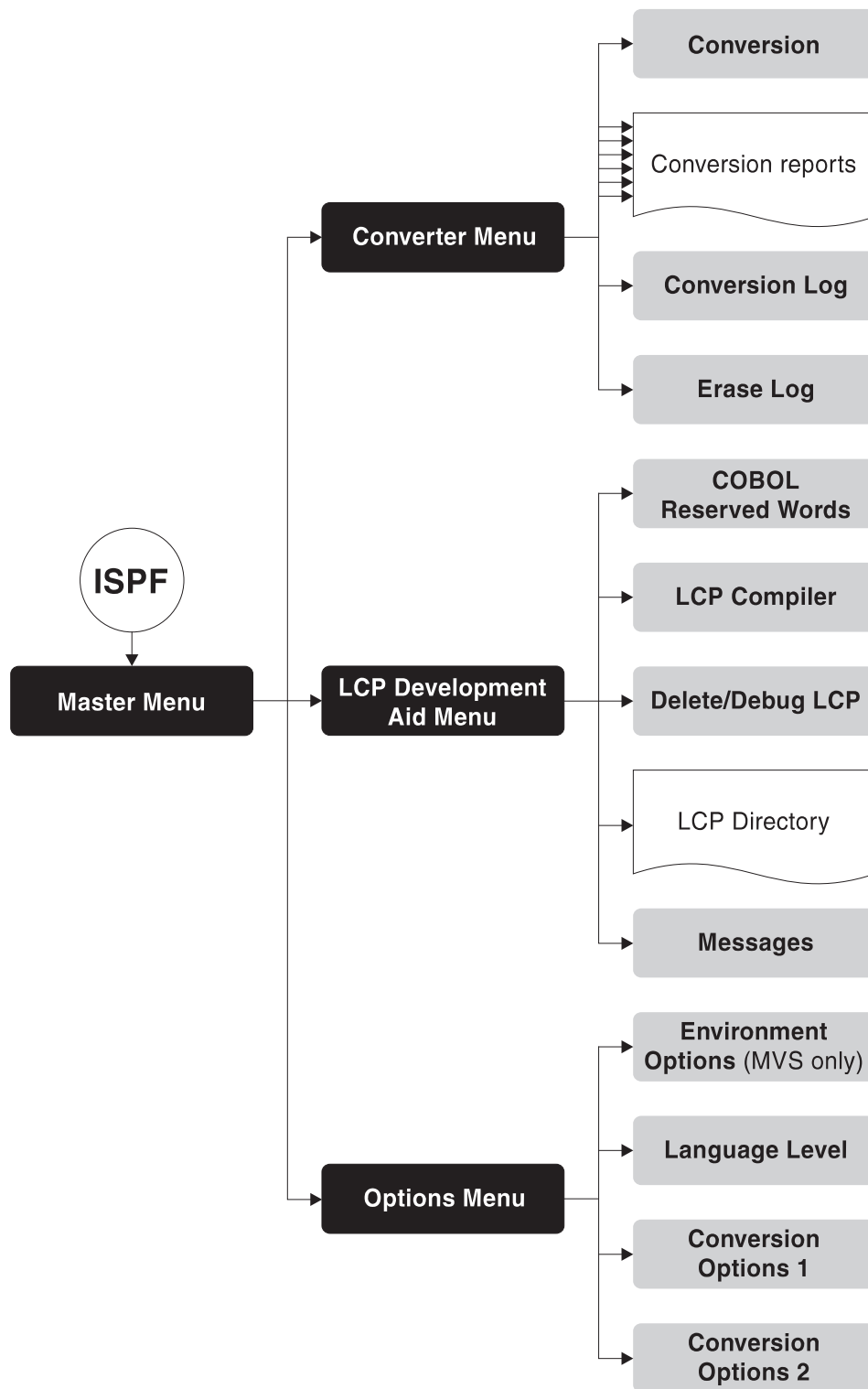


Figure 3. Map of CCCA menus and panels

The following sections describe the CCCA menus.

Master menu

The Master Menu shows the basic CCCA options (see Figure 4).

```
----- CCCA Master Menu -----  
Option ==>  
  
1 CONVERT - Convert COBOL source programs  
2 CUSTOMIZE - LCP Development Aid  
0 OPTIONS - Set environment and conversion options  
  
Userid - VCATRCA  
Terminal - 3278  
Time - 09:42  
PF Keys - 24  
Applid - ABJ  
  
COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM  
5648-B05 Version 2 Release 1  
Copyright (C) IBM Corp 1982, 1998 - All rights reserved  
  
PF1 Help PF3 Exit PF4 Return
```

Figure 4. Master menu

On this menu, you can select:

1 CONVERT

Shows the Converter Menu.

If you use CCCA as supplied, the Converter Menu contains all the functions you will need.

For details, see “Converter menu.”

2 CUSTOMIZE

Shows the Language Conversion Program (LCP) Development Aid Menu, containing options for customizing CCCA.

For details, see “LCP Development Aid menu” on page 14.

0 OPTIONS

Shows the Options Menu, with options for defining:

- High level qualifiers for CCCA VSAM files (MVS only)
- Source and target language levels
- CCCA conversion job and report job details
- Output that CCCA generates

For details, see “Options menu” on page 15.

Converter menu

To view the Converter Menu (shown in Figure 5 on page 13), select option 1 from the Master Menu.

```

----- CCCA Converter Menu -----
Option ==>

  1  OPTIONS          - Set environment and conversion options
  2  CONVERT PROGRAM - Convert COBOL source programs
  3  PROGRAM/FILE    - Generate Program/File report
  4  FILE/PROGRAM    - Generate File/Program report
  5  COPY/PROGRAM    - Generate Copy/Program report
  6  PROGRAM/COPY    - Generate Program/Copy report
  7  CALL/PROGRAM    - Generate Call/Program report
  8  PROGRAM/CALL    - Generate Program/Call report
  L  CONVERSION LOG  - Browse and update conversion statistics
  E  ERASE LOG       - Delete conversion statistics
PF1 Help  PF3 Exit  PF4 Return

```

Figure 5. Converter menu

On this menu, you can select:

1 OPTIONS

Shows the Options Menu.

For details, see “Options menu” on page 15.

2 CONVERT PROGRAM

Shows panels that allow you to submit a conversion job for one or more COBOL source programs.

For details, see “Submitting the conversion job under MVS” on page 27 or “Running the conversion job under VM” on page 32.

3 PROGRAM/FILE

4 FILE/PROGRAM

5 COPY/PROGRAM

6 PROGRAM/COPY

7 CALL/PROGRAM

8 PROGRAM/CALL

Generates a report of program conversion statistics.

For details, see Chapter 5, “Conversion reports and the conversion log,” on page 49.

L CONVERSION LOG

Shows a panel that allows you to:

- Browse a summary of program conversion statistics
- Update manual conversion statistics

For details, see “Using the conversion log” on page 57.

E ERASE LOG

Shows a panel that allows you to delete all program conversion statistics.

For details, see “Erasing the conversion log” on page 58.

LCP Development Aid menu

The LCP Development Aid Menu contains options for customizing CCCA.

If you use CCCA as supplied, you do not need to use this menu.

To view the LCP Development Aid Menu (shown in Figure 6), select option **2** from the Master Menu.

```
----- CCCA LCP Development Aid Menu -----
Option ==>

  1  RESERVED WORDS  - Update COBOL Reserved Word data set
  2  COMPILE LCP     - Compile LCP source
  3  DELETE/DEBUG LCP - Delete LCP or activate/deactivate debugging for an LCP
  4  LCP DIRECTORY  - Generate a directory of the LCP library
  5  MESSAGES       - Update Message file
  6  OPTIONS        - Set environment and conversion options
  7  CONVERT PROGRAM - Convert COBOL source programs

PF1 Help  PF3 Exit  PF4 Return
```

Figure 6. LCP Development Aid menu

On this menu, you can select:

1 RESERVED WORDS

Shows a panel that allows you to browse and update the COBOL Reserved Word data set.

For details, see “Updating the COBOL reserved word Data Set” on page 66.

2 COMPILE LCP

Shows a panel that allows you to submit a compile job for one or more LCP source members.

For details, see “Compiling LCPs under MVS” on page 68 or “Compiling LCPs under VM” on page 70.

3 DELETE/DEBUG LCP

Shows a panel that allows you to:

- Delete LCPs from the LCP library
- Activate or deactivate debugging for each LCP

For details, see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71

4 LCP DIRECTORY

Generates a directory of the LCP library.

For details, see “Generating a directory of the LCP library” on page 72.

5 MESSAGES

Shows a panel that allows you to browse, add, update, or delete CCCA messages.

For details, see “Updating the message file” on page 73.

6 OPTIONS

Shows the Options Menu.

For details, see “Options menu.”

7 CONVERT PROGRAM

Shows a panel that allows you to submit a conversion job for one or more COBOL source programs.

For details, see “Submitting the conversion job under MVS” on page 27 or “Running the conversion job under VM” on page 32.

Options menu

Before converting COBOL programs, you must specify the options you want to use. You can select the Options Menu in several ways:

- From the Master Menu, select option **O**
- From the Converter Menu, select option **1**
- From the LCP Development Aid Menu, select option **6**

Figure 7 shows the Options Menu.

```

----- CCCA Options Menu -----
Option ==>

  1  ENVIRONMENT      - Set environment options
  2  LANGUAGE         - Set language level
  3  CONVERSION       - Set conversion options 1
  4  CONVERSION       - Set conversion options 2

PF1 Help  PF3 Exit  PF4 Return

```

Figure 7. Options menu

On this menu, you can select:

ENVIRONMENT

Shows the Environment Options panel, where you specify:

- CCCA conversion and report job details
- High level qualifiers for CCCA VSAM files

For details, see “Setting environment options (MVS only)” on page 8.

LANGUAGE

Shows the Language Level panel, where you specify:

- Source language level CCCA converts from
- Target language level CCCA converts to

For details, see “Setting source and target language levels” on page 17.

Getting started

CONVERSION

Shows the Conversion Options panels, where you specify the output that CCCA generates.

For details, see “Setting conversion options” on page 19.

Chapter 3. Converting COBOL programs

This chapter describes the procedure for converting COBOL programs:

1. Setting source and target language levels
2. Setting conversion options
3. Submitting the conversion job
4. Reading the Diagnostic listing

Setting source and target language levels

CCCA converts programs from a *source* COBOL language level to a *target* COBOL language level.

To set the source and target language levels:

1. Go to the Language Level panel (O.2), shown in Figure 8.

```
----- CCCA Language Level -----
Command ==>

Source language level ==> 3  1. DOS/VS COBOL LANGLVL(1)
                             2. DOS/VS COBOL LANGLVL(2)
                             3. OS/VS COBOL LANGLVL(1)
                             4. OS/VS COBOL LANGLVL(2)
                             5. VS COBOL II Release 1.0 1.1 2.0, or
                               any COBOL with the CMPR2 option
                             6. VS COBOL II NOCMPR2 Release 3.0 3.1 3.2
                             7. VS COBOL II NOCMPR2 Release 4.0
                             8. COBOL/370 NOCMPR2
                             9. COBOL for VSE/ESA NOCMPR2
                             10. COBOL for MVS & VM NOCMPR2
                             11. COBOL for OS/390 & VM NOCMPR2
                             12. Enterprise COBOL (prior to Version 5)

Target language level ==> 4  1. VS COBOL II
                             2. COBOL for VSE/ESA
                             3. IBM COBOL
                             4. Enterprise COBOL for z/OS & OS/390
                             5. Enterprise COBOL V5

PF1 Help  PF3 Exit  PF4 Return  Enter Save options
```

Figure 8. Language LEVEL panel

2. Update the panel options:

Source language level

The language level of the program you are converting:

1. DOS/VS COBOL—LANGLVL(1) (COBOL 68 Standard)
2. DOS/VS COBOL—LANGLVL(2) (COBOL 74 Standard)
3. OS/VS COBOL—LANGLVL(1) (COBOL 68 Standard)
4. OS/VS COBOL—LANGLVL(2) (COBOL 74 Standard)
5. VS COBOL II (COBOL 74 Standard) Release 1.0, Release 1.1, or Release 2.0 (or any COBOL with the CMPR2 option)
6. VS COBOL II—NOCMPR2 (COBOL 85 Standard) Release 3.0, Release 3.1, or Release 3.2

Converting

- 7 VS COBOL II—NOCMPR2 (COBOL 85 Standard) Release 4.0
- 8 COBOL/370 NOCMPR2 (COBOL 85 Standard)
- 9 COBOL for VSE/ESA NOCMPR2 (COBOL 85 Standard)
- 10 COBOL for MVS & VM NOCMPR2 (COBOL 85 Standard)
- 11 COBOL for OS/390 & VM NOCMPR2 (COBOL 85 Standard)
- 12 Enterprise COBOL (prior to Version 5)

Default is 3.

Target language level

The language level (COBOL 85 Standard) you want the program converted to:

- 1 VS COBOL II—NOCMPR2 Release 4
- 2 COBOL for VSE/ESA NOCMPR2 Release 1
- 3 IBM COBOL (COBOL for MVS & VM NOCMPR2 Release 2, and COBOL for OS/390 VM NOCMPR2 Version 2 Release 2)
- 4 Enterprise COBOL for z/OS and OS/390 Version 3 Release 1
- 5 Enterprise COBOL V5

Default is 5.

Note: If you select target language level 2, 3, or 4, you can also select the DATE FORMAT conversion option (option 8 on the Conversion Options Panel 2—see Figure 10 on page 23).

3. Press Enter to save the options.

Table 3 shows the valid combinations of source and target language levels.

Table 3. Valid combinations of source and target language levels

Target Language Level	Source Language Level											
	1	2	3	4	5	6	7	8	9	10	11	12
1 ¹	✓	✓	✓	✓	✓	✓	✓ ³					
2 ²	✓	✓	✓	✓	✓	✓	✓	✓	✓ ³			
3 ²	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ ³		
4 ²	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ ³	
5 ²	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓ ³

Note:

1. Does not perform the DATE FORMAT conversion.
2. Target language level supports the DATE FORMAT clause.
3. Source and target language levels are the same. For the types of conversion that CCCA performs, see “When the source and target language levels are the same.”

When the source and target language levels are the same

Even if you set the target language level to the same as the source language level, CCCA may still perform some conversion, depending on the conversion options you have selected:

DATE FORMAT conversion option

If your program has been written using a level of COBOL that *supports* the DATE FORMAT clause but the program source does not *include* DATE FORMAT clauses, you can use CCCA to simply perform the DATE FORMAT conversion. This applies to the following levels of COBOL:

- COBOL for VSE/ESA
- IBM COBOL
- Enterprise COBOL (prior to version 5 levels)

In this case, you specify the same level of COBOL for both the source and target languages and select the DATE FORMAT conversion option.

For full details, see Chapter 4, “DATE FORMAT Conversion Option,” on page 39.

Remove obsolete elements conversion option: You can use CCCA to simply remove language elements that have become obsolete with the COBOL 85 Standard.

In this case, you specify the same level of COBOL for both the source and target languages and select the *Remove obsolete elements* conversion option.

For details, see Figure 10 on page 23.

Setting conversion options

Conversion options determine the output generated by conversion jobs.

To set the conversion options:

1. Go to the Conversion Options panel 1 (O.3), shown in Figure 9.

```

----- CCCA Conversion Options 1 -----
Command ==>

Lines per report page . . . . . ==> 60          01 to 99
VSE system date format. . . . . ==>           MM/DD/YY or DD/MM/YY
Resequence source lines . . . . . ==> N        Y/N
Sequence number increment . . . . . ==> 0010    0001 to 9999

Reserved word suffix. . . . . ==> 74
Generate new program. . . . . ==> Y          Y/N
Generate new copy members . . . . . ==> Y      Y/N
Replace like-named copy members ==> N        Y/N
Print old source lines. . . . . ==> Y        Y/N
Print copy members. . . . . ==> Y          Y/N
Print diagnostics of level >= . ==> 00      00 to 99
Report heading. . . . . ==> SAMPLE RUN
Generate tokenization listing . ==> N        Y/N

PF1 Help  PF3 Exit  PF4 Return  Enter Save options
    
```

Figure 9. Conversion Options panel 1

2. Update the panel options:

Lines per report page

The number of lines per page on the Diagnostic listing and conversion reports.

Converting

Must be in the range 01 to 99.

Default is 60.

VSE system date format

(For converting DOS/VS COBOL only.)

The date format used by the VSE system on which the old program ran:
MM/DD/YY or **DD/MM/YY**.

CCCA uses this date format to convert the **CURRENT-DATE** and **WHEN-COMPILED** special registers.

Note: This entry field only appears if the source language level on the Language Level panel is 1 or 2 (see Figure 8 on page 17).

Resequence source lines

Either:

Y CCCA resequences line numbers in columns 1 through 6 of the new source program and new source copy members, according to the **Sequence number increment** option (see below).

N CCCA does not resequence line numbers.

Default is N.

Sequence number increment

(Only has an effect if the **Resequence source lines** option is set to Y.)

Increment for resequenced line numbers.

Must be in the range 0001 to 9999.

Default is 0010.

Reserved word suffix

If the program you are converting contains user-defined words that are reserved words in the target language, CCCA appends this suffix to the user-defined words. (If left unchanged, these words would receive compiler errors from the target language compiler.)

Must be a two-digit number.

Default suffix is 74.

Generate new program

Either:

Y CCCA generates a new source program.

CCCA puts the new source program in the:

- (MVS only) **Output source—Program library** specified on the Conversion Selection panel (see Figure 12 on page 28).
- (VM only) **Output Source—Program file name** specified on the Conversion Selection panel (see Figure 15 on page 32).

Note: If the **Generate new program** option is set to Y, then CCCA generates new source members regardless of whether there were any changes applied.

N CCCA does not generate a new source program.

Default is Y.

Note: CCCA generates a Diagnostic listing whether or not it generates a new source program.

Generate new copy members

Either:

Y CCCA generates new source for copy members called by the source program.

CCCA puts the new source copy members in the:

- (MVS only) **Output source—Copy library** specified on the Conversion Selection panel (see Figure 12 on page 28).
- (VM only) **Output source—Copy library (MACLIB)** specified on the Conversion Selection panel (see Figure 15 on page 32).

If the copy member already exists, CCCA does not replace it, unless the **Replace like-named copy members** option is set to Y (see below).

CCCA does not issue any message or warning if it does not replace a copy member.

Note: If the **Generate new copy members** option is set to Y, then CCCA generates new source members regardless of whether there were any changes applied.

N CCCA does not generate new source copy members.

Default is Y.

Replace like-named copy members

(Only has an effect if the **Generate new copy members** option is set to Y.)

If the new source copy member already exists in the output copy library:

Y CCCA replaces it.

N CCCA does not replace it.

Default is N.

Print old source lines

Either:

Y Old source lines appear in the Diagnostic listing immediately before the converted or flagged line, with *OLD** in place of the sequence number.

For example:

000182	*OLD**	OTHERWISE	ABJ6021 00 OTHERWISE REPLACED BY ELSE
000183		ELSE	

*OLD** usually indicates a change has been made, or a manual change should be made, to *this* line. Sometimes, however, *OLD** appears on a line because there are added or deleted language elements *related* to that line.

For example, CCCA flags the WORKING-STORAGE SECTION header with *OLD** because the related line 77 LCP-FILE-STATUS-01 PIC XX. is inserted immediately after.

Converting

N Old source lines do not appear in the Diagnostic listing.

Default is Y.

Print copy members

Either:

Y CCCA prints copy members (specified in COPY statements) in the Diagnostic listing.

N CCCA does not print copy members in the Diagnostic listing.

Default is Y.

Print diagnostics of level >=

CCCA prints diagnostics of severity greater than or equal to this value.

Must be in the range 00 to 99.

Default is 00 (CCCA prints all diagnostics).

CCCA issues a diagnostic of severity level:

00 when it converts a language element.

04 when it converts a language element, but the converted language element may require additional, manual conversion. The new source program that contains this converted language element may compile and run successfully, but you should still manually inspect the converted code.

08 when a language element is encountered that either needs to be, or may need to be, manually converted.

Report heading

The heading that appears at the top of each page of the Diagnostic listing and conversion reports.

Maximum length is 25 characters.

Default is SAMPLE RUN.

Generate tokenization listing

Either:

Y CCCA generates a tokenization listing (see "Tokenization" on page 238).

N CCCA does not generate a tokenization listing.

Default is N.

3. Press Enter to save the options.
4. Go to the Conversion Options panel 2 (**O.4**), shown in Figure 10 on page 23.

```

----- CCCA Conversion Options 2 -----
Command ==>
Option
1. Check procedure names . . . . . ==> Y Y/N
2. Flag Report Writer statements . . . . . ==> Y Y/N
3. Remove obsolete elements. . . . . ==> N Y/N
4. Negate implicit EXIT PROGRAM. . . . . ==> N Y/N
5. Generate END PROGRAM header . . . . . ==> N Y/N
6. Compile after converting. . . . . ==> Y Y/N
7. Flag manual changes in new source program . . ==> Y Y/N
8. Add DATE FORMAT clause to date fields . . . ==> Y Y/N
9. Remove VALUE clauses in File/Linkage Sections ==> Y Y/N
10. Flag FILE-STATUS conditional statements . . . ==> Y Y/N
11. Flag BLL cell arithmetic. . . . . ==> Y Y/N
12. BLL cell conversion method. . . . . ==> B A/B
13. Search source for literal delimiter . . . . . ==> Y Y/N
14. Literal delimiter (QUOTE or APOST). . . . . ==> A Q/A
15. . . . . ==> N Y/N

Note: Option numbers appear on the Program/File report

PF1 Help  PF3 Exit  F4 Return  ENTER Save options

```

Figure 10. Conversion Options panel 2

5. Update the panel options:

Check procedure names

(For converting DOS/VS COBOL or OS/VS COBOL programs only.)

- Y** CCCA flags the following language elements in the Diagnostic listing:
- CALL...USING statements that specify a procedure name in the USING option.
 - USE FOR DEBUGGING declaratives that specify a name that is not a procedure name.
- N** CCCA does not flag these language elements.

Default is Y.

Note: You must convert these language elements. Flagging is optional for performance reasons; setting the option to N reduces conversion time.

Flag Report Writer statements

(For converting DOS/VS COBOL or OS/VS COBOL programs only.)

- Either:
- Y** CCCA flags Report Writer statements in the Diagnostic listing.
- N** CCCA does not flag Report Writer statements.

Default is Y.

Remove obsolete elements

- Either:
- Y** CCCA removes language elements that have become obsolete with the COBOL 85 Standard.
- N** CCCA does not remove obsolete elements.

Default is Y.

Converting

Note: These obsolete elements will not be supported in the next COBOL standard. It is therefore highly recommended that any such elements are removed (option Y).

Negate implicit EXIT PROGRAM

(For converting COBOL 68 Standard and COBOL 74 Standard programs only—see Source language level.)

Either:

Y If the last physical statement in the program is not EXIT PROGRAM, STOP RUN, or GOBACK, CCCA adds to the end of the program a section that includes a CALL to an abend module.

N CCCA does not add this section.

Default is Y.

Generate END PROGRAM header

Either:

Y CCCA adds an END PROGRAM header to the end of the new source program.

N CCCA does not add an END PROGRAM header.

Default is N.

Compile after converting

Either:

Y After conversion, the new source is compiled by the target language compiler.

Note: The new source is not compiled if the program conversion receives a return code of 08 or higher.

The return code of the compile appears in the Program/File report.

N The new source program is not compiled.

Default is Y.

Flag manual changes in new source programs

Either:

Y CCCA inserts a flagging line in the new source program before any line with diagnostic level 08 or higher, indicating that this line requires manual conversion.

The new source program will not compile unless you remove this flagging line. This ensures that you do not overlook any lines with this level of diagnostic.

If you want to use this option, but there are some diagnostics of level 08 that you don't want flagged, change the severity level of these diagnostics using the Messages panel. See "Updating the message file" on page 73.

N CCCA does not insert flagging lines.

Default is N.

Add DATE FORMAT clause to date fields

Either:

Y CCCA adds a DATE FORMAT clause to the data description entry

of each data item that has been identified as being used to contain a date. (The names of these data items are specified in the date identification file—see Chapter 4, “DATE FORMAT Conversion Option,” on page 39 for a full description.)

Note:

- a. You can only select this option if the target language level is set to 2, 3, or 4—see “Setting source and target language levels” on page 17.
- b. You enter the name of the date identification file on the Conversion Selection panel (MVS)—see Figure 12 on page 28, or the Conversion Selection panel (VM)—see Figure 15 on page 32.

N CCCA does not add DATE FORMAT clauses.

Default is N.

Remove VALUE clauses in File/Linkage Sections

(For converting DOS/VS COBOL or OS/VS COBOL programs only.)

Either:

Y CCCA removes any VALUE clauses from data items (which are not level 88) in either the File or Linkage sections of the program.

N CCCA does not remove VALUE clauses.

Default is Y.

Flag FILE-STATUS conditional statements

(For converting COBOL 68 Standard and COBOL 74 Standard programs only—see Source language level.)

Either:

Y CCCA flags all conditional statements that check a FILE STATUS variable (IF, PERFORM... UNTIL..., SEARCH... WHEN...).

N CCCA does not flag conditional statements that check a FILE STATUS variable.

Default is Y.

Flag BLL cell arithmetic

(For converting CICS programs written in DOS/VS COBOL or OS/VS COBOL only.)

Either:

Y CCCA flags any statements where arithmetic is being performed on a CICS BLL cell.

N CCCA does not flag statements where arithmetic is being performed on a CICS BLL cell.

Default is Y.

BLL cell conversion method

(For converting CICS programs written in DOS/VS COBOL or OS/VS COBOL only.)

In order to identify the BLL cells in the program, CCCA invokes the OS/VS COBOL compiler to compile sections of the source program.

Converting

Either:

- A** CCCA compiles the Linkage section of the source program only.
- B** CCCA compiles the Working Storage and the Linkage sections of the source program.

Default is A.

Note: Normally, it is sufficient (and fastest) to use option A. However, if the compile fails due to there being references in the Linkage section to the Working Storage section, then you should resubmit the conversion using option B.

Search source for literal delimiter

Either:

- Y** CCCA uses the following procedure to determine the value of the literal delimiter used in the program:
 - a. CCCA scans the CBL cards of the source program for the QUOTE or APOST compiler options. If a CBL card is found that specifies one of these compiler options, CCCA uses that value as the delimiter. (If both QUOTE and APOST are specified, CCCA uses the last value.)
 - b. If there are no CBL cards, or neither the QUOTE nor APOST compiler option is specified, CCCA scans the source and copy code until it finds a quote or an apostrophe that is:
 - Not in a comment line
 - Not in a comment paragraph
 - Not in a NOTE statement (DOS/VS and OS/VS COBOL only)and, if found, uses that value as the delimiter.
 - c. If after scanning the source and copy code, CCCA has not determined a value for the literal delimiter, CCCA will use the value specified for the option **Literal delimiter (QUOTE or APOST)**—see following description.
- N** CCCA does not search for the literal delimiter and uses the value specified for the option **Literal delimiter (QUOTE or APOST)**—see following description.

Default is Y.

Literal delimiter (QUOTE or APOST)

CCCA only uses the value of the literal delimiter specified here when one of these conditions is true:

- The option **Search source for literal delimiter** is set to N
- The option **Search source for literal delimiter** is set to Y but, after searching the source and copy code, CCCA cannot find a value for the delimiter

Either:

- Q** Indicates a literal delimiter of a quote (")
- A** Indicates a literal delimiter of a apostrophe (')

Default is Q.

6. Press Enter to save the options.

Submitting the conversion job under MVS

Use the Conversion panels to submit a batch job to convert one or more programs.

To submit a conversion job:

1. Go to panel **1.2** to display the Conversion Job Statement Information panel, shown in Figure 11.

```

----- CCCA Conversion job statement information -----
Command ==>

Job statement information:      (Verify before proceeding)
====> //VCATRCA JOB (9999,040,090,ST3),'CCCA',
====> // NOTIFY=VCATRCA,TIME=5,
====> // REGION=4096K,USER=VCATRCA,MSGCLASS=V,CLASS=C
====> /*JOBPARM FORMS=SP1

SYSOUT class ==> *

PF1 Help  PF3 Exit  PF4 Return  ENTER Proceed

```

Figure 11. Conversion Job Statement Information panel (MVS)

2. If necessary, update the text in:

Job statement information

The JCL for the conversion job card.

SYSOUT class

The output class. to which you want the output of the conversion job sent.

SYSOUT class can be:

- Any letter (A through Z)
- Any numeral (0 through 9)
- An asterisk (*)

3. Press Enter to display the Conversion Selection panel (see Figure 12 on page 28).

```

----- CCCA Conversion selection -----
Command ==>
Program source:                                Options:
Project . . . ==> VCATRC2                      Language level ==> *    (* 1-11)
Library . . . ==> CCCA                        CICS . . . . . ==> Y    (Y N)
Type . . . . . ==> SOURCE                     SQL . . . . . ==> N    (Y N)
Member . . . . . ==>                          DLI . . . . . ==> N    (Y N)
                                           (Blank for member list, * for all members)

Other source file:
Data set name ==>

Copy libraries:
DDNAME ==> SYSLIB    LIBRARY ==> 'CCCA.REGTEST.PIRCPY1'
==>                                ==> 'VCATRC2.CCCA.COPYLIB'
==>                                ==> 'CCCA.REGTEST.PDSE'
==>                                ==> 'CCCA.V2R1.LEVEL2.SABJSAM1'
==>                                ==> 'TAUTEST.CCCACPY'
==>                                ==>

Output source:
Program library ==> 'VCATRC2.CCCA.OUTSRCE'
Copy library. . ==> 'VCATRC2.CCCA.OUTCPY'
Date identification file:
Data set name* ==> 'VCATRC2.CCCA.MLESEED'
*If PDS without member name, then program source member names used.
PF1 Help  PF3 Exit  PF4 Return  ENTER Build JCL
    
```

Figure 12. Conversion Selection panel (MVS)

4. Enter values for:

Program source

If the program source that you want converted is in a sequential data set, enter the data set name in the usual manner in the *Other source file* field (**Data set name**).

If the program source that you want converted is in a partitioned data set, enter the data set name and the member name in the usual manner in either the *Program source* fields (**Project**, **Library**, **Type**, and **Member**) or the *Other source file* field (**Data set name**). If you want all members of the data set converted, enter an asterisk (*) instead of the member name. If you do not specify a member name or an asterisk, a member list will be displayed after you press Enter (see Figure 13). Place an "S" in front of all members in the list that you want converted.

```

Functions  Help
-----
MEMBER LIST  VCATRCA.OLDV.S.PMR                               Row 00001 of 00016
Command ==>                               Scroll ==> PAGE
Name      VV MM Created    Changed    Size  Init  Mod  ID
- BRAD1                               01.56 97/03/27 98/01/12 10:01  46   8   46 VCATRC2
- BRAD2   SELECTED                    01.26 97/04/15 97/09/30 11:11   78  40   0 VCATRC2
- BRAD3                               01.02 97/04/16 97/04/16 15:25  318 316   0 VCATRCA
- CANCEL                    01.00 98/02/10 98/02/10 14:34   17  17   0 VCATRC2
- CCCA88
- COBCICS1                    01.01 97/11/14 97/11/19 10:30   47  42   0 VCATRC2
- DF0100
- KAMJ32P
- KEE     NO_D.I.F                    01.08 96/12/30 97/03/07 11:24   20  17   0 VCATRC2
- KEE2                               01.23 96/12/30 97/04/10 16:13   40  18   0 VCATRCA
- KEE3                               01.01 96/12/31 96/12/31 09:28   18  18   0 VCATRCA
- KEE4                               01.01 97/01/20 97/01/20 15:55   19  19   0 VCATRC2
- KEE7                               01.04 97/03/07 97/04/01 10:24   14  12   0 VCATRC2
- LCPTST
- P05812PG                    01.01 97/11/21 97/11/21 14:09  2143 2143   0 VCATRC2
- W64582
- **End**
    
```

Figure 13. Conversion Member List panel (MVS)

Note: If you have selected the DATE FORMAT conversion option (option 8 on the Conversion Options panel 2—see Figure 10 on page 23), and you have not specified a specific member for the date identification file on the Conversion Selection panel, the message “NO_D.I.F” appears against any member that you select if that member name does not exist in the date identification file data set.

Options

Most options for the conversion are specified in the option panels. Three however can be set on this panel:

Language level

Overrides—for this conversion job only—the **Source language level** specified on the Language Level panel. For a list of source language level values, see “Setting source and target language levels” on page 17.

If you specify an asterisk (*), CCCA uses the value specified in the Language Level panel.

CICS

Either:

- Y The program you are submitting for conversion contains EXEC CICS commands.
- N The program does not contain EXEC CICS commands.

SQL

Either:

- Y The program you are submitting for conversion contains SQL statements in the Linkage Section.
- N The program does not contain SQL statements in the Linkage Section.

DLI

Either:

- Y The program you are submitting for conversion contains EXEC DLI statements.
- N The program does not contain EXEC DLI statements.

Copy libraries

(Only required if the program you are converting contains COPY statements.)

The copy libraries of the old source copy members:

- The copy libraries are usually accessed through ddname SYSLIB. The COPY statement gives the member name in the library as specified by the SYSLIB DD statement.
- The COPY statement may also indicate a specific library.

For example:

COPY MOD OF LIB1

or

COPY MOD IN LIB1

In this case, the library is accessed by specifying a ddname that defines the data set itself.

Converting

```
DDNAME ==>SYSLIB  LIBRARY ==>'CCCA.INCLUDE.LIB1'  
DDNAME ==>        LIBRARY ==>'CCCA.INCLUDE.LIB2'  
DDNAME ==>        LIBRARY ==>'CCCA.INCLUDE.LIB3'  
DDNAME ==>LIB1    LIBRARY ==>'CCCA.SPECIAL.INCLUDE.LIB1'  
DDNAME ==>        LIBRARY ==>'CCCA.SPECIAL.INCLUDE.LIB2'
```

Concatenation

Concatenation of libraries is possible for any ddname that you specify. The normal MVS rules for concatenation of libraries apply, and you must ensure that the data set with the largest block size is listed first.

In this case, if different modules have the same name in different libraries, the module copied is the first encountered in the sequence of the libraries.

As the above example shows, you can concatenate up to six libraries under ddname SYSLIB or any other ddname.

Output source (Program library)

(You can specify this only if the **Generate new program** field on Conversion Options panel 1 is set to Y.)

The output library that you specify should be the same organization as the input library.

CCCA puts the new source program into this library.

If the data set is partitioned, the member name of the new source program will be the same as the member name of the old source program in the input library.

If a member with this name already exists in the output library, CCCA replaces it.

CCCA checks that the library you specify is:

- A valid library name and that it exists
- Not the same as the input source library
- Not the same as any of the input copy libraries
- Not the same as the output copy library

Note: You must enter the name of a library, even if the **Generate new program** field is set to N.

Output source (Copy library)

(You can specify this only if the **Generate new copy members** field on Conversion Options panel 1 is set to Y.)

CCCA puts new source for copy members called by the source program into this library. The new copy member will have the same name as the old copy member.

If a member with this name already exists in the output copy library, it is not replaced unless the **Replace like-named copy members** field on Conversion Options panel 1 is set to Y.

CCCA checks that the library you specify is:

- A valid library name and that it exists
- A partitioned data set
- Not the same as any of the input copy libraries
- Not the same as the input source library
- Not the same as the output source library

You can specify only one output copy library.

Date identification file

If you have selected the DATE FORMAT conversion option (option 8 on the Conversion Options panel 2—see Figure 10 on page 23), an entry field appears on the Conversion Selection panel into which you enter the data set name for the date identification file.

Note:

- a. If you specify a PDS without a member name, CCCA uses the same member name as for the program source.
 - b. If the input source is a sequential file, you must specify a member for the date identification file.
 - c. If you specify a PDS and a member name, CCCA searches that member for entries that match the program name of the source program. For a detailed description, refer to “Date Identification file” on page 42.
 - d. If you specify a sequential file, CCCA searches the file for entries that match the name of the source program. For a detailed description, refer to “Date Identification file” on page 42.
5. Press Enter.
ISPF generates the JCL for the conversion and then displays the Conversion Submission panel (see Figure 14).

```

----- CCCA Conversion submission -----
Command ==>

Instructions:
  Press ENTER  to continue generating JCL.
  Press PF3   to submit job and exit
  Press PF4   to submit job and return
  Press PF12  to exit without submitting job
  Enter C     command to exit without submitting job.

          1 member(s) built for conversion.
          1 selection(s) ignored because no date identification member found

Job statement information:
//VCATRCAG JOB (9999,040,090,ST3),'CCCA',
// NOTIFY=VCATRCA,TIME=5,
// REGION=4096K,USER=VCATRCA,MSGCLASS=V,CLASS=C
/*JOBPARM FORMS=SP2

PF1 Help   PF3 Submit Job   PF4 Submit job   PF12 Cancel   ENTER Generate JCL
           and exit       and return
  
```

Figure 14. Conversion Submission panel (MVS)

The Conversion Submission panel shows how many members have been selected (and also how many selections have not been successful) and redisplay the Job card parameters for information only. This panel can no longer be overtyped, since the Job statement has already been generated.

To select additional programs to be converted, press Enter,
To cancel the submission of the job, type C on the command line and press Enter.

- 6. Press either PF3 or PF4.
ISPF submits the generated JCL for execution.
The message JOB xxxxxc SUBMITTED appears once for each member that you selected for conversion (where xxxxxc is the specified job name). The final message is followed by three asterisks (***)

Converting

You may press Enter or any other interrupt key to return to the Converter panel.

Running the conversion job under VM

Use the Conversion Selection panel to convert one or more programs.

To run a conversion job:

1. Go to panel 1.2 to display the Conversion Selection panel, shown in Figure 15.

```
----- CCCA Conversion selection -----
Command ==>
Program source:                               Options:
Project . . . ==> CCCA                        Language level ==> *   (* 1-11)
Library . . . ==> REGTEST                     CICS . . . . . ==> Y   (Y N)
Type. . . . . ==> COBOL
Member. . . . ==>                            (Blank for member list, * for all members)
CMS file:
File ID ==>                                  If not linked, specify:
Owner's ID ==>                               Device addr. ==>     Link access mode ==>
Read password ==>                           Update password ==>
Copy libraries (MACLIBs):
DDNAME ==> SYSLIB   LIBRARY ==> CCCACOPY MACLIB J
==>                                     ==> CCCACPY2 MACLIB J
==>                                     ==> CCCACPY3 MACLIB J
==> PRIVLIB       ==> CCCAPRIV MACLIB A
==>                                     ==>
==>                                     ==>
Output source:
Program file name ==> = OUTSRC A
Copy library (MACLIB) ==> CCCAOCY MACLIB J

Date identification file:
==> PIRM01 MLESEED J

PF1 Help  PF3 Exit  PF4 Return
```

Figure 15. Conversion Selection panel (VM)

2. Enter values for:

Program source

If the program source that you want converted is within an ISPF partitioned data set, enter the data set name and the member name in the **Project**, **Library**, **Type**, and **Member** fields.

If the program source that you want converted is within a MACLIB, enter the MACLIB file name in **File ID** and the member name in the **Member** field.

If you want all members of the MACLIB or ISPF partitioned data set converted, enter an asterisk (*) instead of the member name. If you do not specify a member name or an asterisk, CCCA displays the Conversion Member Selection panel after you press Enter (see Figure 16 on page 33).

```

----- CCCA Conversion member selection -----
Command ==>

Select the member(s) to be converted and press Enter
Press PF3 to initiate conversion

NB: Members marked with NO D.I.F cannot be selected for conversion

NAME      SELECT

```

Figure 16. Conversion Member Selection panel (VM)

Place an “S” in front of all members in the list that you want converted.

Place a “C” in front of any selected member in the list to cancel the selection.

Note: If you have selected the DATE FORMAT conversion option (option 8 on the Conversion Options panel 2—see Figure 10 on page 23), and you have not specified a specific member for the date identification file on the Conversion Selection panel, the message “NO_D.I.F” appears against any member in the member selection list that does not have a corresponding member in the date identification file data set. You cannot select these members for conversion.

CMS file—File ID

If the program source that you want converted is a simple CMS file, enter the file details (fn ft fm).

Linkage fields

If you are not already linked to the minidisk where the file resides enter the appropriate details in **Owner's ID**, **Device addr**, and **Link access mode**.

Passwords

If required, enter the appropriate passwords in the **Read password** and **Update password** fields.

Options

Most options for the conversion are specified in the option panels. Two however can be set on this panel:

Language level

Overrides—for this conversion job only—the **Source language level** specified on the Language Level panel. For a list of source language level values, see “Setting source and target language levels” on page 17.

If you specify an asterisk (*), CCCA uses the value specified in the Language Level panel.

CICS

Either:

- Y The program you are submitting for conversion contains EXEC CICS commands.
- N The program does not contain EXEC CICS commands.

Copy libraries (MACLIBs)

(Only required if the program you are converting contains COPY statements.)

The copy libraries containing the old source copy members:

- The copy libraries are usually accessed through ddname SYSLIB.

Converting

The COPY statement gives the member name in the library as specified by the SYSLIB DD statement.

- The COPY statement may also indicate a specific library.

For example:

```
COPY MOD OF LIB1
```

or

```
COPY MOD IN LIB1
```

In this case, the library is accessed by specifying a ddname that defines the data set itself.

```
DDNAME ===>SYSLIB LIBRARY ===>'COBCOPY1 MACLIB A'  
DDNAME ===> LIBRARY ===>'COBCOPY2 MACLIB A'  
DDNAME ===> LIBRARY ===>'COBCOPY3 MACLIB C'  
DDNAME ===>LIB1 LIBRARY ===>'SPCLCOPY MACLIB D'  
DDNAME ===> LIBRARY ===>'SPCLCPY2 MACLIB D'
```

As the above example shows, you can concatenate up to six libraries under ddname SYSLIB or any other ddname.

Concatenation of libraries is possible for any ddname that you specify. The normal CMS rules for concatenation of libraries apply.

Note: Each input copy library that you specify must be a MACLIB, and have a filetype of MACLIB.

Output source–Program file name

(You can specify this only if the **Generate new program** field on Conversion Options panel 1 is set to Y.)

Enter the name of the file where you want CCCA to put the new source program.

The format of the Program File Name can be:

- fn ft fm
- = ft fm (the fn is taken from the input source filename or member name)
- fn MACLIB fm

If you specify an input file that is either a MACLIB or an ISPF PDS without including a member name, you must use either option *b* or *c* above when specifying the Program File Name.

Note: If the file (in option *a* above) or member that you specify already exists, CCCA replaces it.

Output source–Copy library (MACLIB)

(You can specify this only if the **Generate new copy members** field on Conversion Options panel 1 is set to Y.)

CCCA puts new source for copy members called by the source program into this library. The new copy member will have the same name as the old copy member. The copy library that you specify must be a MACLIB, and have a filetype of MACLIB.

If a member with this name already exists in the output copy library, it is not replaced unless the **Replace like-named copy members** field on Conversion Options panel 1 is set to Y.

CCCA checks that the library you specify is:

- A valid library name and that it exists
- A MACLIB
- Not the same as any of the input copy libraries
- Not the same as the input source library
- Not the same as the output source library

You can specify only one output copy library.

Date identification file

If you have selected the DATE FORMAT conversion option (option 8 on the Conversion Options panel 2—see Figure 10 on page 23), an entry field appears on the Conversion Selection panel into which you enter the file name for the date identification file.

The date identification file name can be:

- a. A simple CMS file (fn ft fm)
- b. A MACLIB
- c. An ISPF partitioned data set, with member specified
- d. An ISPF partitioned data set, with no member specified
- e. A simple CMS file with the filename specified as “=” (= ft fm)

Note:

- a. For options *b*, *d*, and *e*, the member name is taken from the input source file name, or member name.
- b. If you specify option *a* or *c*, CCCA searches that file or member for entries that match the name of the source program. For a detailed description, refer to “Date Identification file” on page 42.

3. Press Enter.

CCCA converts the member (or members) that you have selected in foreground mode. If errors are encountered during the conversion process, CCCA displays a message.

If you used the Conversion Member Selection panel to select one or more members for conversion, or specified an asterisk (*) to convert all members, CCCA displays a message indicating which member it is currently converting.

When the conversion process is complete, CCCA redisplay the Conversion Selection panel, with a message indicating the highest return code for the conversion.

Reading the Diagnostic listing

The conversion job generates a Diagnostic listing containing:

- Converted source code
- Diagnostic messages

You can tailor the contents of the Diagnostic listing using the following conversion options (for details, see “Setting conversion options” on page 19):

- **Print old source lines**
- **Print copy members**
- **Print diagnostics of level >=**

Figure 17 on page 36 shows an extract from a sample Diagnostic listing.

This sample was generated with:

- **Print old source lines** set to Y
- **Print diagnostics of level >= 0** (print all diagnostic messages)

Converting

```
1 2 3 4 5 6
000179 IF ERROR-FLAG = ZERO
000180 MOVE "TEST CASE LCPTST09 IS SUCCESSFUL." TO OUTPUT-RECORD
000181 WRITE OUTPUT-RECORD
000182 *OLD** OTHERWISE ABJ6021 00 OTHERWISE REPLACED BY ELSE
000183 ELSE
000184 MOVE "TEST CASE LCPTST09 FAILED." TO OUTPUT-RECORD
000185 WRITE OUTPUT-RECORD.
000186
000187 COPY CLOSEA.
000188+ CLOSE IN-FILE1.
000189+ CLOSE IN-FILE2.
000190+ CLOSE OUT-FILE.
000191+ CLOSE PRINT-FILE.
000192 STOP RUN.
```

Figure 17. Extract from a diagnostic listing

The columns of this report are described below.

- 1** Line ID and copy book indicator.
CCCA assigns a sequential line ID to each converted and each old source line appearing in the Diagnostic listing. Each diagnostic message appearing at the end of the listing uses the line ID to reference the line to which it refers.
The copy book indicator (“+”) appears when the line is from a copy book.
- 2** Converted program sequence numbers or old source line indicator.
For converted program source lines, if the **Resequence source lines** field on Conversion Options panel 1 was set to:
Y this column contains the new sequence numbers
N this column contains the contents of columns 1 through 6 from the old source lines
For old program source lines, this column contains *OLD**.
- 3** If column **2** contains *OLD**, this is the old source line. (Old source lines appear only if the **Print old source lines** field on Conversion Options panel 1 was set to Y.)
If column **2** does not contain *OLD**, this is the converted program source line.
- 4** Diagnostic message identifier, in the format ABJnnnn (where nnnn is a 4-digit number).
- 5** Diagnostic severity level:
 - 00** The language element has been converted into its equivalent in the target language.
 - 04** The language element has been converted, but you should inspect the change.
 - 08** Either you must, or you may have to, make a change to this language element, if you want the program to behave in the same way it did before conversion.
- 6** Diagnostic message text.

Each diagnostic message for the converted program appears twice in the Diagnostic listing:

- Alongside the source line to which it applies

- At the end of the Diagnostic listing, alongside the Line ID to which it applies

Conversion return codes

CCCA issues a return code for each converted program. This return code appears in the job log alongside the program conversion step:

- 00** CCCA did not issue any diagnostics. No changes were made to the program and no language elements were flagged for a manual change.
- 01** CCCA issued diagnostics of severity 00, but there were no diagnostics of severity greater than 00.
- 04** CCCA issued diagnostics of severity 04 and lower, but there were no diagnostics of severity greater than 04.
- 08** CCCA issued diagnostics of severity 08 and lower, but there were no diagnostics of severity greater than 08.
- 16** Required copy members were missing.
- 21** Abend occurred during conversion phase 1.
- 22** Abend occurred during conversion phase 2.
- 23** Abend occurred during conversion phase 3.

Converting

Chapter 4. DATE FORMAT Conversion Option

This chapter describes:

1. Millennium language extensions (MLE) and date fields
2. MLE terms
3. The DATE FORMAT clause
4. What you need to supply to CCCA for the DATE FORMAT conversion option
5. Selecting the DATE FORMAT conversion option
6. How the DATE FORMAT conversion option works

The DATE FORMAT conversion option is one of several options within CCCA that you can select. By selecting this option, CCCA will perform a DATE FORMAT conversion *in addition to* any other type of conversion that it may carry out (according to the source and target language levels that you have specified).

The DATE FORMAT conversion option adds a DATE FORMAT clause to selected data description entries to identify those entries as **date fields**.

The DATE FORMAT clause is part of the **millennium language extensions**.

Millennium Language Extensions (MLE) and Date Fields

Many applications use 2 digits rather than 4 digits to represent the year in date fields, and assume that these values represent years from 1900 to 1999. This compact date format works well for the 1900s, but it does not work for the year 2000 and beyond because these applications interpret “00” as 1900 rather than 2000, producing incorrect results.

The millennium language extensions are designed to allow applications that use 2-digit years to continue performing correctly in the year 2000 and beyond, with minimal modification to existing code. This is achieved using a technique known as windowing, which removes the assumption that all 2-digit year fields represent years from 1900 to 1999. Instead, windowing enables 2-digit year fields to represent years within any 100-year range, known as a **century window**.

For example, if a 2-digit year field contains the value 15, many applications would interpret the year as 1915. However, with a century window of 1960–2059, the year would be interpreted as 2015.

The millennium language extensions provide support for the most common operations on date fields: comparisons, moving and storing, incrementing and decrementing. This support is limited to date fields of certain formats; for details, see “DATE FORMAT Clause” on page 40.

For further information on MLE, see the *IBM COBOL Millennium Extensions Guide*.

Definition of terms

This book uses the following terms when referring to the millennium language extensions.

DATE FORMAT

Date Field

For the purposes of CCCA, a date field is a data item whose data description entry includes a DATE FORMAT clause.

The term date field refers to both **expanded date fields** and **windowed date fields**.

Windowed Date Field

A windowed date field is a date field that contains a **windowed year**. A windowed year consists of 2 digits, representing a year within the century window.

Expanded Date Field

An expanded date field is a date field that contains an **expanded year**. An expanded year consists of 4 digits.

The main use of expanded date fields is to provide correct results when these are used in combination with windowed date fields; for example, where migration to 4-digit year dates is not complete. If all the dates in an application use 4-digit years, there is no need to use the millennium language extensions.

Century window

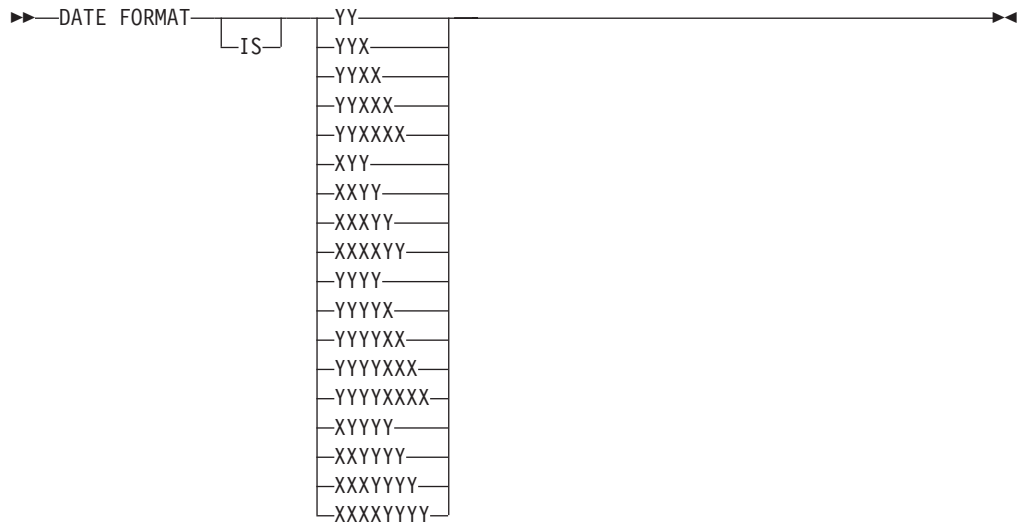
A century window is a 100-year interval within which any 2-digit year is unique. For windowed date fields, it is specified by the YEARWINDOW compiler option.

DATE FORMAT Clause

In order to indicate that a data item is a date field, the DATE FORMAT clause is used in the data description entry in the Data Division.

The DATE FORMAT clause specifies the format of the date contained in the data item.

Format



DATE FORMAT clause...

Specifies that the data item contains...

YY A windowed year.

YYX	A windowed year followed by 1 character.
YYXX	A windowed year followed by 2 characters; for example, digits representing a month (01–12).
YYXXX	A windowed year followed by 3 characters; for example, digits representing a day of the year (001–365).
YYXXXX	A windowed year followed by 4 characters; for example, 2 digits representing a month and 2 digits representing a day of the month.
XYY	A windowed year preceded by 1 character.
XXYY	A windowed year preceded by 2 characters.
XXXYY	A windowed year preceded by 3 characters.
XXXXYY	A windowed year preceded by 4 characters.
YYYY	An expanded year.
YYYYX	An expanded year followed by 1 character.
YYYYXX	An expanded year followed by 2 characters.
YYYYXXX	An expanded year followed by 3 characters.
YYYYXXXX	An expanded year followed by 4 characters.
XYYYY	An expanded year preceded by 1 character.
XXYYYY	An expanded year preceded by 2 characters.
XXXYYYY	An expanded year preceded by 3 characters.
XXXXYYYY	An expanded year preceded by 4 characters.

Examples

77 YEAR1	PIC 99 DATE FORMAT YY.
77 DATEA	PIC 9(5) DATE FORMAT YYXXX.
77 DATEB	PIC 9(4) DATE FORMAT XXYY.
77 DATEC	PIC 9(7) DATE FORMAT XXXYYYY.
77 DATED	PIC 9(8) DATE FORMAT YYYYXXXX.

What you need to supply to CCCA

CCCA does not, itself, identify which data items within a COBOL program are used to contain dates. Instead, CCCA requires the names (and format) of each of these data items to be supplied as additional input. Typically, this information is supplied by a Year 2000 tool.

The DATE FORMAT conversion option within CCCA requires:

DATE FORMAT

- The COBOL source program that is to be converted
- A **date identification file** that identifies each data item in that COBOL source program that is used to contain a date. The date identification file contains the *program name* followed by details for each such data item:
 - The *line number* of the data item (used only as a delimiter by CCCA)
 - The *format* of the data item
 - The *name* of the data item, qualified as necessary; see “Qualification of data names” on page 45

Note: Details of data items for more than one program can be held in the same date identification file. For more information, see “Format.”

Date Identification file

The purpose of the date identification file is to identify which data items in the COBOL program and copy members to be converted are used to contain dates so that CCCA can add an appropriate DATE FORMAT clause to the corresponding data description entries.

It is your responsibility to create the date identification file. You must use the format as described in this document, and supply the file to CCCA.

The method used to produce the date identification file does not matter. It could be, for example, that you choose to create the file manually, inserting the details of data items in the program that is to be converted that you know are used to contain dates. However, it is much more likely that you will use one of the Year 2000 tools that can generate a date identification file for you.

In either case, it is essential that you carefully check the contents of the date identification file for completeness and accuracy *before* supplying the date identification file to CCCA for the actual program conversion.

CCCA performs some syntax checking before adding a DATE FORMAT clause to a data description entry (see “Checking DATE FORMAT Clause syntax” on page 46). However, CCCA cannot check which data items are used to contain dates. The onus is therefore on you to ensure that the date identification file correctly identifies all such data items.

Format

The information in the date identification file relates to each data item, within a specific program, that has been identified (by some external means) as containing a date.

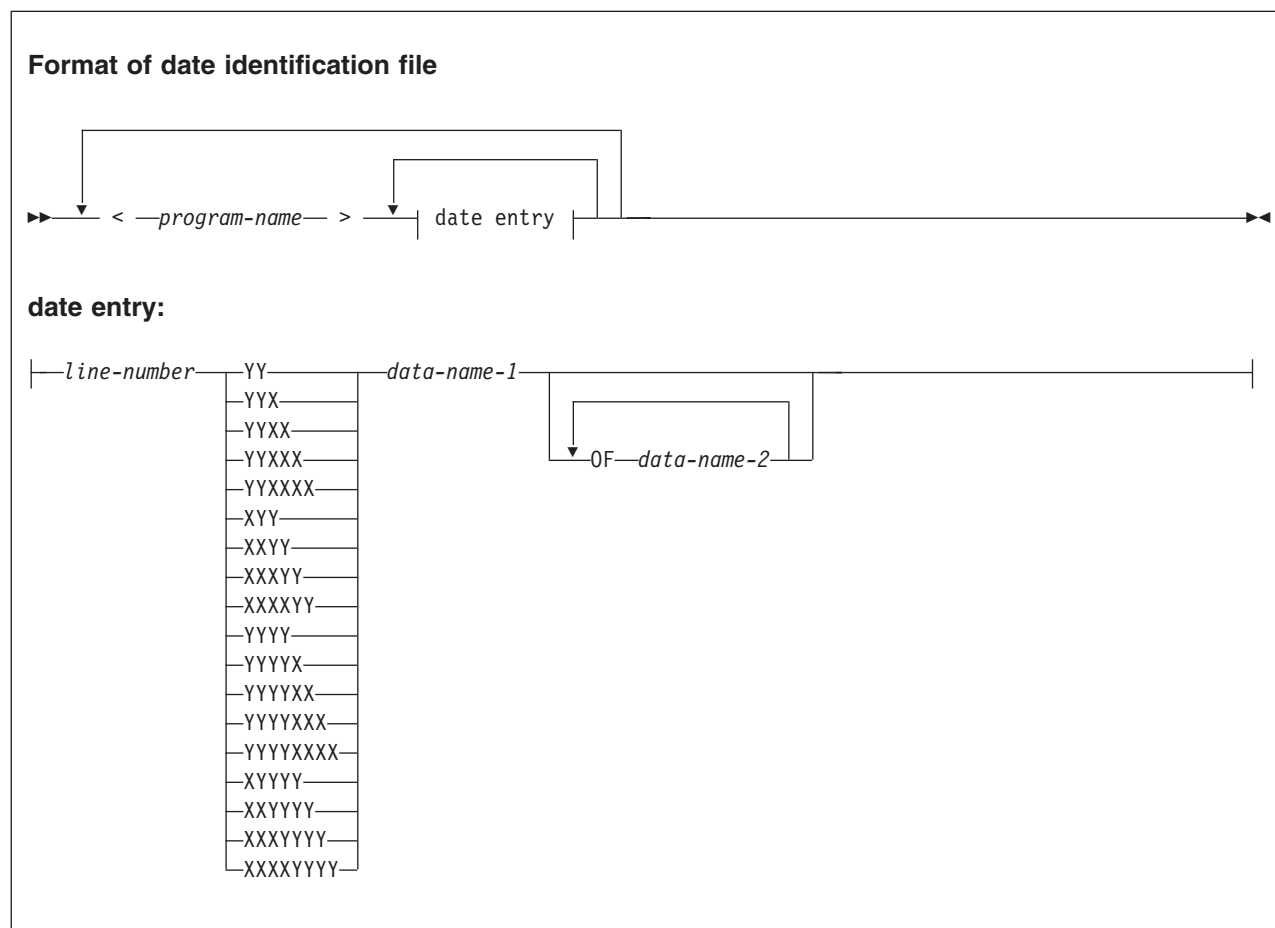
The date identification file consists of 80-byte records containing data in a free-format style. Each record may contain one or more fields. Each field within a record is separated by one or more spaces.

Note:

1. While the date identification file is free-format, you will find it far more readable, and therefore much easier to reference, if a formatted style is used. (Refer to “Examples of date identification file contents” on page 45.)
2. The date identification file can contain the Double-Byte Character Set (DBCS).

The details for each data item are “grouped” by program name, allowing the same date identification file to be used for more than one program. The program to

which each group of data item details relate is identified by means of the *program name* preceding the group.



Note:

1. Line breaks are ignored (except for comments, in which case each line must have an "*" in column 1—see "Comment lines" on page 45).
2. Each item may start in any column and must be separated from the previous item by at least 1 space.

program-name

The name of the program to which the data item information (that follows) applies. The *program name* can be a maximum of 30 characters and must be enclosed (with no intervening spaces) in "<" and ">" symbols. For example, <ABJIVP01>

line-number

A numeric field that *can* be used as a reference to the line number in the source COBOL program where the data item appears. For example, 223

Note: CCCA uses this field as a delimiter only and does not use its actual value. You can use this field as a useful reference when checking the content of the date identification file against the source program before you input the date identification file to CCCA. However, even if you are not using the field for this purpose, you must still place *some* numeric value ("1", for example) in this position.

DATE FORMAT

YY Specifies the data item contains a windowed year.

YYX

Specifies the data item contains a windowed year followed by 1 character.

YYXX

Specifies the data item contains a windowed year followed by 2 characters; for example, digits representing a month (01–12).

YYXXX

Specifies the data item contains a windowed year followed by 3 characters; for example, digits representing a day of the year (001–365).

YYXXXX

Specifies the data item contains a windowed year followed by 4 characters; for example, 2 digits representing a month and 2 digits representing a day of the month.

XYY

Specifies the data item contains a windowed year preceded by 1 character.

XXYY

Specifies the data item contains a windowed year preceded by 2 characters.

XXXYY

Specifies the data item contains a windowed year preceded by 3 characters.

XXXXYY

Specifies the data item contains a windowed year preceded by 4 characters.

YYYY

Specifies the data item contains an expanded year.

YYYYX

Specifies the data item contains an expanded year followed by 1 character.

YYYYXX

Specifies the data item contains an expanded year followed by 2 characters.

YYYYXXX

Specifies the data item contains an expanded year followed by 3 characters.

YYYYXXXX

Specifies the data item contains an expanded year followed by 4 characters.

XYYYY

Specifies the data item contains an expanded year preceded by 1 character.

XXYYYY

Specifies the data item contains an expanded year preceded by 2 characters.

XXXYYYY

Specifies the data item contains an expanded year preceded by 3 characters.

XXXXYYYY

Specifies the data item contains an expanded year preceded by 4 characters.

Note:

1. MLE does not provide support for any forms of date other than those specified above.
2. MLE does not perform any special processing for any parts of dates except for the year part. Other forms of date that have the same general form as the explicitly supported dates will be treated in the same way. For instance,

MLE regards year and week dates of the form YYWW as if they were year and month dates of the form YYMM (represented by the *date format* YYXX).

data-name-1

The lowest-level name associated with the data item used to hold a date. For example, DATE-1

data-name-2

A qualifier which is a higher-level name that helps to uniquely identify *data-name-1*. For example, A-RECORD

Qualification of data names: The syntax for qualifying names within normal COBOL source code allows either the word "IN" or the word "OF" to be used between the lower-level data name and the higher-level data name. A detailed description of qualification can be found in the *IBM COBOL Language Reference* for your platform.

However, only "OF" is acceptable in the case of qualified names in the date identification file input to CCCA.

Comment lines

Comment lines can be included in the date identification file. They are identified by having an "*" in column 1 of the record. Comment lines are ignored by CCCA.

Format of date identification file comment line

(1)
 ▶ *—————*comment-text*—————▶

Notes:

- "*" must be in column 1.

Examples of date identification file contents**Example 1** (Recommended formatted style)

```
*      STUDENT FILE PRODUCED 04/16/98
<STUDPRG1>
127    YYXXXX    BIRTH-DATE
157    YY        ENROL-YEAR
162    YYXX      GRAD-MONTH
195    YYXXXX    FEE-DUE-DATE OF CUR-SEMEST OF SUBJECT-CODE OF
                COLLEGE-NUM OF STATE-CODE

<STUDPRG2>
96     YYXXXX    ARREARS-DATE OF ARR-1
98     YYXXXX    ARREARS-DATE OF ARR-2
100    YYXXXX    ARREARS-DATE OF ARR-3

<STUDPRG3>
388    YYXX      PAID-DATE
```

Example 2

```
<VETSYS01>
1
YYXX
REG-DATE
1
YYXX
NEXT-INNOC-DATE
1
YYXX
LAST-INNOC-DATE
```

DATE FORMAT

```
* REMINDER DATE  
1  
YYXXX  
REMIND-DATE
```

Example 3

```
<ACCT1> 1 YYXXXX LOAN-DATE 1 YY VAL-YEAR 1 YYXX DUE-MONTH 1  
YYXXXX REPAY-DATE OF CUR-PERIOD <ACCT2> 1 YYXXXX ARREARS-DATE OF  
ARR-1 1 YYXXXX ARREARS-DATE OF ARR-2 1 YYXXXX ARREARS-DATE OF  
ARR-3 <ACCT3> 1 YYXXX PEN-DATE
```

Selecting the DATE FORMAT Conversion Option

To select the DATE FORMAT conversion option, specify Y for option 8 (**Add DATE FORMAT clause to date fields**) on the Conversion Options 2 panel (see “Setting conversion options” on page 19).

When you select this option, an additional field appears on the Conversion Selection panel (MVS)— see Figure 12 on page 28, or the Conversion Selection panel (VM)— see Figure 15 on page 32, into which you enter the name of the *date identification file*.

Note: You can only select the DATE FORMAT conversion option if the target language level supports the DATE FORMAT clause. For details, see Table 3 on page 18.

How the DATE FORMAT Conversion Option works

If you have selected the DATE FORMAT conversion option, CCCA scans the date identification file for the name of the program being converted.

When the program name is found:

1. CCCA reads the data item details in the date identification file that follow the program name and stores them in an internal table.
2. CCCA checks each data item in the Data Division of the program being converted to determine if its name is in the internal table.
3. If the name is in the internal table, CCCA performs various syntax checking (see “Checking DATE FORMAT Clause syntax”) to determine if a DATE FORMAT clause is allowed for the data description entry.
4. If no syntax violations are found, CCCA adds a DATE FORMAT clause using the date format specified for that data item in the internal table.

Checking DATE FORMAT Clause syntax

Before adding the DATE FORMAT clause, CCCA checks that the addition of the clause does not violate the following syntax rules.

The DATE FORMAT clause can only be specified for a data description entry which:

- Does not already contain a DATE FORMAT clause
- Does not have a:
 - BLANK WHEN ZERO clause
 - JUSTIFIED clause
 - SIGN clause with a SEPARATE CHARACTER phrase
- Has a level number other than 66 or 88

- In the case of an elementary data item:
 - Has a PICTURE string that contains:
 - All 9's
 - An S followed by 9's
 - 9's, A's, and X's only, and not all A's
 - Has a computer storage format (USAGE clause) of DISPLAY, COMPUTATIONAL-3, PACKED-DECIMAL, BINARY, COMPUTATIONAL, or COMPUTATIONAL-4
 - Where the length of the PICTURE clause (999999, for example) matches the length of the corresponding format field in the date identification file (YYXXXX, for example)
- In the case of a group data item:
 - Contains a USAGE clause of DISPLAY
- Is not an external data item or part of an external data item

If any of the above syntax rules are violated, CCCA issues a diagnostic message stating the reason the DATE FORMAT clause was not added. Otherwise, CCCA adds the DATE FORMAT clause to the data description entry.

Note:

1. The above is not a comprehensive list of the DATE FORMAT clause syntax rules.
2. It is possible that CCCA may add the DATE FORMAT clause where it is not allowed. In these cases, the post-conversion compile, if specified, will identify the error.

If you select both the DATE FORMAT conversion option and the **Compile after converting** option (see Figure 10 on page 23), CCCA compiles the converted program with the compiler option DATEPROC(FLAG) and the installation default value of the YEARWINDOW option.

The diagnostics in the resultant compiler listing will indicate whether manual changes to the program are required.

Chapter 5. Conversion reports and the conversion log

This chapter describes how to:

- Generate conversion reports
- Browse, update, and erase the conversion log

Generating conversion reports

Conversion reports list program conversion statistics.

To see the types of conversion report you can generate, go to panel 1 (the Converter Menu, shown in Figure 5 on page 13).

The Converter Menu contains options for generating conversion reports:

Report	Lists details of...	Sorted by
Program/File	Converted programs, and the files they use	Program name
File/Program	As above (with fewer program details)	File name
Copy/Program	Copy members used by converted programs	Copy member name
Program/Copy	As above	Program name
Call/Program	CALL statements in converted programs	CALL subroutine identifier or subroutine literal
Program/Call	As above	Program name

Conversion reports list details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58).

When there's nothing to report...

With the exception of the Program/File report, CCCA can only produce any of the conversion reports that you request if there are matching records in the CCCA Control file.

For example, if you used CCCA to convert a number of programs none of which contained references to any file names, and you then requested CCCA to generate a File/Program report (option 4 on the Converter Menu, see Figure 5 on page 13), CCCA would be unable to produce the requested report.

In this situation (when no matching records for a requested report exist in the CCCA Control file), CCCA displays one of the following messages in the top right-hand corner of the screen:

- Report not generated
- Nothing to Report on

Appendix H, “Sample output,” on page 199 contains sample report listings.

The following sections describe each conversion report in detail.

Program/File report

The Program/File report lists details of converted programs:

- Date and time the program was last converted
- Options specified for the conversion
- Conversion statistics
- Converted program status
- Details of files used by the program

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58).

Figure 18 shows a sample Program/File report.

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE 04/15/98 17:36:40 Page 1

..... PROGRAM -- FILE REPORT

1	2	3	4	5	6	7		13
PGM.NAME	REV	PBR	SUFF	DEL	LANG	MEMBER	STATUS	FILES DEFINED
CONV	WORD	L	S	12345	67890	12345	DATE/TIME	OLD NEW CNV SYSTEM COBOL
ORG	ORG	REQ	NAME	NAME	NAME	NAME		NAME
ABJIVP01	01	213	0	Q 2	N	YYYY YNNNN NNNNN ABJIVP01	COMPLETE 98/04/15 16:24 COMPILE RC=00 98/04/15 16:25 MANUAL COMPLETION	S S N DDPRINT PRINT-FILE
ABJIVP02	01	208	2	Q 2	N	YYYY YNNNN NNNNN ABJIVP02	COMPLETE 98/04/15 16:25 COMPILE RC=00 98/04/15 16:26 MANUAL COMPLETION	S S N PRINT PRINT-OUT

Figure 18. Program/File report

The columns of this report are described below.

- 1** The name of the converted program, specified in the Identification Division PROGRAM-ID paragraph.
- 2** The number of times you have converted the program.
- 3** The number of Language Conversion Programs (LCPs) invoked during program conversion.
- 4** The number of user-defined words in the program to which CCCA appended suffixes.
- 5**
 - DEL** Literal delimiter used in the program:
 - A** Apostrophe (')
 - Q** Quotation mark (")
 - LVL** Source language level used for the conversion, as specified on the Language Level panel (Figure 8 on page 17) or the
 - (MVS only) Conversion (Selection) panel (see Figure 12 on page 28):
 - (VM only) Conversion Selection panel (see Figure 15 on page 32):
 - 1** DOS/VS COBOL—LANGLVL(1)
 - 2** DOS/VS COBOL—LANGLVL(2)

- 3 OS/VS COBOL—LANGLVL(1)
- 4 OS/VS COBOL—LANGLVL(2)
- 5 VS COBOL II Release 1.0, Release 1.1, or Release 2.0 (or any COBOL with the CMPR2 option)
- 6 VS COBOL II—NOCMPR2 Release 3.0, Release 3.1, or Release 3.2
- 7 VS COBOL II—NOCMPR2 Release 4.0
- 8 COBOL/370—NOCMPR2
- 9 COBOL for VSE/ESA—NOCMPR2
- 10 COBOL for MVS & VM—NOCMPR2
- 11 COBOL for OS/390 & VM—NOCMPR2
- 12 Enterprise COBOL (prior to Version 5)

CICS CICS processing option used for the conversion, as specified on the

- (MVS only) Conversion Selection panel (see Figure 12 on page 28).
- (VM only) Conversion Selection panel (see Figure 15 on page 32).

You should have set this option to:

- Y** If the program you submitted for conversion contained EXEC CICS statements
- N** If the program had no EXEC CICS commands

6 Options CCCA used to convert the program, as specified on Conversion Options panel 2 (Figure 10 on page 23).

For a description of these options, see “Setting conversion options” on page 19.

7

Under MVS

Member name of the program (if the old source program was in a partitioned data set).

CCCA uses the same name for the new source member (if it is generated).

Under VM

The *fn* of the program, or the member name if the old source program was in a CMS MACLIB or ISPF partitioned data set.

CCCA uses the same name for the new source file (*fn*) or source member (if it is generated).

8

Status of the converted program:

NOCHANGE

The last conversion of this program received return code 00. CCCA made no changes to the program. No manual changes to the program are required.

COMPLETE

The last conversion of this program received return code 01. The program has been completely converted. No manual changes to the program are required.

WARNING

The last conversion of this program received return code 04. The program has been converted. The program may compile and execute successfully, but you should inspect the converted language elements that received level 04 diagnostics.

Conversion reports

ERROR

The last conversion of this program received return code 08. CCCA issued level 08 diagnostics, indicating you may need to manually convert these program elements.

ABEND

The last attempted conversion of this program abnormally terminated:

ABEND-002

Abend occurred in conversion phase 2

ABEND-003

Abend occurred in conversion phase 3

- 9** Date and time this program was last converted by CCCA.
- 10** Return code of the post-conversion compile (shown only if the program was compiled after its last conversion).

A program is compiled after conversion if these conditions are met:

- The **Compile after converting** field on Conversion Options panel 2 (Figure 10 on page 23) is set to Y
- The status of the converted program is NOCHANGE, COMPLETE, or WARNING

Note: If these conditions are met, and the **CICS** field on the Conversion panel is set to Y, the new source code is translated by the CICS command language translator before compilation.

- 11** Date and time of the last post-conversion compile of the new source program.
- 12** Date and time you completed manual changes to the new source program. You enter this information on the Conversion Log panel (see “Browsing or updating the conversion log” on page 57).
- 13** For each file that the program uses, the report lists:

Old Org

Organization of the file before conversion:

- A Actual track addressing
- D Direct organization
- I Indexed organization
- R Relative organization
- S Standard sequential organization
- U Actual track addressing (REWRITE)
- W Direct organization (REWRITE)

New Org

Organization that the file requires after the program is converted:

- I VSAM Indexed organization
- R VSAM Relative organization
- S Sequential organization

Cnv Req

Does the file require conversion?

- N You will not have to convert the file.
- Y You will have to convert the file.

System Name

System name (ddname) of the file, as specified in the ASSIGN clause of the COBOL program.

Note: If this system name is not used consistently at your installation, it may be associated with different data sets.

COBOL Name

The name used for the file in the COBOL program, as specified in the SELECT statement.

File/Program report

The File/Program report lists details of files used by converted programs:

- System name (ddname) of the file
- COBOL name of the file
- File organization required by the converted program
- Whether or not you will have to convert the file

This report is sorted by the system name of the files.

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58). Also, see “When there’s nothing to report...” on page 49.

Use this report for planning file conversions.

Figure 19 shows a sample File/Program report.

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE 04/15/98 17:49:47 Page 1

```

.....  F I L E  --  P R O G R A M  R E P O R T  .....
  1      2      3      4      5
SYSTEM  PROGRAM  ORG  CONVERSION  COBOL
NAME    NAME    REQUIRED  NAME

EIPARM  EI030BPF  I    YES    EIPARM
IBDAM   LCPI0105      R    YES    BDAM-IN
        LCPI0107      R    YES    BDAM-IN
    
```

Figure 19. File/Program report

The columns of this report are described below.

- 1** System name (ddname) of the file, as specified in the ASSIGN clause of the COBOL program.

Note: If this system name is not used consistently at your installation, it may be associated with different data sets.

- 2** The names of the converted programs (as specified in the PROGRAM-ID paragraph of the Identification Division) that use the file with the given system name.

- 3** Organization required for the file after the program is converted:
 - I** VSAM Indexed organization
 - R** VSAM Relative
 - S** Sequential

- 4** Does the file require conversion?
 - NO** You will not have to convert the file.
 - YES** You will have to convert the file.

Conversion reports

- 5 The name used for the file in the COBOL program, as specified in the SELECT statement.

Copy/Program report

The Copy/Program report lists details of copy members used by converted programs:

- Programs that use the copy member
- For each program:
 - The section of the program into which the member is copied
 - The associated name in the COPY statement (if it exists)

This report is sorted by copy member name.

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58). Also, see “When there’s nothing to report...” on page 49.

Figure 20 shows a sample Copy/Program report.

```
5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE    04/15/98 17:50:27    Page 1

..... C O P Y  -- P R O G R A M  R E P O R T  .....
 1      2      3      4
COPY   PROGRAM  LOCATION  ASSOCIATED
NAME   NAME     WORKING-STORAGE DATA-PARAM-CARD
ALTPCB AMPM2AA  ALT-IO-PCB
CPNMA  CPGM1501  WORKING-STORAGE DATA-PARAM-CARD
CPNMD  CPGM1501  WORKING-STORAGE DATA-SEL-HEADER
```

Figure 20. Copy/Program report

The columns of this report are described below.

- 1 The name of the copy member.
- 2 The names of the programs that use this copy member.
- 3 Section of the COBOL program into which the member is copied. This is one of the following:
 - Environment Division
 - File Section
 - Identification Division
 - Input-Output Section
 - Linkage Section
 - Procedure Division
 - Report Section
 - Working-Storage Section
- 4 Associated name in the COPY statement (if it exists).

COBOL 68 Standard language allows the COPY statement with an associated name. For example:

```
01 INPUT-RECORD COPY RDIN2.
```

(where INPUT-RECORD is the associated name)

Program/Copy report

The Program/Copy report lists details of copy members used by converted programs:

- Copy members each program uses
- For each copy member:
 - The section of the program into which the member is copied
 - The associated name in the COPY statement (if it exists)

This report is sorted by program name.

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58). Also, see “When there’s nothing to report...” on page 49.

Figure 21 shows a sample Program/Copy report.

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE      04/15/98 17:53:02  Page 1

..... P R O G R A M - - C O P Y   R E P O R T .....
 1 2 3 4
PROGRAM COPY LOCATION ASSOCIATED
NAME NAME
LCPTST09 L090PT1 FILE SECTION OUTPUT-RECORD
          L090PT2 FILE SECTION
          L090PT3 WORKING-STORAGE NUM-OF-ITEMS
          L090PT3A WORKING-STORAGE
          L090PT4 WORKING-STORAGE
LCPTST20 L200PT1 WORKING-STORAGE
..... E N D O F R E P O R T .....
    
```

Figure 21. Program/Copy report

The columns of this report are described below.

- 1** The name of the program, as specified in the PROGRAM-ID paragraph of the Identification Division.
- 2** The names of the copy members used in this program.
- 3** Section of the COBOL program into which the member is copied. This is one of the following:
 - Environment Division
 - File Section
 - Identification Division
 - Input-Output Section
 - Linkage Section
 - Procedure Division
 - Report Section
 - Working-Storage Section
- 4** Associated name in the COPY statement (if it exists).

COBOL 68 Standard language allows the COPY statement with an associated name. For example:

```
01 INPUT-RECORD COPY RDIN2.
```

(where INPUT-RECORD is the associated name)

Call/Program report

The Call/Program report lists CALL statements in converted programs.

This report is sorted by CALL statement subroutine identifier or subroutine literal.

Conversion reports

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58). Also, see “When there’s nothing to report...” on page 49.

Figure 22 shows a sample Call/Program report.

```
5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE    04/15/98 17:55:50  Page 1
.....  C A L L  --  P R O G R A M  R E P O R T  .....
1      2      3
PROGRAM  NO OF  CALL
NAME     CALLS  NAME

BLGA201  00005  'CBLBTS'
BLGF200  00001
AMPM2AA  00010  'CBLTDLI'
FCCMENU  00012
MENU     00039
RDT01    00013
RDT02    00014
RDT03    00014

.....          E N D  O F  R E P O R T  .....
```

Figure 22. Call/Program report

The columns of this report are described below.

- 1** The name of the program that contains the CALL 'name' statement.
- 2** The number of CALL 'name' statements in the program.
- 3** CALL statement subroutine identifier or subroutine literal.

Program/Call report

The Program/Call report lists CALL statements in converted programs.

This report is sorted by program name.

Note: This report lists details only for programs converted since you last erased the conversion log (see “Erasing the conversion log” on page 58). Also, see “When there’s nothing to report...” on page 49.

Figure 23 shows a sample Program/Call report.

```
5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE    04/15/98 17:59:32  Page 1
.....  P R O G R A M  --  C A L L  R E P O R T  .....
1      2      3
PROGRAM  NO OF  CALL
NAME     CALLS  NAME

AMPM2AA  00010  'CBLTDLI'
          00001  'CSERR'
          00001  'MPM202'

BLGA201  00005  'CBLBTS'
          00002  'DATMAN'
          00003  'DISTHD'
          00004  'LDCALL'
          00026  'PROGMS'
```

Figure 23. Program/Call report

The columns of this report are described below.

- 1** The name of the program that contains the CALL 'name' statement.
- 2** The number of CALL 'name' statements in the program.
- 3** CALL statement subroutine identifier or subroutine literal.

Using the conversion log

CCCA records program conversion statistics in the conversion log.

CCCA uses these statistics to generate the conversion reports.

The conversion log is stored in your Control file.

To browse a summary of the conversion log, and update statistics of manual conversions, see “Browsing or updating the conversion log.”

To erase the conversion log, see “Erasing the conversion log” on page 58.

Browsing or updating the conversion log

To browse or update the conversion log, go to panel 1.L (the Conversion Log panel, shown in Figure 24).

```

----- CCCA Conversion Log ----- Row 1 to 2 of 2
Command ==>                               SCROLL ==> HALF

Enter manual completion details

PF1 Help  PF3 Exit  PF4 Return  PF7 Up  PF8 Down  ENTER Save details

1      2      3      4
Program name  Status  Date    Time    YY/MM/DD  HH:MM  Comments
PIR001      COMPLETE  98/04/16  19:18   / /      :
PIR002      COMPLETE  98/04/15  15:48   / /      :
PIR003      COMPLETE  98/04/15  17:01   / /      :
PIR003B     WARNING   98/04/16  12:07   / /      :
PIR004      COMPLETE  98/04/15  18:20   / /      :
PIR007      COMPLETE  98/04/20  14:09   / /      :
PIR008      MAN. COMP 98/04/19  14:08   / /      :

***** Bottom of data *****

```

Figure 24. Conversion Log panel

For each program converted since you last erased the conversion log, this panel lists:

- Status of the program after it was last converted by CCCA
- Date and time the program was last converted by CCCA

If you have to manually convert a program, this panel allows you to record:

- Date and time you completed manual conversion
- Comments about the conversion

When you have updated these details, press Enter to save them.

Conversion log

Note: Enter information on this panel only if you are using the log to keep track of manual conversion effort. CCCA does not use the information you enter on this panel. (The date and time you enter appear on the Program/File report, under the heading “Manual completion”.)

To scroll through the conversion log, use PF7 and PF8.

The columns of this panel are described below.

- 1** The name of the converted COBOL program.
- 2** Status of the program after it was last converted by CCCA:
 - NOCHANGE**
CCCA made no changes to the program. No manual changes to the program are required.
 - COMPLETE**
The program has been completely converted. No manual changes to the program are required.
 - WARNING**
The program has been converted. It may compile and execute successfully, but you should inspect the converted language elements that received level 04 diagnostics.
 - MAN. COMP**
Manual changes to the program may be required. Check the language elements that received level 08 diagnostics.
 - ABEND**
The last attempted conversion of this program abnormally terminated:
 - ABEND-002**
Abend occurred in conversion phase 2
 - ABEND-003**
Abend occurred in conversion phase 3
- 3** Date and time the program was last converted by CCCA.
- 4** Enter:
 - Date and time you completed manual changes to the program
 - Comments you want to make about the conversion of this program

Erasing the conversion log

Erasing the conversion log deletes the program conversion statistics.

You should erase the conversion log when it becomes too large or when you have converted an application, and you are no longer interested in the conversion statistics.

Attention (MVS only)

Do not erase the conversion log while you are running a batch conversion. The conversion results may be unpredictable.

To erase the conversion log:

1. Go to panel **1.E** (the Confirm Erase Log panel, shown in Figure 25 on page 59).

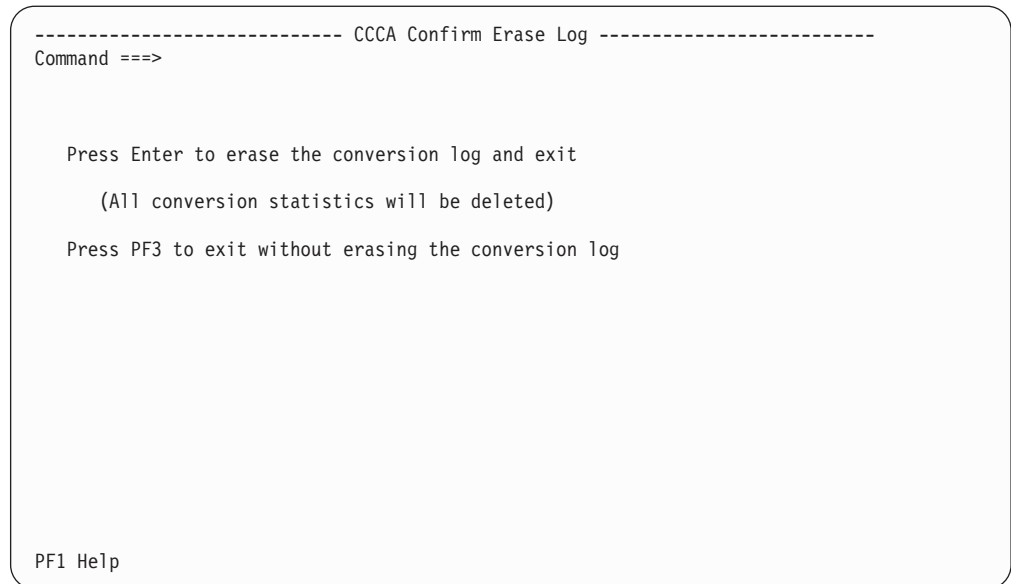


Figure 25. Confirm Erase Log panel

2. To erase the conversion log and exit the panel, press Enter.
or
To exit the panel without erasing the log, press PF3.

Conversion log

Chapter 6. Customizing CCCA

This chapter describes how to:

- Customize CCCA
- Update the COBOL Reserved Word data set
- Compile LCPs
- Delete LCPs from the LCP library
- Activate and deactivate debugging for each LCP
- Print a directory of the LCP library
- Update messages

You can use CCCA as supplied to convert your COBOL programs.

However, if you want to:

- Convert, flag, or remove additional (possibly non-COBOL) language elements
- Change how CCCA converts particular language elements

then you need to customize CCCA by:

- Modifying the supplied Language Conversion Programs (LCPs)
- Writing new LCPs

An LCP is a COBOL-like program that converts one or more COBOL language elements.

For a list of supplied LCPs, see Appendix G, “LCP directory,” on page 191.

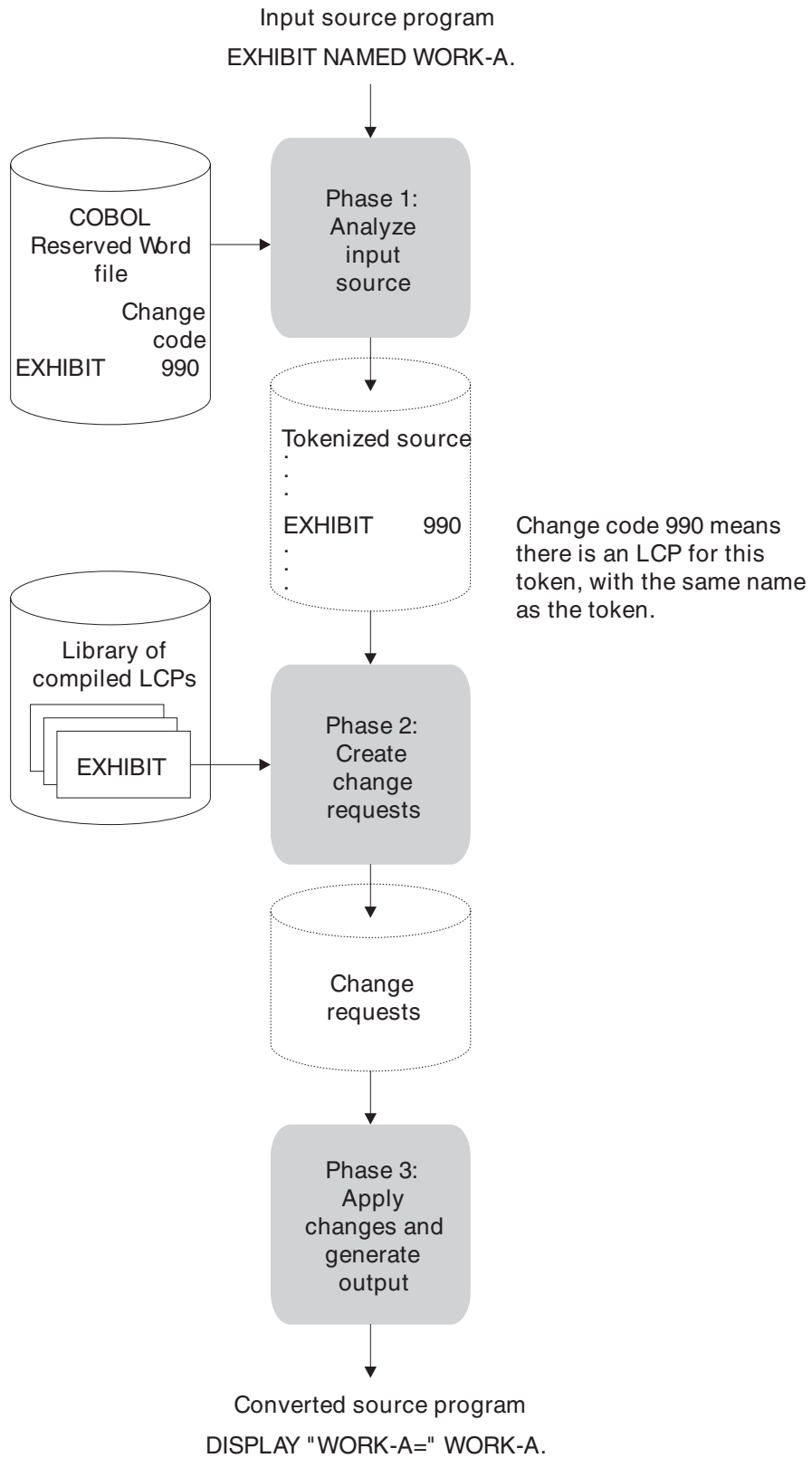


Figure 26. How CCCA invokes LCPs

How CCCA invokes LCPs

Before customizing CCCA, you need to understand how CCCA invokes LCPs during conversion.

This section assumes you have already read the introductory section “How CCCA works” on page 4.

Figure 26 on page 62 shows how CCCA uses the tokenized source and the *COBOL Reserved Word data set* to determine which LCPs to invoke during conversion.

In conversion phase 1, CCCA reads the input source program and creates a token record for each:

- COBOL word
- Literal
- Picture character-string
- Separator
- Line of the following comment paragraphs in the Identification Division:
 - AUTHOR
 - INSTALLATION
 - DATE-WRITTEN
 - DATE-COMPILED
 - SECURITY
 - REMARKS (DOS/VS COBOL and OS/VS COBOL only)

Comment lines, and the following compiler directives, are *not* tokenized:

- SKIP1
- SKIP2
- SKIP3
- EJECT
- TITLE
- *CBL
- *CONTROL

CCCA checks whether each COBOL word in the input source program is in the COBOL Reserved Word data set or not.

The COBOL Reserved Word data set lists words that may invoke LCPs, and specifies:

Word type

Where a word can occur in a COBOL program

Change code

The LCP (if any) to invoke when a word occurs

If CCCA finds the word in the COBOL Reserved Word data set, it adds the word type and change code to the token record.

In conversion phase 2, CCCA reads the tokenized source. When a token record is encountered that has something other than change code 999, CCCA invokes the LCP that is indicated by the code.

The invoked LCPs generate detailed change requests for converting the input source program.

In conversion phase 3, CCCA applies the change requests to the input source program.

Customizing the way CCCA converts a language element

If CCCA already converts a language element, but you want to customize the way it is converted:

1. Determine which LCP converts the language element

Determine the word in the language element that invokes the LCP.

The word will be on the Reserved Words panel (2.1, shown in Figure 27 on page 66) with a change code other than 999.

If the word's **Change code** is

990 This word invokes an LCP that has the word in its CONVER statement.

This LCP is not invoked by any other words.

The LCP source member name is the **Reserved word name**, or an abbreviation.

nnn (Other than 990 and 999) this word invokes an LCP that has LCP-*nnn* in its CONVER statement.

This LCP may be invoked by other words.

The LCP source member name is LCP*nnn*.

Examine the LCP source member to confirm that this is the LCP that converts this language element.

If you are not sure, delete the LCP from the LCP library, and see if CCCA still converts the language element (see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71). If it does, this isn't the LCP you're after.

Replace the LCP by compiling it (see “Compiling LCPs under MVS” on page 68 or “Compiling LCPs under VM” on page 70), then continue looking for the correct LCP.

If you are replacing an existing LCP which has a change code of 990 (invoked by word), then delete the old LCP (see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71) before updating the reserved word table and recompiling the new LCP.

2. Edit the LCP source

Update the LCP source member to convert the language element as required.

For details, see Chapter 7, “Developing Language Conversion Programs,” on page 77.

3. Compile the LCP

For details, see “Compiling LCPs under MVS” on page 68 or “Compiling LCPs under VM” on page 70.

4. Test the LCP

Convert sample programs containing the language element and all its variants.

To activate debugging for the LCP, see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71.

Customizing CCCA to convert an additional language element

To convert a language element not currently converted by CCCA:

1. Choose the word in the language element that will invoke the LCP

If there is more than one candidate for this word, then choose the word that occurs least often in other language elements. (You should try to minimize the number of times an LCP is called unnecessarily.)

2. Determine whether the word already invokes an LCP

Go to panel 2.1 (the Reserved Words panel, shown in Figure 27 on page 66).

A word that satisfies the following conditions already invokes an LCP:

- Appears in the reserved word list.
- Has any **Change code** except 999.

Otherwise, the word does not invoke an LCP.

3. If the word already invokes an LCP

If the word's **Change code** is

990 this word invokes an LCP that has the word in its CONVER statement.

This LCP is not invoked by any other words.

The LCP source member name is the **Reserved word name**, or an abbreviation.

a. Update the word's **Change code** to 999.

If necessary, update the **Word type**.

For details, see "Updating the COBOL reserved word Data Set" on page 66.

b. Edit the LCP source to convert the language element.

For details, see Chapter 7, "Developing Language Conversion Programs," on page 77.

nnn (other than 999 and 990) this word invokes an LCP that has LCP-*nnn* in its CONVER statement.

This LCP may be invoked by other words.

The LCP source member name is LCP*nnn*.

a. Update the word's **Change code** to 999.

If necessary, update the **Word type**.

For details, see "Updating the COBOL reserved word Data Set" on page 66.

b. Copy the code from the existing LCP source member to a new member.

c. In the new LCP source, change the LCP-*nnn* in the CONVER statement to the reserved word.

d. Edit the new LCP source to convert the language element.

For details, see Chapter 7, "Developing Language Conversion Programs," on page 77.

If the word does not invoke an LCP

a. If the word does not appear in the Reserved Word list, add the word to the list. Specify a **Change code** of 999.

b. If necessary, update the **Word type**.

For details, see "Updating the COBOL reserved word Data Set" on page 66.

c. Write a new LCP to convert the language element.

For details, see Chapter 7, "Developing Language Conversion Programs," on page 77.

4. Compile the LCP

For details, see "Compiling LCPs under MVS" on page 68 or "Compiling LCPs under VM" on page 70.

5. Test the LCP

Convert sample programs containing the language element and all its variants.
 To activate debugging for the LCP, see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71.

Updating the COBOL reserved word Data Set

As supplied, the COBOL Reserved Word file contains a record for each reserved word in the source language levels.

To browse or update the Reserved Word data set, go to panel 2.1 (the Reserved Words Update panel, shown in Figure 27).

```

----- CCCA COBOL Reserved Words -----
Command ==>

Command line:
  L string Scroll to reserved word

Action column:
  S Select a word for update
  D Delete word

Add or update word ==>
  Name:                Change Code:      Word Type:

PF1 Help  PF3 Exit  PF7 Up  PF8 Down  Enter Update

Action Reserved word name Change code Word type Default Code
      ACCEPT          990         03
      ACCESS          990         02
      ...
PF1 Help  PF3 Exit  PF4 Return  PF7 Up  PF8 Down  Enter Add/Delete/Update
  
```

Figure 27. Reserved Words panel

This panel lists the reserved words in alphabetical order.

To scroll through the list:

- Use PF7 and PF8.

To locate a specific reserved word:

- On the command line, type **L**, **LOC**, or **LOCATE**, followed by the word you want to find.
- Press Enter.

To add a new reserved word:

- Tab to **Add or update word**.
- Enter:
 - Name (required)
 - Change code (required)
 - Word type (optional)

To update a reserved word:

- Locate the reserved word.
- Type **S** in the adjacent **Action** entry field.
- Press Enter.

This places the reserved word into the **Add or update word** section of the panel, ready for updating.

- Overtyping existing fields.

- Press Enter.

To delete a reserved word:

- Locate the reserved word.
- Type **D** in the adjacent **Action** entry field.

The word is deleted from the file when you exit the panel. While you are using the panel, you can reinstate the word at any time by overtyping the **D** in the **Action** entry field with a space.

The fields in this panel are described below.

Reserved word name

This is the key field.

Change code

Indicates which LCP (if any) CCCA invokes when it encounters this word in the source program being converted:

- 999** Word does not invoke an LCP.
- 990** Invokes an LCP that has the word in the CONVER statement.
- nnn* (Other than 999 and 990). This word invokes an LCP that has LCP-*nnn* in its CONVER statement.

The following list shows the change codes used by CCCA and the change codes you can use for your own LCPs:

000

860-989

992-998

Reserved, used by CCCA.

001-799

Available for your own LCPs.

800-859

Used by supplied LCPs.

991

Used by CCCA.

Word type

Specifies where in a COBOL program the word occurs. You specify this value as two characters. Each character can be:

Indicates the word occurs...

- 1** In a paragraph or section name
- 2** At the beginning of a clause
- 3** At the beginning of a statement and its operands
- 5** At the beginning of a phrase

A pair of spaces or pair of zeros indicates the reserved word does not occur in any of the above places.

For definitions of division header, section header, paragraph header, clause, statement and phrase, see the *COBOL Language Reference* for your platform.

Default Code

The original IBM-supplied change code, displayed for informational purposes only. You cannot update this field.

Compiling LCPs under MVS

Use the LCP Compiler panels to submit a batch job to compile one or more LCPs.

To submit a compilation job:

1. Go to panel 2.2 to display the LCP Compiler job statement information panel (see Figure 28).

```
----- CCCA LCP Compiler job statement information -----
Command ==>

Job statement information:      (Verify before proceeding)
====> //VCATRCAH JOB (9999,040,090,ST3),'CCCA',
====> // NOTIFY=VCATRCA,TIME=5,
====> // REGION=4096K,USER=VCATRCA,MSGCLASS=V,CLASS=C
====> /*JOBPARM FORMS=SP2

SYSOUT class ==> *

PF1 Help  PF3 Exit  PF4 Return  ENTER Proceed
```

Figure 28. LCP compiler job statement information panel (MVS)

2. If necessary, update the text in:

Job statement information

The JCL for the LCP compile job card.

SYSOUT class

The output class to which you want the output of the LCP compile job sent.

SYSOUT class can be:

- Any letter (A through Z)
- Any numeral (0 through 9)
- An asterisk (*)

3. Press Enter to display the LCP Compiler selection panel (see Figure 29 on page 69).

```

----- CCCA LCP Compiler selection -----
Command ==>

LCP source:
Project. . . . ==> VCATRC2
Library. . . . ==> CCCA
Type . . . . ==> SABJLCP
Member . . . . ==>          (Blank for member selection list)

Other source file:
Data set name ==>

PF1 Help  PF3 Exit  PF4 Return  ENTER Generate JCL

```

Figure 29. LCP Compiler Selection panel (MVS)

4. Enter values for the full data set name and member name for one of these:
 - An ISPF library
 - Other partitioned data set name.

If you do not specify a member name, a member list is displayed. You may select members from the list by entering an S in front of the member names.

5. Press Enter.

ISPF generates the JCL for the compilation and then displays the LCP Compiler submission panel (see Figure 30).

```

----- CCCA LCP Compiler submission -----
Option ==>

Instructions:
Press ENTER to continue generating JCL.
Press PF3   to submit job and exit
Press PF4   to submit job and return
Press PF12  to exit without submitting job
Enter Cancel command to exit without submitting job.

      4 LCP member(s) built for compilation.

Job statement information:
//VCATRCAH JOB (9999,040,090,ST3),'CCCA',
// NOTIFY=VCATRCA,TIME=5,
// REGION=4096K,USER=VCATRCA,MSGCLASS=V,CLASS=C
/*JOBPARM FORMS=SP2

PF1 Help  PF3 Submit Job  PF4 Submit job  PF12 Cancel  ENTER Generate JCL
          and exit      and return      for member

```

Figure 30. LCP Compiler Submission panel (MVS)

This panel shows the number of LCP members that have been selected for compilation and redisplayes the Job card parameters for information only. This panel can no longer be overtyped, since the Job statement has already been generated.

To select additional LCPs to be compiled, press Enter,

Customizing

To cancel the submission of the job, type C on the command line and press Enter.

6. Press either PF3 or PF4.

ISPF submits the generated JCL for execution.

The message JOB *xxxxxc* SUBMITTED appears once for each member that you selected for compilation (where *xxxxxc* is the specified job name). The final message is followed by three asterisks (***) .

You may press Enter or any other interrupt key to return to the LCP Development Aid menu.

Compiling LCPs under VM

Use the LCP Compiler Selection panel to compile one or more LCPs.

To compile an LCP:

1. Go to panel 2.2 to display the LCP Compiler Selection panel see (Figure 31).

```
----- CCCA LCP Compiler selection -----
Command ==>

LCP source:
Project. . . . ==> CCCA
Library. . . . ==> REGTEST
Type . . . . ==> COBOL
Member . . . . ==>          (Blank for member selection list)

CMS file:
File ID      ==> sample1 cobol A
If not linked, specify:
Owner's ID ==>          Device addr. ==>          Link access mode ==>

Read password ==>          Update password ==>

PF1 Help  PF3 Exit  PF4 Return  ENTER Proceed
```

Figure 31. LCP Compiler Selection panel (VM)

2. Enter values for:

LCP source

If the LCP that you want to compile is within an ISPF partitioned data set, enter the data set name and the member name in the **Project**, **Library**, **Type**, and **Member** fields.

If the LCP that you want to compile is within a MACLIB, enter the MACLIB file name in **File ID** and the member name in **Member**.

If you do not specify a member name or an asterisk, CCCA displays the LCP Compiler Member Selection panel after you press Enter (see Figure 32 on page 71).

```

----- CCCA LCP Compiler member selection -----
Command ==>

Select the LCP member(s) to be compiled and press Enter
Press PF3 to initiate compilation

NAME      SELECT
UTIL00
UTIL01
UTIL05

```

Figure 32. LCP compiler member selection panel (VM)

Place an “S” in front of all members in the list that you want compiled.

CMS file–File ID

If the LCP that you want compiled is a simple CMS file, enter the file details (fn ft fm).

Linkage fields

If you are not already linked to the minidisk where the LCP resides, enter the appropriate details in **Owner's ID**, **Device addr**, and **Link access mode**.

Passwords

If required, enter the appropriate passwords in the **Read password** and **Update password** fields.

3. Press Enter.

CCCA compiles the LCP (or LCPs) that you have selected in foreground mode. If errors are encountered during the compilation process, CCCA displays a message.

If you used the LCP Compiler Member Selection panel to select one or more members for compilation, or specified an asterisk (*) to compile all members, CCCA displays a message indicating which member it is currently compiling.

When the compilation process is complete, CCCA redisplay the LCP Compiler Selection panel, with a message indicating the return code for the compilation.

4. To return to the LCP Development Aid menu, press PF3.

Deleting LCPs and activating/deactivating debugging for LCPs

To delete LCPs or activate/deactivate debugging for LCPs, go to panel 2.3 (the Delete/Debug LCP panel, shown in Figure 33 on page 72).

If debugging for an LCP is activated, during conversion CCCA generates a “trace” of each executed statement of the LCP.

Deleting an LCP only deletes the LCP from the LCP library. It does not delete the LCP source member.

```

----- CCCA Delete/Debug LCP ----- Row 1 to 48 of 178
Command ==>                               Scroll ==> PAGE

Actions:                                     Commands:
DBG   Activate debugging for an LCP         DBG   Activate debugging for all LCPs
blank Deactivate debugging for an LCP      CLR   Deactivate debugging for all LCPs
DEL   Delete LCP from LCP library          L string Scroll to string

PF1 Help  PF3 Exit  PF4 Return  PF7 Up  PF8 Down

Action  LCP name
...     ACCEPT
...     ACCESS
...     ACTUAL
...     ADD
...     ALL
...     ALPHABETIC
...     ALTER
...     APPLY
...     ASCENDING
...     ASSIGN
...     BLANK

```

Figure 33. Delete/Debug LCP panel

The panel lists the names of the LCPs in the LCP library in alphabetical order.

To scroll through the list, use PF7 and PF8.

To delete an LCP from the LCP library

Type **DEL** next to the LCP, then press PF3.

To put the LCP back in the library, compile the LCP (see “Compiling LCPs under MVS” on page 68 or “Compiling LCPs under VM” on page 70).

To activate debugging for an LCP

Type **DBG** next to the LCP, then press PF3.

To deactivate debugging for an LCP

Erase the **DBG** next to the LCP, then press PF3.

To activate debugging for all LCPs

Type **DBG** on the command line, then press Enter.

To clear actions for all LCPs

Type **CLR** on the command line, then press Enter.

To locate an LCP using a string search

Type **L xxx** on the command line, then press Enter.

Generating a directory of the LCP library

To generate a directory of the LCP library, go to panel 2 (the LCP Development Aid menu), then select option 4 (LCP DIRECTORY).

Figure 34 on page 73 is an extract from a directory of the LCP library.

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID -          04/16/98 15:47:58   Page   1
.....                L C P   D I R E C T O R Y           .....
1 RESERVED WORD      2 PROCESSING DESCRIPTION          3 DATE          4 TIME          CORE DBG
                                                                SIZE  OPT
                                                                5      6
-----
ACCESS                UPDATE FILE INFORMATION IN CONTROL FILE      04/15/98 09:49:44   575
ACTUAL                REPLACE BY RELATIVE                            04/15/98 09:49:52   630
ADD                   ADD WITH BLL'S                               04/15/98 09:49:58  8205
ALL                   MOVE ALL ...                                  04/15/98 09:50:44   885
ALPHABETIC           CHANGE TO ALPHABETIC-UPPER                   04/15/98 09:50:39   260

```

Figure 34. Extract from a directory of the LCP library

For each LCP in the LCP library, the directory lists:

- 1** COBOL reserved word or LCP-*nmn* identifier specified in the CONVER statement of the LCP.

As supplied, this is also the LCP source member name (or, if the reserved word is too long, the member name is an abbreviation of the reserved word).

- 2** Descriptive text in the CONVER statement of the LCP.
- 3** Date that the LCP was last compiled (in the format MM/DD/YY).
- 4** Time that the LCP was last compiled.
- 5** Size of the compiled LCP in bytes.

The maximum permitted size for a compiled LCP is 12600 bytes.

- 6** Indicates whether debugging for the LCP is activated:

blank Debugging is not activated

DBG Debugging is activated

The complete directory of the LCP library (as supplied) is shown in “LCP directory” on page 201.

Updating the message file

The message management facility interactively handles message file processing, making individual messages directly accessible at any time. Through the facility you can browse, add, update, and delete messages.

Select option 5 from the LCP Development Aid menu (see Figure 6 on page 14) to display the Messages panel (see Figure 35 on page 74).

```

----- CCCA Messages -----
Command ==>

  A - Add a new message
  U - Update existing message
  D - Delete existing message
  blank - Display of existing message

Message ID ==>
Severity . ==>      (00 - 99)

Short message:
  ==>                -
  ==>                -
  ==>                -
  ==>                -

Long message:
  ==>                -
  ==>                -
  ==>                -
  ==>                -

PF1 Help  PF3 Exit  PF4 Return

```

Figure 35. Messages panel

The fields are:

Command

Choose the command code from:

- A** Add a new message
- U** Update an existing message
- D** Delete an existing message
- Blank** Display an existing message

Message ID

This must always be entered. It takes the form `ABJnnnn`, where `nnnn` is the 4-digit identifier.

Severity

A 2-digit number (00 through 99). The severity level of a message can affect the output of CCCA. For details, see “Setting conversion options” on page 19.

Short message

This message will appear on the conversion diagnostic listing.

Long message

This message is informational only and will not be displayed on the listing. It is useful for displaying supplementary information or guidelines for handling statements that require manual inspection.

If this message will be longer than four 60-character lines, press Enter to display a message continuation panel, which allows you to create a longer message.

To display an existing message:

1. A **Command** value is not required
2. Enter a value for **Message ID**
3. Press Enter

To add a new message:

1. Enter a **Command** value of **A**

2. Enter values for: **Message ID**, **Severity**, **Short message**, and, optionally, **Long message**.
3. Press Enter

To update the severity level and message text of a message:

1. Enter a **Command** value of **U**
2. Enter a value for **Message ID**
3. Press Enter
4. Enter modified values for **Severity**, **Short message**, and **Long message** as required
5. Press Enter

To delete an existing message:

1. Enter a **Command** value of **D**
2. Enter a value for **Message ID**
3. Press Enter

Note: CCCA is delivered with messages in English. The message management facility may be used to replace the message text with that of a different language, as long as the language uses characters of the EBCDIC character set.

Chapter 7. Developing Language Conversion Programs

Read this chapter if you are planning to develop your own Language Conversion Programs (LCPs) or if you want to change the supplied LCPs.

This chapter describes:

- LCP language structure and syntax
- How to use LCP functions to:
 - Edit the tokenized source program
 - Read and update the files CCCA uses during conversion
- How to debug LCPs
- Differences between processing tokens and elements
- COBOL Reserved Word data set processing

This chapter documents intended Programming Interfaces that allow the customer to write programs to obtain the services of CCCA.

What is an LCP?

An LCP is a COBOL-like program containing:

- A subset of COBOL statements
- Calls to CCCA functions

What LCPs do

LCPs generate change requests to convert language elements from one COBOL implementation to another.

(CCCA invokes LCPs in conversion phase 2, and applies their change requests in phase 3. For details, see “How CCCA invokes LCPs” on page 63.)

LCPs can:

- Add, replace, or remove words, clauses or statements
- Indicate areas in the converted code you should review for possible manual changes
- Update conversion statistics in the Control file

CCCA is supplied with LCPs for converting between several COBOL implementations (listed in “What CCCA does” on page 1).

You can customize CCCA to meet your installation's requirements by:

- Developing new LCPs
- Modifying the supplied LCPs

For a list of LCPs supplied with CCCA, see Appendix G, “LCP directory,” on page 191.

LCP structure

Here are the details of the structure required in an LCP.

LCP divisions

LCPs contain three separate COBOL-like divisions (but unlike COBOL, there are no division headers):

Identification Division

Consists of only one statement: CONVER, CONVERA, or CONVERQ. This statement identifies and describes the LCP, and specifies whether nonnumeric literals are enclosed in apostrophes (') or quotation marks (").

Data Division (optional)

Consists of data description entries.

Your LCP may not need a Data Division, because many of the data items you use in an LCP are predefined by CCCA, and do not need a Data Division entry.

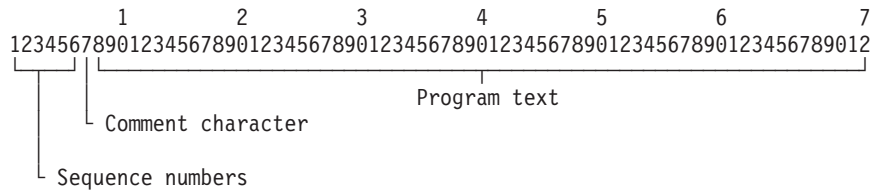
Procedure Division

Consists of the statements and function calls that define the conversion process.

The Procedure Division must start with a paragraph name.

LCP source line format

LCP source lines use the 72-column COBOL reference format:



Column

Is used for...

1 through 6

Sequence numbers

7

Comment character (except for 05 and 77 data description entries and the CONVER statement, this indicates the remainder of the line is comment text)

8 through 72

Program or comment text

Characters

Table 4 lists the characters you can use in an LCP, their meaning, and their use.

Note:

1. You **cannot** use lowercase letters (a–z) in LCPs, except in comments and nonnumeric literals.
2. Comments and nonnumeric literals can contain *any* EBCDIC character.

Table 4. LCP characters—their meanings and uses

Character	Meaning	Use
b	Space or Blank	Punctuation
.	Decimal point or Period	Punctuation

Table 4. LCP characters—their meanings and uses (continued)

Character	Meaning	Use
0–9	Numerals	Data item identifiers Nonnumeric literals Numeric literals Paragraph names Reserved words
A–Z	Alphabet (uppercase only)	Data item identifiers Nonnumeric literals Paragraph names Reserved words
-	Hyphen	Data item identifiers Paragraph names Reserved words
*	Asterisk	In column 7, indicates the remainder of the line is a comment (except for 05 and 77 data entry descriptions and the CONVER statement)
/	Stroke or Slash	In column 7, indicates the remainder of the line is a comment
=	Equal sign	Relational operator in conditions (synonym for EQUAL TO)
>	Greater than	Relational operator in conditions (synonym for GREATER THAN)
<	Less than	Relational operator in conditions (synonym for LESS THAN)
'	Apostrophe	Encloses nonnumeric literals (if you specify the CONVER or CONVERA statement)
"	Quotation mark	Encloses nonnumeric literals (if you specify the CONVERQ statement)

Data item identifiers and paragraph names

Data item identifiers and paragraph names:

- Must start with a letter (A through Z)
- Can contain these characters:
 - 0 through 9
 - A through Z
 - (hyphen)
- Can contain up to 30 characters
- Cannot end with a hyphen

Reserved words

You cannot use LCP reserved words for paragraph names or for your own data item identifiers.

LCP reserved words consist of:

- COBOL language elements, keywords, and related symbols
- LCP function names

Developing LCPs

- Predefined data item identifiers

For a complete list of LCP reserved words, see Appendix D, “LCP reserved words,” on page 167.

Literals

Nonnumeric literals

A nonnumeric literal is a character string enclosed by apostrophes (') or quotation marks (") and containing any EBCDIC character. The maximum length of a nonnumeric literal is 30 characters.

If you want to imbed an enclosing character in a nonnumeric literal, you must specify a pair of enclosing characters. For example:

```
"THIS ISN""T WRONG"
```

The choice of apostrophe or quotation mark is specified by the CONVER statement at the start of an LCP:

CONVER or CONVERA

Specifies that nonnumeric literals are enclosed by apostrophes (')

CONVERQ

Specifies that nonnumeric literals are enclosed in quotation marks (")

Numeric literals

A numeric literal is a string of digits (0 through 9) with a maximum length of 10 digits. Numeric literals are unsigned.

Comment lines

Comments appear on a line by themselves; you cannot mix code and comments on the same source line.

Comment lines can appear anywhere in an LCP.

Format



Notes:

- 1 An asterisk (*) or a slash (/) must appear in column 7.

comment-text

Can contain any EBCDIC characters.

Punctuation

Statements

Each statement must begin on a new line.

Paragraphs

A paragraph is a sequence of statements, beginning with a paragraph name. A paragraph name is a label that can be referred to by GO TO and PERFORM statements.

Paragraph names must appear on a line by themselves.

Periods

A period must appear immediately following:

- The last statement in a paragraph
- A paragraph name
- A data item identifier
- The last statement within an IF statement (for details, see “IF statement” on page 86)

A period may appear after any statement (except CONVER, CONVERA, or CONVERQ); except for the situations described in the list above, these trailing periods are optional and are not significant.

Blank lines

Blank lines can appear anywhere in an LCP.

LCP statement summary

Table 5 shows a summary of LCP statements.

Table 5. LCP statement summary

Division	Statement	Description
Identification Division	CONVER, CONVERA, CONVERQ	Identifies and describes the LCP, and specifies whether nonnumeric literals are enclosed in apostrophes (') or quotation marks (").
Data Division (optional)	01, 05, 77	Defines data items. Your LCP may not need a Data Division, because many of the data items you use in an LCP are predefined by CCCA, and do not need a Data Division entry.
Procedure Division	ADD	Adds one number to another.
	EXIT	Must appear in the last paragraph executed by a PERFORM THRU statement.
	GO TO	Transfers control to another paragraph in the LCP. GO TO END-CHANGE terminates the LCP.
	IF	Controls the execution of statements by testing a condition. For information on conditions, see “Conditions” on page 84.
	MOVE	Copies a numeric or nonnumeric literal or data item to another data item.
	PERFORM	Executes one or more paragraphs a specified number of times or until a specified condition is true.
	SUBTRACT	Subtracts one number from another.

The following sections describe each statement in detail.

Identification Division

Here are the details of the contents of the Identification Division.

CONVER statement

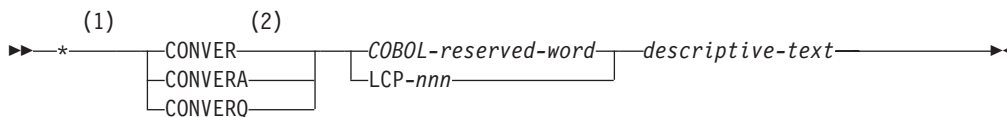
The CONVER statement:

- Identifies and describes the LCP

Developing LCPs

- Specifies whether nonnumeric literals are enclosed in apostrophes (') or quotation marks (").

Format



Notes:

- 1 An asterisk (*) must appear in column 7.
- 2 CONVER, CONVERA, or CONVERQ must appear in columns 12–18.

CONVER or CONVERA

Specifies that nonnumeric literals are enclosed in apostrophes (').

CONVERQ

Specifies that nonnumeric literals are enclosed in quotation marks (").

COBOL-reserved-word

The COBOL reserved word that this LCP converts. Must be alphanumeric, starting with an alphabetic character.

This word must appear in the COBOL Reserved Word data set. For details, see "Updating the COBOL reserved word Data Set" on page 66.

*LCP-*nnn**

If this LCP converts more than one COBOL reserved word, identify the LCP by "LCP-" followed by three digits (for example: LCP-352).

Each COBOL reserved word that this LCP converts must appear in the COBOL Reserved Word data set, with a change code of *nnn*. For details, see "Updating the COBOL reserved word Data Set" on page 66.

descriptive-text

A nonnumeric literal (with a maximum length of 50 characters) that describes what the LCP does.

Must be enclosed in either apostrophes (') or quotation marks ("), depending on whether you specified CONVER, CONVERA, or CONVERQ.

For example: "OTHERWISE replaced by ELSE".

Note:

1. The LCP directory lists the *COBOL-reserved-word* (or *LCP-*nnn**) and the *descriptive-text* of all LCPs. To view or print the LCP directory, see "Generating a directory of the LCP library" on page 72.
2. As supplied, LCP source member names are the same as the identifier in this CONVER statement: either *COBOL-reserved-word* or *LCPnnn*.
Note that you specify *LCP-*nnn** in the CONVER statement, but the LCP source member name is *LCPnnn*, with no hyphen.
3. You can use any name for your LCP source members; CCCA only looks at the identifier in the LCP's CONVER statement (not its source member name).

Data Division (Optional)

Data Division entries define data items.

You can define only elementary data items in an LCP.

Unlike a COBOL program, in an LCP there is no difference between “05” and “77” data items. The 01, 05, and 77 numbers are kept only to maintain a COBOL-like appearance.

Note: Your LCP may not need a Data Division, because many of the data items you use in an LCP are predefined by CCCA, and do not need a Data Division entry. For a complete list of predefined data items, see Appendix E, “Predefined data items,” on page 175.

Format 1 (treated as comment only)



Notes:

- 1 An asterisk (*) or a blank must appear in column 7. 01 must appear in columns 8–9.

Format 2



Notes:

- 1 An asterisk (*) or a blank must appear in column 7. 77 must appear in columns 8–9.

Format 3



Notes:

- 1 An asterisk (*) or a blank must appear in column 7. 05 must appear in columns 12–13.

identifier

A data item identifier:

- Must start with a letter (A through Z)
- Can contain these characters:
 - 0 through 9
 - A through Z
 - (hyphen)
- Can contain up to 30 characters
- Cannot end with a hyphen

Developing LCPs

9(*n*)

Specifies that the data item is numeric (can contain only digits), with length *n* (where *n* is 1 through 10).

If *n* is less than 10, you can add a leading zero. For example: 9(03).

X(*n*)

Specifies that the data item is alphanumeric (can contain any EBCDIC characters), with length *n* (where *n* is 1 through 30).

If *n* is less than 10, you can add a leading zero. For example: X(09).

Format 1

Treated as a comment. An asterisk (*) or a blank must be in column 7, and 01 must appear in columns 8–9.

Format 2

An asterisk (*) or a blank must be in column 7, and 77 must appear in columns 8–9.

Format 3

An asterisk (*) or a blank must be in column 7, and 05 must appear in columns 12–13.

Procedure Division

Here are the details of the Procedure Division

ADD statement

The ADD statement adds two numbers, and stores the result in the data item *identifier-2*.

Format

➤—ADD—*identifier-1*—TO—*identifier-2*—
 └─*n*— └─.─

identifier-1

identifier-2

Numeric data items.

n A numeric literal.

Conditions

The IF and PERFORM UNTIL statements use conditions to determine whether or not to execute statements.

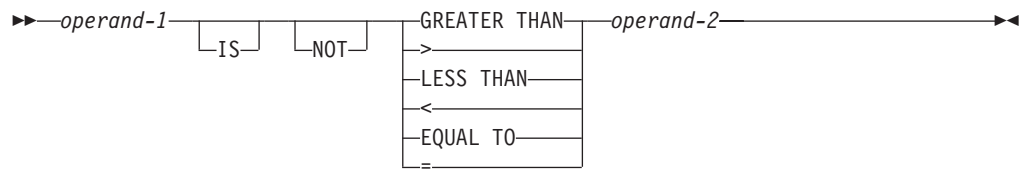
There are two types of condition: simple and combined.

Simple conditions

Simple conditions must appear on the same source line as IF or UNTIL. For example:

```
IF simple-condition  
   statement-1  
ELSE  
   statement-2.
```

Format



operand-1
operand-2

The operands to be compared. These can be literals or data items, but they must be of the same type (numeric or nonnumeric). You cannot compare a numeric operand with a nonnumeric operand.

Combined conditions

A combined condition consists of either:

Format 1

Two or more simple conditions connected by OR

Format 2

Two or more simple conditions connected by AND

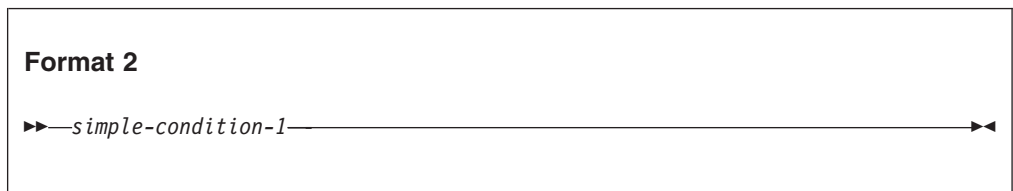
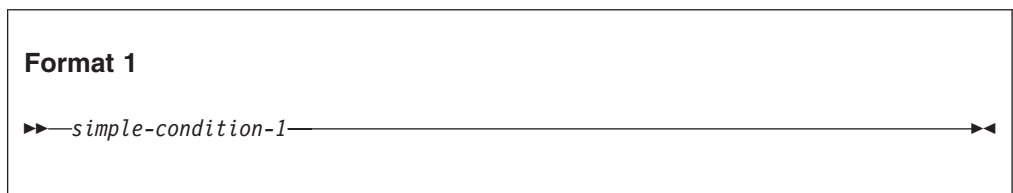
You cannot mix OR and AND in a combined condition.

In a combined condition:

- *simple-condition-1* must appear on the same source line as IF or UNTIL
- “OR *simple-condition-2*” and “AND *simple-condition-2*” must appear on a separate source line, immediately following the IF or UNTIL line

For example:

```
PERFORM function-name
UNTIL simple-condition-1
      OR simple-condition-2
```





simple-condition-1
simple-condition-2

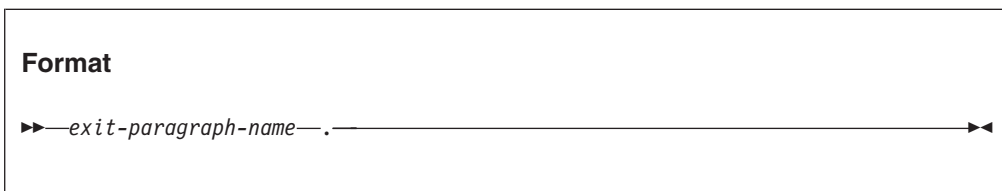
Simple conditions. For details, see “Simple conditions” on page 84.

EXIT statement

The EXIT statement must appear in the last paragraph executed by a PERFORM THRU statement.

The EXIT statement:

- Must appear on a line immediately below a paragraph name
- Must be immediately followed by a period
- Must be the only statement in the paragraph



exit-paragraph-name

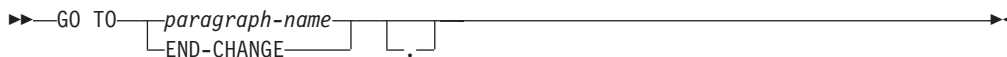
The paragraph name appearing after THRU in the PERFORM statement. For details, see “PERFORM statement” on page 89.

GO TO statement

The GO TO statement transfers control to another paragraph in the LCP.

GO TO END-CHANGE terminates the LCP.

Format



paragraph-name

A paragraph name in the LCP.

END-CHANGE

Terminates the LCP.

An LCP can contain multiple GO TO END-CHANGE statements.

Note: END-CHANGE must not appear as a paragraph name in an LCP.

IF statement

The IF statement controls the execution of statements by testing a condition.

For information on conditions, see “Conditions” on page 84.



statement-1

One or more statements, executed only if the *condition* is true.

Must appear on a separate line from IF (and ELSE).

Cannot contain another IF statement.

ELSE (Format 2 only)

Specifies that the statements to follow are executed only if the *condition* is false.

Must appear on a separate line from *statement-1* and *statement-2*.

***statement-2* (Format 2 only)**

One or more statements, executed only if the *condition* is false.

Must appear on a separate line from ELSE.

Cannot contain another IF statement.

Note:

1. **Periods.** The last statement under the control of the IF statement (and only the last statement) **must** end with a period.

For example:

Developing LCPs

```

IF condition
  statement-1
  statement-1
  statement-1.
.
.
.
IF condition
  statement-1
  statement-1
ELSE
  statement-2
  statement-2.

```

2. *statement-1* and *statement-2* cannot contain IF statements (IF statements cannot be nested).

MOVE statement

The MOVE statement copies a numeric or nonnumeric literal or data item to the data item *identifier-2*.

Format

```

▶▶ MOVE identifier-1 TO identifier-2
      [ literal ]

```

identifier-1

The data item containing the numeric or nonnumeric value that you want to copy to *identifier-2*.

literal

The numeric or nonnumeric literal that you want to copy to *identifier-2*. Nonnumeric literals must appear inside enclosing characters.

You can MOVE only:

- Numeric data to numeric data (right-justified)
- Alphanumeric data to alphanumeric data (left-justified)
- Numeric data to alphanumeric data (left-justified)

MOVE type	<i>identifier-1</i> or <i>literal</i>	<i>identifier-2</i>	
		Before	After
numeric to numeric	2 1 5 4 2 1 7 5 4 2 1	1 0 2 5 1 0 2 5 1 0 2 5	0 0 2 1 5 4 2 1 5 4 2 1
alphanumeric to alphanumeric	H E I C H E F I C H E	D A T E D A T E D A T E	H E I C H E F I C H
numeric to alphanumeric	2 1 5 4 2 1 7 5 4 2 1	D A T E D A T E D A T E	2 1 5 4 2 1 7 5 4 2

Paragraph names

A paragraph is a sequence of statements, beginning with a paragraph name. A paragraph name is a label that can be referred to by GO TO and PERFORM statements.

The Procedure Division must start with a paragraph name.

Paragraph names must appear on a line by themselves.

Format

▶▶—*paragraph-name*—.

paragraph-name

A paragraph name:

- Must start with a letter (A through Z)
- Can contain these characters:
 - 0 through 9
 - A through Z
 - (hyphen)
- Can contain up to 30 characters
- Cannot end with a hyphen
- Must be immediately followed by a period (.)

PERFORM statement

The PERFORM statement:

- Executes a function a specified number of times (default is once)
- Executes a sequence of paragraphs once only
- Executes a function or sequence of paragraphs one or more times, until a specified condition is true

Format 1

▶▶—PERFORM—*function-name*—
└—*numeric-literal*—TIMES └—.

Format 2

▶▶—PERFORM—*paragraph-name*—THRU—*exit-paragraph-name*—
└—.

Format 3

▶▶—PERFORM—*function-name*—
└—*paragraph-name*—THRU—*exit-paragraph-name*—

▶▶—UNTIL—*condition*—
└—.

function-name

An LCP function. For more information, see “LCP functions” on page 91.

numeric-literal

A numeric literal with a maximum value of 12.

Developing LCPs

paragraph-name

The name of the first paragraph in the LCP to be executed by the PERFORM THRU statement.

The paragraphs executed by the PERFORM THRU statement can appear in the LCP source either before or after the PERFORM THRU statement.

paragraph-name must appear in the LCP source before the *exit-paragraph-name*.

For example:

```
FIRST-PARA.  
.  
.  
SECOND-PARA.  
.  
.  
THIRD-PARA.  
.  
.  
END-PARA.  
  EXIT.  
.  
.  
.  
PERFORM FIRST-PARA THRU END-PARA  
UNTIL condition
```

exit-paragraph-name

The name of the last paragraph in the LCP to be executed by the PERFORM THRU statement. This paragraph must contain only an EXIT statement. For details, see “EXIT statement” on page 86.

UNTIL (Format 3 only)

Must appear on a separate source line from PERFORM.

condition

A simple or combined condition. For details, see “Conditions” on page 84.

The *condition* is tested **after** the function or entire paragraph sequence has executed.

SUBTRACT statement

The SUBTRACT statement subtracts a number from *identifier-2*, and stores the result in the data item *identifier-2*.

Format

```
►► ——— identifier-1 ——— FROM — identifier-2 ————— ◀◀  
    └── numeric-literal ───┘
```

identifier-1

The data item containing the numeric value that you want to subtract from *identifier-2*.

identifier-2

A numeric data item.

numeric-literal

An unsigned number (with a maximum length of 10 digits) that you want to subtract from *identifier-2*.

LCP functions

You use LCP functions to manipulate the tokenized source of the program being converted.

In conversion phase 1 (see “How CCCA works” on page 4), CCCA breaks apart the COBOL source program into *elements* and *tokens*.

Elements are:

- Character strings in COBOL COPY statements
- COBOL comment paragraph lines

All other character strings are tokens.

Note: Comment lines, and the following compiler directives, are *not* tokenized:

- SKIP1
- SKIP2
- SKIP3
- EJECT
- TITLE
- *CBL
- *CONTROL

In conversion phase 2, CCCA invokes LCPs to examine the tokenized source and generate change requests.

The LCPs use functions that:

- Retrieve tokenized source
- Bypass token identifiers
- Remove tokenized source
- Modify tokenized source and insert tokens
- Edit tokens
- Construct tokens
- Bypass token processing

The functions write the change requests to the CHANGE data set.

In conversion phase 3, CCCA applies the change requests to the source program.

Using LCP functions

You invoke LCP functions using the PERFORM statement:

```
PERFORM function-name
```

You pass values to and from LCP functions using *predefined data items*.

For example:

```
PERFORM GET-FIRST-TOKEN
```

retrieves information about the first token of the program, and places the information in predefined data items (such as TOKEN-LENGTH and TOKEN-TEXT) that you can examine and modify.

Similarly:

```
MOVE 'NEW COMMAND' TO ADD-TEXT
PERFORM REPLACE-TOKEN
```

Developing LCPs

replaces the current token in the program with the value you moved to the predefined data item ADD-TEXT. (Since ADD-TEXT is a predefined data item, you do not need to include it as an entry in the LCP Data Division.)

The following sections describe the LCP functions, and list their related predefined data items.

For a complete list of LCP functions, see Appendix F, "List of LCP functions," on page 187.

For a complete description of each predefined data item, see Appendix E, "Predefined data items," on page 175.

Retrieving tokenized source

The following functions retrieve tokenized source of the program being converted:

LCP function	Description
GET-FIRST-TOKEN or GET-FIRST	Retrieve the first token of the program
GET-LAST-TOKEN or GET-LAST	Retrieve the last token of the program
GET-TOKEN	Retrieve the token or element for the pointer value currently set
GET-NEXT-TOKEN or GET-NEXT	Retrieve the token following the current record or the pointer value currently set
GET-PREVIOUS-TOKEN or GET-PREVIOUS	Retrieve the token preceding the current record or the pointer value currently set
GET-ELEMENT	Retrieve the token or element for the pointer value currently set
GET-NEXT-ELEMENT	Retrieve the element or token following the current record or the pointer value currently set
GET-PREVIOUS-ELEMENT	Retrieve the element or token preceding the current record or the pointer value currently set

These functions return values in the following predefined data items:

05 TOKEN-SEQUENCE	PIC X(6) .
05 TOKEN-POSITION	PIC 9(2) .
05 TOKEN-LENGTH	PIC 9(3) .
05 TOKEN-TYPE-CODE	PIC X(1) .
05 TOKEN-CHANGE-CODE	PIC 9(3) .
05 TOKEN-LINE-CODE	PIC X(1) .
05 TOKEN-FLAG	PIC X(2) .
05 TOKEN-TEXT	PIC X(30) .
05 TOKEN-SOURCE	PIC X(1) .

TOKEN-POSITION refers to the column number within the program text area (columns 8 through 72). For example, a TOKEN-POSITION value of 5 refers to column 12 in the generated source program.

Moving through the tokenized source

The TOKEN-POINTER predefined data item determines the current token of the program being converted.

You can move through the tokenized source by changing the value of TOKEN-POINTER. Figure 36 shows how to save the current token pointer, then move back to that token later in the LCP.

```

/*****
*
*   CONVERA EXAMPLE      'SHOW USE OF TOKEN POINTER'      *
*   .
*   .
*   .
*   05  TOKEN-POINTER-SAVE      PIC 9(7)  .
*   .
*   .
*   PERFORM GET-NEXT-TOKEN.
*   SAVE CURRENT TOKEN POSITION
*   MOVE TOKEN-POINTER TO TOKEN-POINTER-SAVE.
*   .
*   .
*   .
*   RE-ESTABLISH TOKEN POSITION
*   MOVE TOKEN-POINTER-SAVE TO TOKEN-POINTER.
*   PERFORM GET-TOKEN.
*   .
*   .
*   .
*   GO TO END-CHANGE.

```

Figure 36. Saving and repositioning TOKEN-POINTER

Bypassing token identifiers

The BYPASS-IDENTIFIER function bypasses the tokens that qualify the current token:

LCP function	Description
BYPASS-IDENTIFIER	Bypass qualifier, subscript, index, and reference modifier of a data item

This function returns values in the following predefined data items:

```

05  BYPASSED-REF-TYPES      PIC X(3).
05  BYPASSED-REF-QUAL      PIC X(1).
05  BYPASSED-REF-SUB      PIC X(1).
05  BYPASSED-REF-MOD      PIC X(1).

```

Removing tokenized source

The following functions remove source from the program being generated:

LCP function	Description
REMOVE-TOKEN or REMOVE	Remove the last token or element read
REMOVE-NEXT-TOKEN or REMOVE-NEXT	Get next token and remove it
REMOVE-CLAUSE	Remove the clause
REMOVE-STATEMENT	Remove the statement

Be careful when using the REMOVE-STATEMENT and REMOVE-CLAUSE functions. They remove from the token just read (the current token) until the beginning of a new statement or clause. The beginning of clauses and statements are defined by the **Word type** field in the COBOL Reserved Word data set (see "Updating the COBOL reserved word Data Set" on page 66).

Modifying tokenized source and inserting tokens

The following functions modify or insert code into the program being generated:

LCP function	Description
INSERT-BEFORE-TOKEN or INSERT-BEFORE	Insert new text before the current token Note: If you insert text before the first token of a line, the INSERT-BEFORE function inserts the text after the last token of the preceding line (there is no shuffling of tokens across the line).
REPLACE-TOKEN or REPLACE	Replace the current token
INSERT-AFTER-TOKEN or INSERT-AFTER	Insert text after the current token providing an intervening space
SUFFIX-TOKEN or SUFFIX	Append text to the current token without an intervening space
REMOVE-SUFFIX	Remove suffix from token

You pass values to these functions in the following predefined data items:

```

05 ADD-GROUP.
   10 ADD-LENGTH          PIC 9(2).
   10 ADD-TEXT            PIC X(30).
05 STARTING-POSITION     PIC 9(2).
```

Move the new text into ADD-TEXT.

STARTING-POSITION refers to the column number within the program text area (columns 8 to 72) where you want to add the text. For example, a STARTING-POSITION value of 5 refers to column 12 in the generated source program.

If you want CCCA to determine the length of the data in ADD-TEXT, set ADD-LENGTH to zero.

If ADD-TEXT contains imbedded blanks, or you want ADD-TEXT to have trailing blanks, set ADD-LENGTH to an appropriate value.

Note: Before using these functions, always set ADD-LENGTH to zero or some other appropriate value. Otherwise, you may inadvertently use a previous, and inappropriate, value for ADD-LENGTH.

For example:

```
*****
* Replaces current token with 'GO' (*not* 'GO TO')
*****
* Replace token in same position
  MOVE TOKEN-POSITION TO STARTING-POSITION
* ADD-LENGTH zero tells interpreter to determine length
  MOVE 0 TO ADD-LENGTH
* Note imbedded blank - interpreted as end of string
  MOVE 'GO TO' TO ADD-TEXT
* Replaces current token with 'GO' (*not* 'GO TO')
  PERFORM REPLACE-TOKEN
.
.
.
*****
* Replaces current token with 'GO TO'
*****
  MOVE TOKEN-POSITION TO STARTING-POSITION
  MOVE 5 TO ADD-LENGTH
  MOVE 'GO TO' TO ADD-TEXT
  PERFORM REPLACE-TOKEN
```

Note:

1. If an LCP contains more than one REPLACE-TOKEN function for the same token, only the last REPLACE-TOKEN has an effect. For example:

```
MOVE 'COMMAND ONE' TO ADD-TEXT
PERFORM REPLACE-TOKEN
MOVE 'COMMAND TWO' TO ADD-TEXT
PERFORM REPLACE-TOKEN
MOVE 'COMMAND THREE' TO ADD-TEXT
PERFORM REPLACE-TOKEN
```

has the same effect as:

```
MOVE 'COMMAND THREE' TO ADD-TEXT
PERFORM REPLACE-TOKEN
```

2. If the last statement of a COBOL program is COPY, the last character string (which should be a period) of the main program is considered to be the last token. It is not the last character string of the COPY member. Therefore, if you add code to the end of the program, it will appear on the listing immediately after the COPY statement. The expansion of the COPY module will appear after the section and not right after the COPY statement.

Editing tokens

The following functions edit tokens in the program being generated:

LCP function	Description
SPLIT-LINE	Start a new line
MAINTAIN-LINE-POSITION	Try to write in the same column of the line if there is enough space
COMMENT	Put an asterisk (*) in column 7
DIAGNOSTIC	Write the contents of the ADD-TEXT predefined data item in the diagnostic area
EJECT	Put a slash (/) in column 7
EDIT-MESSAGE	Write a message identifier, return code, and message text in the diagnostic area, according to the value of the MESSAGE-ID predefined data item

DIAGNOSTIC function

The DIAGNOSTIC function causes message text to be written to the Diagnostic listing. Use this function to write messages that do not appear in the Message file.

Before calling DIAGNOSTIC, move the message text to the ADD-TEXT predefined data item.

For example:

```
MOVE 'DIAGNOSTIC MESSAGE' TO ADD-TEXT
PERFORM DIAGNOSTIC
```

writes 'DIAGNOSTIC MESSAGE' in the diagnostic message area of the statement it applies to. (The diagnostic message area is on the right hand side of the Diagnostic listing.) The message is repeated in the message summary at the end of the listing.

Note: CCCA assigns the message identifier ABJ9999 to DIAGNOSTIC messages.

EDIT-MESSAGE function

The EDIT-MESSAGE function causes messages to be written to the Diagnostic listing. Unlike the DIAGNOSTIC function (where you specify the message text directly), with the EDIT-MESSAGE function you refer to a message identifier in the Message file:

```
MOVE 'ABJ6018' TO MESSAGE-ID
PERFORM EDIT-MESSAGE
```

The message text appears in the diagnostic message area of the statement it applies to. (The diagnostic message area is on the right hand side of the Diagnostic listing.) The message is repeated in the message summary at the end of the listing. For more information on the Message file, see “Updating the message file” on page 73.

Constructing tokens

The following functions construct tokens:

LCP function	Description
DETERMINE-LENGTH	Determines the length of the character string in ADD-TEXT, and puts the result in ADD-LENGTH
MOVE-LCP	Move characters

LCP function	Description
STRING-LCP	Concatenate character strings
UNSTRING-LCP	Break apart a character string into one or more character strings
CONVERT-ALPHA-NUMERIC	Convert an alphanumeric string into a numeric string

DETERMINE-LENGTH function

The DETERMINE-LENGTH function determines the length of the character string you have moved to the ADD-TEXT predefined data item, and puts the result in the ADD-LENGTH predefined data item.

For example, when you code:

```
MOVE 'ACCESS' TO ADD-TEXT
PERFORM DETERMINE-LENGTH
```

you use these predefined data items:

```
10 ADD-LENGTH          PIC 9(2).
10 ADD-TEXT            PIC X(30).
```

CCCA analyzes the contents of ADD-TEXT:



and produces the result:

```
ADD-LENGTH = 06
```

ADD-LENGTH is determined by the position of the first blank character in ADD-TEXT.

MOVE-LCP function

The MOVE-LCP function copies characters from one data item to another.

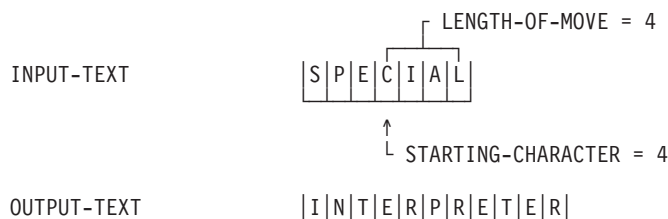
The MOVE-LCP functions uses these predefined data items:

```
05 INPUT-TEXT          PIC X(30).
05 STARTING-CHARACTER  PIC 9(2).
05 RECEIVING-CHARACTER PIC 9(2).
05 LENGTH-OF-MOVE      PIC 9(2).
05 OUTPUT-TEXT         PIC X(30).
```

For example, these statements:

```
MOVE 'SPECIAL' TO INPUT-TEXT
MOVE 'INTERPRETER' TO OUTPUT-TEXT
MOVE 4 TO STARTING-CHARACTER
MOVE 8 TO RECEIVING-CHARACTER
MOVE 4 TO LENGTH-OF-MOVE
```

set these values:




```

STRING-WORD-07  | | | | | | | | | | . . . | |
STRING-WORD-08  | | | | | | | | | | . . . | |
STRING-WORD-09  | | | | | | | | | | . . . | |
STRING-WORD-10  | | | | | | | | | | . . . | |
    
```

producing these results:

```

STRING-TEXT      | C | O | P | Y | R | E | C | O | R | D | - | N | A | M | E |
    
```

STRING-LENGTH = 15

Note: STRING-DELIMITER contains the character that the STRING-LCP function uses to determine the end of each STRING-WORD-*nn* character string. The default STRING-DELIMITER value is SPACE.

UNSTRING-LCP function

The UNSTRING-LCP function breaks apart the character string in the STRING-TEXT predefined data items, and stores the parts in the STRING-WORD-*nn* predefined data items.

The UNSTRING-LCP function uses these predefined data items:

```

      05 STRING-TEXT          PIC X(30).
      05 STRING-DELIMITER    PIC X(1).
      05 STRING-LENGTH      PIC 9(2).
01 STRING-WORDS.
      05 STRING-WORD-01     PIC X(30).
      05 STRING-WORD-02     PIC X(30).
      05 STRING-WORD-03     PIC X(30).
      05 STRING-WORD-04     PIC X(30).
      05 STRING-WORD-05     PIC X(30).
      05 STRING-WORD-06     PIC X(30).
      05 STRING-WORD-07     PIC X(30).
      05 STRING-WORD-08     PIC X(30).
      05 STRING-WORD-09     PIC X(30).
      05 STRING-WORD-10     PIC X(30).
    
```

For example:

```

MOVE 'DA-3340-I-CLIENT' TO STRING-TEXT
MOVE '-' TO STRING-DELIMITER
PERFORM UNSTRING-LCP
    
```

breaks apart the character string:

```

STRING-TEXT      | D | A | - | 3 | 3 | 4 | 0 | - | I | - | C | L | I | E | N | T |
    
```

into:

```

STRING-WORD-01   | D | A | | | | | | | | | | (30 characters)
STRING-WORD-02   | 3 | 3 | 4 | 0 | | | | | | | | |
STRING-WORD-03   | I | | | | | | | | | |
    
```

Developing LCPs

```

STRING-WORD-04  |C|L|I|E|N|T| . . . | |
STRING-WORD-05  | | | | | | | | . . . | |
STRING-WORD-06  | | | | | | | | . . . | |
STRING-WORD-07  | | | | | | | | . . . | |
STRING-WORD-08  | | | | | | | | . . . | |
STRING-WORD-09  | | | | | | | | . . . | |
STRING-WORD-10  | | | | | | | | . . . | |

```

Note: Before breaking apart STRING-TEXT, the UNSTRING-LCP function initializes the STRING-WORD-*nn* predefined data items to SPACES.

CONVERT-ALPHA-NUMERIC function

The CONVERT-ALPHA-NUMERIC function converts the left-aligned character string in the LCP-ALPHA predefined data item to a numeric value, and stores the numeric value in the LCP-NUMERIC predefined data item.

The CONVERT-ALPHA-NUMERIC function uses these predefined data items:

```

05 LCP-ALPHA      PIC X(10).
05 LCP-NUMERIC    PIC 9(10).

```

For example:

```

MOVE '1234' TO LCP-ALPHA
PERFORM CONVERT-ALPHA-NUMERIC

```

converts the alphanumeric string:

```

LCP-ALPHA        |1|2|3|4| | | | | | | |

```

to the numeric string:

```

LCP-NUMERIC      |0|0|0|0|0|0|1|2|3|4|

```

BYPASS-POINTER function

To bypass processing relating to the current token, use the BYPASS-POINTER Function:

LCP function	Description
BYPASS-OPTION	Bypass the conversion process associated with the token currently in storage

If the current token:

- Is after the token that invoked the LCP, and
- Has a change code that will invoke its own LCP

then the invocation of the BYPASS-POINTER function will result in that LCP not being invoked for the current token.

The BYPASS-POINTER function updates the change code in the current token to 994, causing the LCP processing to be bypassed.

Manipulating files

During conversion, CCCA uses two physical files: Control and Work.

The Control file contains five record types:

OPTION

COBOL source program member name and conversion options.

PROGRAM

Program name (as defined inside the COBOL program) before and after conversion, and conversion status.

FILE Information about each file (such as organization and access mode) used in the COBOL program.

CALL Details of CALL statements in the COBOL program.

COPY Details of COPY statements in the COBOL program.

The Work file contains thirteen record types:

KEY KEY clause information (if supplied) for each file used in the COBOL program.

RECORD

Records names linked to each file used in the COBOL program.

WORK-*nn*

(where *nn* is 01 through 10) Storage for miscellaneous conversion information.

CICS Details of BLL statements in the COBOL program.

CCCA makes selected information in these records available to you as predefined data items.

One of the predefined data items for each record is an *access key* that you can use to retrieve or update a specific record. (Except KEY, which is linked to the FILE record, and OPTION, which is a single record.) Figure 37 on page 102 shows the relationships between these records.

Developing LCPs

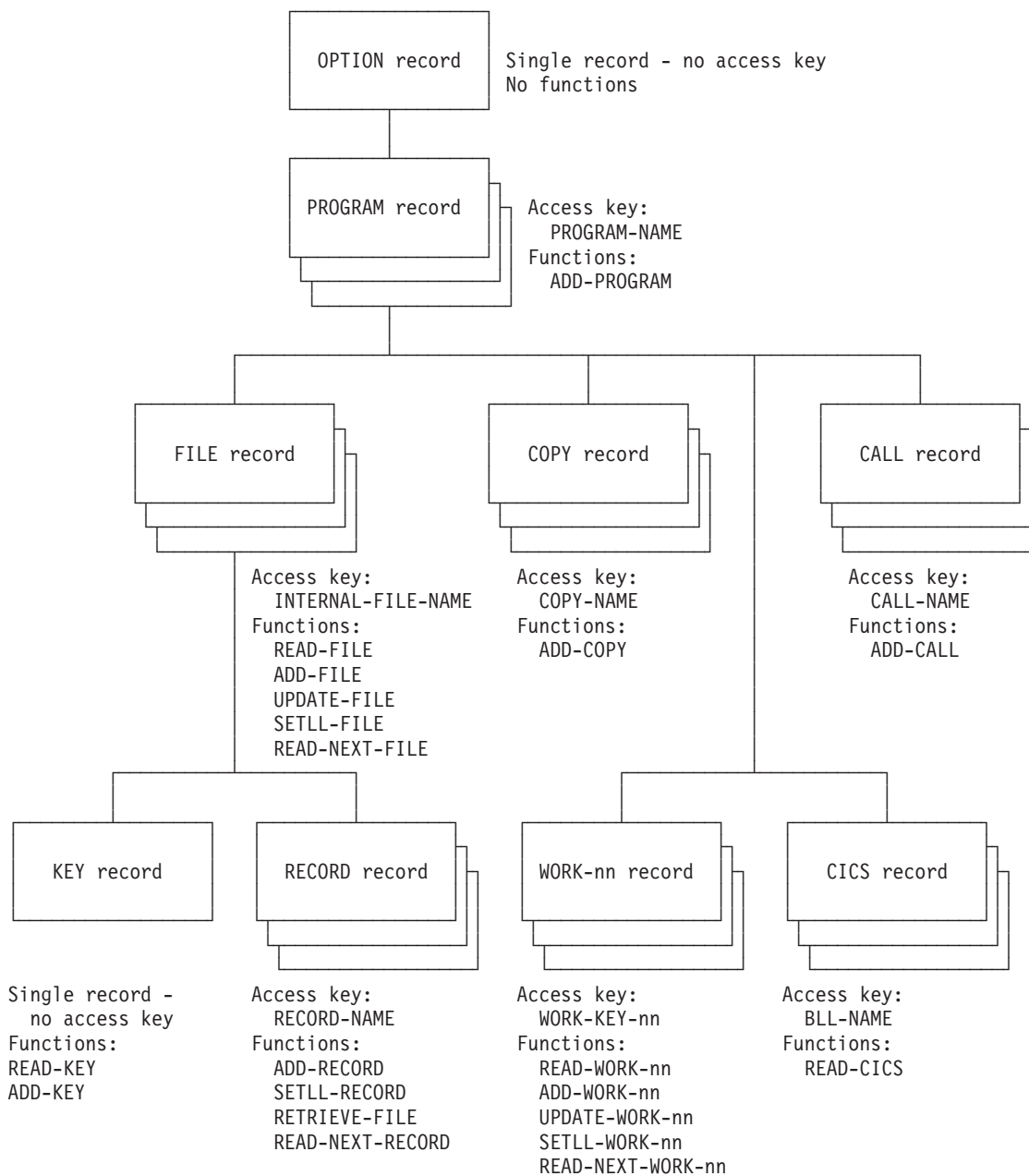


Figure 37. Control and work file record relationships

Control file

Here are the details of the control file.

OPTION record

The OPTION record contains the conversion options, available as these predefined data items:

05 LITERAL-SEPARATOR	PIC X.
05 OPTION-01	PIC X.
05 OPTION-02	PIC X.
05 OPTION-03	PIC X.
05 OPTION-04	PIC X.
05 OPTION-05	PIC X.

```

05 OPTION-06          PIC X.
05 OPTION-07          PIC X.
05 OPTION-08          PIC X.
05 OPTION-09          PIC X.
05 OPTION-10          PIC X.
05 OPTION-11          PIC X.
05 OPTION-12          PIC X.
05 OPTION-13          PIC X.
05 OPTION-14          PIC X.
05 OPTION-15          PIC X.
05 MEMBER-NAME        PIC X(10).
05 COBOL-STANDARD      PIC X(5).
05 TARGET-LANGUAGE     PIC X(5).
05 OPTION-CICS         PIC X.
05 COBOL-TYPE          PIC X(6).
05 DATE-FORMAT         PIC X(8).

```

CCCA gets this information from the details you enter on the Conversion Options panels (see “Setting conversion options” on page 19).

PROGRAM record

Each program CCCA converts has a PROGRAM record. Its access key is the PROGRAM NAME defined in the COBOL program.

The PROGRAM record exists to contain statistics for the Program/File and File/Program reports (for details, see Chapter 5, “Conversion reports and the conversion log,” on page 49). You cannot read this record from an LCP.

LCP function	Description
ADD-PROGRAM	Add a PROGRAM record using the current values in the OPTION record

The PROGRAM record contains these predefined data items:

```

05 PROGRAM-STATUS      PIC X(10).
05 PROGRAM-NAME        PIC X(10). ◀ Access key
05 OLD-PROGRAM-NAME    PIC X(10).

```

The PROGRAM record also contains:

- The conversion options CCCA used to convert each program (in a similar format to the current conversion options stored in the OPTION record)
- A program conversion revision count

The ADD-PROGRAM function:

- Updates (or adds, if no record exists) a PROGRAM record for the PROGRAM-NAME you specify with the current conversion options (from the OPTION record)
- Increments the PROGRAM record revision count by one

FILE record

The FILE record contains information about each file (such as organization and access mode) used in a COBOL program.

There is one FILE record for each each file defined in the COBOL program.

LCP function	Description
READ-FILE	Retrieve a specific FILE record

Developing LCPs

LCP function	Description
UPDATE-FILE	Update a specific FILE record
ADD-FILE	Add a FILE record
SETLL-FILE	Position at the first FILE record
READ-NEXT-FILE	Read the next FILE record

These functions give you access to the following predefined data items:

05 ORGANIZATION-FILE-MODE	PIC X.
05 ACCESS-FILE-MODE	PIC X.
05 SEQUENCE-STATUS-NO	PIC 9(2).
05 EXTERNAL-FILE-NAME	PIC X(10).
05 INTERNAL-FILE-NAME	PIC X(30). ◀ Access key
05 UPDATE-FILE-FLAG	PIC X.
05 ASCII-FILE	PIC X.
05 FILE-CONVERSION	PIC X.
05 OLD-ORGANIZATION-FILE-MODE	PIC X.
05 VSAM-ORGANIZATION	PIC X.

CALL record

There is a CALL record for each CALL statement in the COBOL program.

LCP function	Description
ADD-CALL	Add a CALL record

This function gives you access to the following predefined data item:

05 CALL-NAME	PIC X(30). ◀ Access key
--------------	-------------------------

CCCA uses the CALL file records to generate the Call/Program report and the Program/Call report. Performing the ADD-CALL function results in a CALL record being generated identifying the CALL-NAME provided as a sub-program called by the program being converted.

COPY record

There is a COPY record for each COPY statement in the COBOL program.

LCP function	Description
ADD-COPY	Add a COPY record

This function gives you access to the following predefined data items:

05 COPY-NAME	PIC X(10). ◀ Access key
05 COPY-LOCATION	PIC X(3).
05 ASSOCIATE-NAME	PIC X(30).

CCCA uses the COPY records to generate the Copy/Program and Program/Copy reports. The ADD-COPY function adds a COPY record that identifies the contents of the COPY-NAME predefined data item as a COPY member in the program being converted.

Work file

Here are the details of the work file.

KEY record

When the KEY clause is defined in a file description, a KEY file record is automatically generated, linking the key to the active FILE record.

LCP function	Description
READ-KEY	Retrieve a KEY record for the active file
ADD-KEY	Add or update a KEY record for the active file

These functions give you access to the following predefined data items:

05 NOMINAL-KEY-NAME	PIC X(30).
05 RECORD-KEY-NAME	PIC X(30).
05 RELATIVE-KEY	PIC X(30).
05 FILE-STATUS-NAME	PIC X(30).

RECORD record

Within each file a record name can be defined allowing several records to be linked to the active FILE record.

One record per 01 level data in File Description (FD).

LCP function	Description
ADD-RECORD	Add a RECORD record for the active file
SETLL-RECORD	Position at the beginning of the RECORD file for the active file.
READ-NEXT-RECORD	Read next RECORD record for the active FILE record
RETRIEVE-FILE	Retrieve the FILE record using a RECORD-NAME (first possible file).

These functions give you access to the following predefined data item:

05 RECORD-NAME	PIC X(30). ◀ Access key
----------------	-------------------------

WORK-*nn* records

During LCP execution, you can save conversion information WORK records.

LCP function	Description
READ-WORK- <i>nn</i>	Read a WORK- <i>nn</i> record
UPDATE-WORK- <i>nn</i>	Update a WORK- <i>nn</i> record
ADD-WORK- <i>nn</i>	Add a WORK- <i>nn</i> record
SETLL-WORK- <i>nn</i>	Set to the beginning of the WORK- <i>nn</i> records
READ-NEXT-WORK- <i>nn</i>	Read next WORK- <i>nn</i> record

These functions give you access to the following predefined data items:

05 WORK-KEY- <i>nn</i>	PIC X(30). ◀ Access key
05 WORK-TEXT- <i>nn</i>	PIC X(30).
05 WORK-NUMERIC- <i>nn</i>	PIC 9(7).
05 WORK-TYPE- <i>nn</i>	PIC X(3).
05 WORK-TEXT2- <i>nn</i>	PIC X(30).
05 WORK-NUMERIC2- <i>nn</i>	PIC 9(7).
05 WORK-TYPE2- <i>nn</i>	PIC X(3).

Note:

Developing LCPs

1. *mm* is 01 through 10.
2. WORK records 01 to 03 are available for user-written LCPs.
3. WORK record 04 is also available if MLE conversions are *not* required.
4. WORK records 05 to 10 are reserved for use by CCCA.
5. The supplied DEBUGGING LCP contains an example of how to use WORK records.

CICS record

For each BLL statement defined in the converted COBOL program, there is a corresponding record.

LCP function	Description
READ-CICS	Read the CICS record that is used to relate the BLL to the 01 level data area as mapped in the Linkage Section.

This function gives you access to the following predefined data items:

```
01 CICS-REC.  
   05 BLL-NAME          PIC X(30).  ◀ Access key  
   05 CICS-RECORD-NAME PIC X(30).  (Name of data area pointed to by BLL)
```

Using LCPs

Here is how you use LCPs.

Controlling LCP invocation

The name of the LCP that CCCA invokes to convert a language element is determined by the value of the TOKEN-CHANGE-CODE predefined data item.

During phase 1 (tokenization of the input source program) a TOKEN record is written for every tokenized word in the source program. As each word is tokenized, the word is used as a search argument to search the COBOL Reserved Word data set. If a match is found, the change code in the matching data set entry is stored in the TOKEN-CHANGE-CODE field of the TOKEN record (every word in the COBOL Reserved Word data set has a change code in the range 000 through 999). If no match is found, a value of 999 is stored in the TOKEN-CHANGE-CODE.

During phase 2, CCCA uses the value in the TOKEN-CHANGE-CODE field of each TOKEN record to determine the name of the LCP that is invoked to process the tokenized word. The name of the LCP is determined as follows:

- If the value in the TOKEN-CHANGE-CODE field is 990, the name of the LCP is the same as the tokenized word. For example, in the supplied COBOL Reserved Word data set, the word OTHERWISE has a change code of 990, which indicates that the LCP named OTHERWISE is to be invoked. One of the supplied LCPs is an LCP named OTHERWISE that is used to convert the reserved word OTHERWISE.
- If the value in the TOKEN-CHANGE-CODE field is 999, no LCP is invoked to convert the language element. For example, in the supplied Reserved Word data set, the word ALTERNATE has a change code of 999. That is, no LCP is invoked to convert the ALTERNATE language element.
- If the TOKEN-CHANGE-CODE field has a value other than 990 or 999, the name of the LCP is LCP*mmm*, where *mmm* is the value of the TOKEN-CHANGE-CODE field. For example, in the supplied Reserved Word data set, the reserved

words UPSI-0 through UPSI-7 have a change code of 850. That is, the supplied LCP named LCP850 is invoked to process conversion of these reserved words in a source program. Note that one LCP can be used to convert more than one reserved word.

For details on adding new words to the supplied COBOL Reserved Word file and setting change codes, see “Updating the COBOL reserved word Data Set” on page 66.

Processing LCPs

Figure 38 is an example of source code for the OTHERWISE LCP. During conversion, each time a token with the value OTHERWISE is found within the COBOL source program, the OTHERWISE LCP is executed. The purpose of this LCP is to change the COBOL reserved word OTHERWISE to ELSE.

```
*****00001000
*
*          CONVERA OTHERWISE 'REPLACE OTHERWISE BY ELSE'          *00002000
*
*          REPLACE OTHERWISE BY ELSE                               *00003000
*
*
*
*          *****00004000
*          *****00005000
*          *****00006000
*          *****00007000
*          LICENSED MATERIALS - PROPERTY OF IBM                    00008000
*          5785-ABJ 5785-CCC 5648-B05 5686-A07                     00008800
*          (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. 00009600
*          US GOVERNMENT USERS RESTRICTED RIGHTS - USE,           00010400
*          DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP        00011200
*          SCHEDULE CONTRACT WITH IBM CORP.                        00012100
*
*          OTHER-WISE-010 .
*          IF COBOL-TYPE NOT = 'DOS/VS'                             00013000
*          AND COBOL-TYPE NOT = 'OS/VS'                             00014000
*          GO TO END-CHANGE.                                        00015000
*          IF WHERE-USED IS NOT EQUAL TO 'PR'                       00016000
*          GO TO END-CHANGE.                                        00017000
*          MOVE '04ELSE' TO ADD-GROUP .                             00018000
*          MOVE '04ELSE' TO ADD-GROUP .                             00019000
*          MOVE '04ELSE' TO ADD-GROUP .                             00020000
*          MOVE '04ELSE' TO ADD-GROUP .                             00021000
*          MOVE 'ABJ6021' TO MESSAGE-ID.                            00022000
*          MOVE 'ABJ6021' TO MESSAGE-ID.                            00023000
*          PERFORM REPLACE .                                        00023000
*          PERFORM EDIT-MESSAGE.                                    00023000
*          GO TO END-CHANGE .                                        00023000
```

Figure 38. OTHERWISE LCP source code

When an LCP is compiled, CCCA produces *intermediate text* and a listing. The intermediate text is written to the LCP library (also known as the DRIVEN data set). The listing that is produced can be used during debugging. The statement numbers contained in the listing are the statement numbers referred to in LCP traces.

Figure 39 on page 108 is an example of the OTHERWISE LCP listing produced by the LCP compiler.

Developing LCPs

```
5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE RUN

STMT SEQNBR  A 1 B.. ... 2 ... ...   LCP SOURCE STATEMENTS ... 6 ... ... 7

*****
*
1  *   CONVERA OTHERWISE 'REPLACE OTHERWISE BY ELSE'          *
*
*   REPLACE OTHERWISE BY ELSE                                *
*
*****
*  LICENSED MATERIALS - PROPERTY OF IBM
*  5785-ABJ 5785-CCC 5648-B05 5686-A07
*  (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED.
*  US GOVERNMENT USERS RESTRICTED RIGHTS - USE,
*  DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP
*  SCHEDULE CONTRACT WITH IBM CORP.

2      OTHER-WISE-010 .
3          IF COBOL-TYPE NOT = 'DOS/V5'
4          AND COBOL-TYPE NOT = 'OS/V5'
5          GO TO END-CHANGE.
6          IF WHERE-USED IS NOT EQUAL TO 'PR'
7          GO TO END-CHANGE.
8          MOVE '04ELSE' TO ADD-GROUP .
9          PERFORM REPLACE .
10         MOVE 'ABJ6021' TO MESSAGE-ID.
11         PERFORM EDIT-MESSAGE.
12         GO TO END-CHANGE .

TEXT DESCRIPTION -      REPLACE OTHERWISE BY ELSE
LCP PROGRAM NAME -      OTHERWISE
TABLE DRIVEN CORE SIZE - 290
```

Figure 39. *OTHERWISE* LCP compilation listing

Tokenization

During conversion phase 1 (see “How CCCA works” on page 4), the COBOL source program is analyzed in terms of character strings called tokens. You can print a listing of the tokenization by setting the **Generate tokenization listing** field on Conversion Options panel 1 (see Figure 9 on page 19) to Y.

Figure 40 on page 109 is an example of the tokenization of a COBOL source program.

```

                                SEQ-NO/POS/LNGTH/TYPE/CODE/FLAG 1
002160  EXAMINE FILE-A-RECORD TALLYING ALL ".".                01772202 2
EXAMINE .....:002160 05 007 W 990 03
      FILE-A-RECORD .....:002160 13 013 W 000
            TALLYING .....:002160 27 008 W 000
                  ALL .....:002160 36 003 W 990
                        ". " .....:002160 40 003 L 000
                                . .....:002160 43 001 000
002170  DISPLAY FILE-A-RECORD " COUNT OF " TALLY.          01772300
DISPLAY .....:002170 05 007 W 999 23
      FILE-A-RECORD .....:002170 13 013 W 000
            " COUNT OF " .....:002170 27 013 L 000
                  TALLY .....:002170 41 005 W 000
                        . .....:002170 46 001 000
002180  IF (TALLY = 3)                                     01772402
IF .....:002180 05 002 W 999 03
  ( .....:002180 08 001 000
    TALLY .....:002180 09 005 W 000
      = .....:002180 15 001 1 997
        3 .....:002180 17 001 N 000
          ) .....:002180 18 001 000
002190  AND (FILE-A-RECORD = ".B.D.F")                    01772502
AND .....:002190 07 003 W 000
  ( .....:002190 11 001 000
    FILE-A-RECORD .....:002190 12 013 W 000
      = .....:002190 26 001 1 997
        ".B.D.F" .....:002190 28 008 L 000
          ) .....:002190 36 001 000
002200  THEN DISPLAY " TST-504-A2 WAS SUCCESSFUL"          01772602
THEN .....:002200 09 004 W 990 03
  DISPLAY .....:002200 14 007 W 999 23
    " TST-504-A2 WAS SUCCESSFUL" .....:002200 22 031 L 000
002210  OTHERWISE MOVE "Y" TO CONVERSION-ERROR-SWITCH    01772702
OTHERWISE .....:002210 08 009 W 990
  MOVE .....:002210 18 004 W 851 03
    "Y" .....:002210 23 003 L 000
      TO .....:002210 27 002 W 000
        CONVERSION-ERROR-SWITCH .....:002210 30 023 W 000
002220  DISPLAY " TST-504-A2 WAS UNSUCCESSFUL".            01772802
DISPLAY .....:002220 10 007 W 999 23
  " TST-504-A2 WAS UNSUCCESSFUL" .....:002220 18 033 L 000
      . .....:002220 51 001 000
002230  DISPLAY " END TEST PIR-025-A SUCCESSFUL RUN ".    01772902
DISPLAY .....:002230 05 007 W 999 23
  " END TEST PIR-025-A SUCCESSFUL RUN " .....:002230 13 037 L 000
      . .....:002230 50 001 000
002240  STOP RUN.                                         01773002
STOP .....:002240 05 004 W 999 03
  RUN .....:002240 10 003 W 000
      . .....:002240 13 001 000

```

Figure 40. Section of a tokenized COBOL source program

The main features of the listing are:

- 1 The report headings:
SEQ-NO TOKEN-SEQUENCE
POS TOKEN-POSITION
LNGTH TOKEN-LENGTH
TYPE TOKEN-TYPE-CODE
CODE TOKEN-CHANGE-CODE
FLAG TOKEN-FLAG

See Appendix E, "Predefined data items," on page 175 for a description of these fields.

- 2 Input program source line.
- 3 Tokenized source. There is a record written to the TOKEN data set for each token.

Developing LCPs

Note: CCCA does not tokenize literals which are greater than 30 chars long. The tokenization report reflects this fact with a filler of asterisks as shown in the following example:

```

01 TEST-LINE-2 PIC X(80) VALUE                00311001
01 :::::::::::::::::::::::::::::::::::::::::::: 000410 01 002 N 990
    TEST-LINE-2 :::::::::::::::::::::::::::::::::::: 000410 05 011 W 000
                PIC :::::::::::::::::::::::::::::::::::: 000410 17 003 P 990 02
                  X(80) :::::::::::::::::::::::::::: 000410 21 005 P 000
                    VALUE :::::::::::::::::::: 000410 27 005 W 990 02
>> "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB" . 00312006
>> ***** :::::::::::::::::::: 000420 05 036 L 864 00
      . :::::::::::::::::::: 000420 41 001 000
  
```

Debugging LCPs

To aid in the debugging of LCPs, a facility is provided that will generate trace output for specific LCPs.

To activate debugging for one or more LCPs, use the Delete/Debug LCP panel (see Figure 33 on page 72).

An example of the OTHERWISE LCP trace is shown in Figure 41. The columns of this listing are described below.

```

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN          04/13/98 12:15:24      PAGE 1
1
*CONVER:OTHERWISE          *TEXT:REPLACE OTHERWISE BY ELSE          *DATE:041398
12915
2
3 4 5
LCP LCP ID
STMT OP CODE FILE *... .. 1 ... .. 2 ... .. 3 ... .. 4 ... .. 5 ... .. 6 ... .. 7
OTHERWISE          3 IFEQA
OTHERWISE          4 IFEQA
OTHERWISE          6 IFEQA
OTHERWISE          8 MOVE
OTHERWISE          9 RP
                   CHANGE 002210086 04ELSE          01 Y
                   CHANGE 002210085                    Y
OTHERWISE          10 MOVE
OTHERWISE          11 EDMSG
                   CHANGE 002210083 00OTHERWISE REPLACED BY ELSE          YABJ602100
OTHERWISE          12 GOTO
  
```

Figure 41. Trace of OTHERWISE LCP execution

1 LCP Name

The name of the LCP that is currently in control. This is the program name or LCP-identifier provided on the CONVER statement of the LCP.

2 TOKEN-TEXT

The value of the token currently being processed.

3 LCP Statement Number

The LCP statement number of the LCP being executed. This can be matched to the statement numbers from the compilation listing for the LCP.

4 LCP OP Code

The instruction code of the LCP function associated with the LCP statement number being executed. See Appendix F, "List of LCP functions," on page 187.

5 Logical File and Record

This identifies the record and the record used by the LCP instruction being executed.

The result of OTHERWISE LCP execution is shown in Figure 42.

```

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN PIR025A 04/13/98 12:15:53 PAGE 7
0SEQNBR-A 1 B.. ... 2 ... .. COBOL SOURCE STATEMENTS ... 6 ... .. 7 .IDENTFCN OLD/SQ S MSGID SEV --- D I A G N O S T I C S --
0
024000 MOVE ZERO TO TALLY 017721
*OLD** EXAMINE FILE-A-RECORD TALLYING ALL ". ". 01772202 017722
024100 INSPECT FILE-A-RECORD TALLYING TALLY FOR ALL ". ". 01772202 017722 ABJ6019 00 EXAMINE REPLACED BY INSPECT
024200 DISPLAY FILE-A-RECORD " COUNT OF " TALLY. 01772300 017723
024300 IF (TALLY = 3) 01772402 017724
024400 AND (FILE-A-RECORD = ".B.D.F") 01772502 017725
024500 THEN DISPLAY " TST-504-A2 WAS SUCCESSFUL" 01772602 017726
*OLD** OTHERWISE MOVE "Y" TO CONVERSION-ERROR-SWITCH 01772702 017727
024600 ELSE MOVE "Y" TO CONVERSION-ERROR-SWITCH 01772702 017727 ABJ6021 00 OTHERWISE REPLACED BY ELSE
024700 DISPLAY " TST-504-A2 WAS UNSUCCESSFUL". 01772802 017728
024800 DISPLAY " END TEST PIR-025-A SUCCESSFUL RUN ". 01772902 017729
*OLD** STOP RUN. 01773002 017730
024900 STOP RUN. 01773002 ABJ6126 99 *-----*
025000 END PROGRAM PIR025A. 017730 * END OF COBOL CONVERSION *
* 5648-B05 COBOL CONVERSION*
*-----*

```

Figure 42. Section of the Diagnostic listing showing result of OTHERWISE conversion

Processing differences between tokens and elements

The following section explains the differences that exist between the processing of tokens and elements. Differences exist in the way they are tokenized and how they are retrieved from the TOKEN data set. The differences are shown through the use of an LCP and a sample COBOL program.

Figure 43 on page 112 shows the LCP TKNTTEST.

When TKNTTEST is invoked, it uses the GET-NEXT-ELEMENT and GET-NEXT-TOKEN functions. These two functions show the difference in the way tokens and elements are treated by functions that retrieve records from the TOKEN data set.

Developing LCPs

5648-B05 V2R1

- IBM COBOL CONVERSION AID - SAMPLE RUN

04/14/98 12:40:36

STMT SEQNBR A 1 B.. ... 2 LCP SOURCE STATEMENTS ... 6 7

```
*****
*
1  *   CONVERA  TKNTEST 'SHOW DIFFERENCE BETWEEN TOKEN AND ELEMENT'*
*****
*   TO SHOW THE DIFFERENCES BETWEEN PROCESSING ELEMENTS AND
*   TOKENS, THIS LCP WILL:
*   1. SAVE THE CURRENT TOKEN/ELEMENT POSITION
*   2. READ 20 ELEMENTS
*   3. REPOSITION TO THE SAVED POSITION
*   4. READ 20 TOKENS
*   5. REPOSITION TO THE SAVED POSITION
*   6. EXIT
*****

2      05 SAVE-POINTER      PIC 9(7).
3      SHOW-USAGE.
4      MOVE TOKEN-POINTER  TO SAVE-POINTER.
5      PERFORM GET-NEXT-ELEMENT 10 TIMES.
6      PERFORM GET-NEXT-ELEMENT 10 TIMES.
7      MOVE SAVE-POINTER  TO TOKEN-POINTER.
8      PERFORM GET-TOKEN.
9      PERFORM GET-NEXT-TOKEN 10 TIMES.
10     PERFORM GET-NEXT-TOKEN 10 TIMES.
11     MOVE SAVE-POINTER  TO TOKEN-POINTER.
12     GO TO END-CHANGE.
TEXT DESCRIPTION -      SHOW DIFFERENCE BETWEEN TOKEN AND ELEMENT
LCP PROGRAM NAME -      TKNTEST
TABLE DRIVEN CORE SIZE - 385
```

Figure 43. TKNTEST LCP compilation listing

Figure 44 shows the COBOL program SAMPLPRG which is written to show the differences that exist between the processing of tokens and elements.

In the program, the word TKNTEST is used to show how tokenization affects the invoking of LCPs. The program also provides examples of the two types of element.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLPRG.
DATE-WRITTEN. 05/05/1998
    Uses the LCP TKNTEST to show the difference between 1
    TOKENS and ELEMENTS.

ENVIRONMENT DIVISION.
SKIP2
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
77 PRG-NAME      PIC X(10)  VALUE 'SAMPLPRG'.
77 TKNTEST      PIC X(10).  2
77 PRG-NAME1    PIC X(10)  VALUE SPACES.
COPY TSTMEMBR  REPLACING TEMP-FLD BY TKNTEST. 3
EJECT
PROCEDURE DIVISION.
* comments are not tokenized
START-HERE.
    IF PRG-NAME1      = SPACES
        MOVE PRG-NAME TO PRG-NAME1.
    DISPLAY 'TEST COMPLETE '.
    STOP RUN.
```

Figure 44. Source of program to be converted

The word TKNTEST occurs three times in the program SAMPLPRG:

- 1** In a comment paragraph
- 2** In a data item description entry
- 3** in a COBOL COPY statement

As you will see, not all occurrences of the word TKNTEST result in the LCP TKNTEST being invoked.

Figure 45 is the tokenized source of the program SAMPLPRG and shows token and element tokenization.

	SEQ-NO/POS/LNGTH/TYPE/CODE/FLAG
IDENTIFICATION DIVISION.	
IDENTIFICATION :::::::::::::::::::::	000010 01 014 W 990 01
DIVISION :::::::::::::::::::::	000010 16 008 W 990
. :::::::::::::::::::::	000010 24 001 000
PROGRAM-ID. SAMPLPRG.	
PROGRAM-ID :::::::::::::::::::::	000020 01 010 W 990 01
. :::::::::::::::::::::	000020 11 001 000
SAMPLPRG :::::::::::::::::::::	000020 14 008 W 000
. :::::::::::::::::::::	000020 22 001 000
DATE-WRITTEN. 05/05/1998	
DATE-WRITTEN :::::::::::::::::::::	000030 01 012 W 856 1
. :::::::::::::::::::::	000030 13 053 * 000
Uses the LCP TKNTEST to show the difference between	
. :::::::::::::::::::::	000040 01 065 * 000 1
TOKENS and ELEMENTS.	
. :::::::::::::::::::::	000050 01 065 * 000 1
ENVIRONMENT DIVISION.	
ENVIRONMENT :::::::::::::::::::::	000070 01 011 W 990 01
DIVISION :::::::::::::::::::::	000070 13 008 W 990
. :::::::::::::::::::::	000070 21 001 000
SKIP2	
DATA DIVISION.	
DATA :::::::::::::::::::::	000090 01 004 W 999 21
DIVISION :::::::::::::::::::::	000090 06 008 W 990
. :::::::::::::::::::::	000090 14 001 000
FILE SECTION.	
FILE :::::::::::::::::::::	000100 01 004 W 999 01
SECTION :::::::::::::::::::::	000100 06 007 W 990
. :::::::::::::::::::::	000100 13 001 000
WORKING-STORAGE SECTION.	
WORKING-STORAGE :::::::::::::::::::::	000110 01 015 W 990 01
SECTION :::::::::::::::::::::	000110 17 007 W 990
. :::::::::::::::::::::	000110 24 001 000
77 PRG-NAME PIC X(10) VALUE 'SAMPLPRG'.	
77 :::::::::::::::::::::	000120 01 002 N 990
PRG-NAME :::::::::::::::::::::	000120 05 008 W 000
PIC :::::::::::::::::::::	000120 24 003 P 990 02
X(10) :::::::::::::::::::::	000120 28 005 P 000
VALUE :::::::::::::::::::::	000120 36 005 W 990 02
'SAMPLPRG' :::::::::::::::::::::	000120 42 010 L 864 00
. :::::::::::::::::::::	000120 52 001 000
77 TKNTEST PIC X(10).	
77 :::::::::::::::::::::	000130 01 002 N 990
TKNTEST :::::::::::::::::::::	000130 05 007 W 000 3
PIC :::::::::::::::::::::	000130 24 003 P 990 02
X(10) :::::::::::::::::::::	000130 28 005 P 000
. :::::::::::::::::::::	000130 33 001 000
77 PRG-NAME1 PIC X(10) VALUE SPACES.	
77 :::::::::::::::::::::	000140 01 002 N 990
PRG-NAME1 :::::::::::::::::::::	000140 05 009 W 000
PIC :::::::::::::::::::::	000140 24 003 P 990 02
X(10) :::::::::::::::::::::	000140 28 005 P 000
VALUE :::::::::::::::::::::	000140 36 005 W 990 02
SPACES :::::::::::::::::::::	000140 42 006 W 999
. :::::::::::::::::::::	000140 48 001 000

Figure 45. Tokenization of the COBOL source program containing tokens and elements (Part 1 of 2)

Developing LCPs

	SEQ-NO/POS/LNGTH/TYPER/CODE/FLAG
COPY TSTMEMBR REPLACING TEMP-FLD BY TKNTTEST.	
COPY ::::::::::::::::::::::::::::::::::::::	000150 01 004 C 995 03 4
TSTMEMBR ::::::::::::::::::::::::::::::	000150 06 008 C 000 4
REPLACING ::::::::::::::::::::::::::::	000150 16 009 C 000 4
TEMP-FLD ::::::::::::::::::::::::::::	000150 26 008 C 000 4
BY ::::::::::::::::::::::::::::	000150 35 002 C 000 4
TKNTTEST ::::::::::::::::::::::::::::	000150 38 007 C 000 4
. ::::::::::::::::::::::::::::	000150 45 001 C 000 4
EJECT	2
PROCEDURE DIVISION.	
PROCEDURE ::::::::::::::::::::::::::::	000170 01 009 W 990 01
DIVISION ::::::::::::::::::::::::::::	000170 11 008 W 990
. ::::::::::::::::::::::::::::	000170 19 001 000
* comments are not tokenized	5
START-HERE.	
START-HERE ::::::::::::::::::::::::::::	000190 01 010 W 860 01
. ::::::::::::::::::::::::::::	000190 11 001 000
IF PRG-NAME1 = SPACES	
IF ::::::::::::::::::::::::::::	000200 05 002 W 999 03
PRG-NAME1 ::::::::::::::::::::::::::::	000200 08 009 W 000
= ::::::::::::::::::::::::::::	000200 24 001 W 997 00
MOVE PRG-NAME TO PRG-NAME1.	
SPACES ::::::::::::::::::::::::::::	000200 26 006 W 999
MOVE ::::::::::::::::::::::::::::	000210 09 004 W 851 03
PRG-NAME ::::::::::::::::::::::::::::	000210 14 008 W 000
TO ::::::::::::::::::::::::::::	000210 24 002 W 999
PRG-NAME1 ::::::::::::::::::::::::::::	000210 27 009 W 000
. ::::::::::::::::::::::::::::	000210 36 001 000
DISPLAY 'TEST COMPLETE '.	
DISPLAY ::::::::::::::::::::::::::::	000220 05 007 W 990 23
'TEST COMPLETE ' ::::::::::::::::::::	000220 13 016 L 864 00
. ::::::::::::::::::::::::::::	000220 29 001 000
STOP RUN.	
STOP ::::::::::::::::::::::::::::	000230 05 004 W 990 03
RUN ::::::::::::::::::::::::::::	000230 10 003 W 999
. ::::::::::::::::::::::::::::	000230 13 001 000

Figure 46. Tokenization of the COBOL source program containing tokens and elements (Part 2 of 2)

- 1** The comment paragraph line is treated as a single element. If the conversion option *Remove obsolete elements* (see “Setting conversion options” on page 19) is set to Y, these lines are commented out.
- 2** SKIP n and EJECT compiler directives are not tokenized.
- 3** Example of a token.
- 4** The COPY statement is analyzed into elements.
- 5** Comment lines are not tokenized.

Once tokenized, tokens and elements are identified by their TOKEN-TYPE-CODE value. See Appendix F, “List of LCP functions,” on page 187.

During conversion the LCP TKNTTEST will be invoked by:

- 3** The token TKNTTEST contained in the data item definition

The LCP is **not** invoked by:

- 1** The TKNTTEST in the comment paragraph.
- 4** The TKNTTEST in the COPY statement.

Figure 47 on page 115 shows the trace of LCP TKNTTEST generated during the conversion of the program SAMPLPRG.

CODE- TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... .. 1 2 3 4 5 6 7
TKNTEST	4	MOVE	1	
TKNTEST	5	GTNXX		
PIC	5	GTNXX	TOKEN 00013024003P990 02PIC	YP
x(10)	5	GTNXX	TOKEN 00013028005P000 00x(10)	YP
.	5	GTNXX	TOKEN 00013033001 000 00.	NP
77	5	GTNXX	TOKEN 00014001002N000 0077	NP
PRG-NAME1	5	GTNXX	TOKEN 00014005009W000 00PRG-NAME1	YP
PIC	5	GTNXX	TOKEN 00014024003P990 02PIC	YP
x(10)	5	GTNXX	TOKEN 00014028005P000 00x(10)	YP
VALUE	5	GTNXX	TOKEN 00014036005W990 02VALUE	YP
SPACES	5	GTNXX	TOKEN 00014042006W000 00SPACES	YP
.	6	GTNXX	TOKEN 00014048001 000 00.	NP
COPY	6	GTNXX	TOKEN 00015001004C995 03COPY	NP 2
TSTMEMBR	6	GTNXX	TOKEN 00015006008C000 00TSTMEMBR	YP 2
REPLACING	6	GTNXX	TOKEN 00015016009C999 02REPLACING	YP 2
TEMP-FLD	6	GTNXX	TOKEN 00015026008C000 00TEMP-FLD	YP 2
BY	6	GTNXX	TOKEN 00015035002C000 00BY	YP 2
TKNTEST	6	GTNXX	TOKEN 00015038007C990 TKNTEST	YP 2
.	6	GTNXX	TOKEN 00015045001C000 00.	NP 2
01	6	GTNXX	TOKEN 00016001002N990 01	NC
TEMP-LINE	6	GTNXX	TOKEN 00016005009W000 00TEMP-LINE	YC
.	7	MOVE		
.	8	GTTKN	3	
TKNTEST	9	GTNXT	TOKEN 00013005007W990 TKNTEST	YP
PIC	9	GTNXT	TOKEN 00013024003P990 02PIC	YP
x(10)	9	GTNXT	TOKEN 00013028005P000 00x(10)	YP
.	9	GTNXT	TOKEN 00013033001 000 00.	NP
77	9	GTNXT	TOKEN 00014001002N000 0077	NP
PRG-NAME1	9	GTNXT	TOKEN 00014005009W000 00PRG-NAME1	YP
PIC	9	GTNXT	TOKEN 00014024003P990 02PIC	YP
x(10)	9	GTNXT	TOKEN 00014028005P000 00x(10)	YP
VALUE	9	GTNXT	TOKEN 00014036005W990 02VALUE	YP
SPACES	9	GTNXT	TOKEN 00014042006W000 00SPACES	YP
			TOKEN 00014048001 000 00.	NP

Figure 47. Trace of TKNTEST LCP execution (Part 1 of 2)

Developing LCPs

CODE- TOKEN-TEXT	LCP	LCP	ID	STMT	OPCODE	FILE	*... .. 1 2 3 4 5 6 7
.	10	GTNXT	4	TOKEN	00015001004C995	03COPY	NP
				TOKEN	00015006008C000	00TSTMEMBR	YP
				TOKEN	00015016009C999	02REPLACING	YP
				TOKEN	00015026008C000	00TEMP-FLD	YP
				TOKEN	00015035002C000	00BY	YP
				TOKEN	00015038007C990	TKNTEST	YP
				TOKEN	00015045001C000	00.	NP
				TOKEN	00016001002N990	01	NC
01	10	GTNXT		TOKEN	00016005009W000	00TEMP-LINE	YC
TEMP-LINE	10	GTNXT		TOKEN	00016014001	000 00.	NC
.	10	GTNXT		TOKEN	00017005002N000	0005	YC
05	10	GTNXT		TOKEN	00017009006W000	00FILLER	YC
FILLER	10	GTNXT		TOKEN	00017033003P990	02PIC	YC
PIC	10	GTNXT		TOKEN	00017037005P000	00X(30)	YC
X(30)	10	GTNXT		TOKEN	00017042001	000 00.	NC
.	10	GTNXT		TOKEN	00018005002N000	0005	YC
05	10	GTNXT		TOKEN	00018009008W000	00TEMP-FLD	YC
TEMP-FLD	11	MOVE					
TEMP-FLD	12	GOTO					

Figure 48. Trace of TKNTEST LCP execution (Part 2 of 2)

Items of interest in the trace listing:

1 Entry into LCP

The first invocation of the LCP. Triggered by the token TKNTEST in the data item definition.

2 COBOL COPY elements

The elements of the COPY statement are retrieved by the GET-NEXT-ELEMENT function.

3 Reposition TOKEN data set

Repositioning of the TOKEN data set after performing the function GET-NEXT-ELEMENT 20 times. This is done to be able to show the difference when the same TOKEN data set records are read using the function GET-NEXT-TOKEN.

4 GET-NEXT-TOKEN

GET-NEXT-TOKEN function called to retrieve the next TOKEN.

5 COBOL COPY elements

The elements that constitute the COPY statement are bypassed by the GET-NEXT-TOKEN function.

Appendix A. Converted COBOL language elements

Table 6 describes the language elements converted, flagged, or removed by CCCA.

The columns of this table are described below.

Language element

The language element in the input source program.

Conversion status

The status of the language element after the program is converted by CCCA:

C	Converted
R	Removed
F	Flagged
I	Information

Language level

The source language level(s) for which the conversion and/or flagging is performed:

1	DOS/VS COBOL—LANGLVL(1) (COBOL 68 Standard)
2	DOS/VS COBOL—LANGLVL(2) (COBOL 74 Standard)
3	OS/VS COBOL—LANGLVL(1) (COBOL 68 Standard)
4	OS/VS COBOL—LANGLVL(2) (COBOL 74 Standard)
5	VS COBOL II (COBOL 74 Standard) Release 1.0, Release 1.1, or Release 2.0 (or any COBOL with the CMPR2 option)
6	VS COBOL II—NOCMPR2 (COBOL 85 Standard) Release 3.0, Release 3.1, or Release 3.2
7	VS COBOL II—NOCMPR2 (COBOL 85 Standard) Release 4.0
8	COBOL/370—NOCMPR2 (COBOL 85 Standard)
9	COBOL for VSE/ESA—NOCMPR2 (COBOL 85 Standard)
10	COBOL for MVS & VM—NOCMPR2 (COBOL 85 Standard)
11	COBOL for OS/390 & VM—NOCMPR2 (COBOL 85 Standard)
12	Enterprise COBOL (prior to Version 5)

Table 6. Language elements converted to specified target language

Language element	Conversion status	Language level	Notes
ACCEPT MESSAGE COUNT statement Communication feature	F	3,4	This is a Communication statement. The Communication module is not supported by the target languages and there is nothing with which it can be replaced.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
ACTUAL KEY clause	C F	1-4	The ACTUAL KEY clause is replaced by the RELATIVE KEY clause. The clause is used for BDAM files. You should convert the file to which the clause refers, to VSAM/RRDS. If the new file organization is not relative (ORGANIZATION clause), the RELATIVE clause is flagged as being incompatible with the new file organization.
ALPHABET clause	C	1-5	The keyword ALPHABET is added in front of the alphabet name within the ALPHABET clause of the SPECIAL-NAMES paragraph.
ALPHABETIC class	C	1-5	ALPHABETIC is changed to ALPHABETIC-UPPER.
APPLY CORE-INDEX clause	R	1-4	This is an ISAM file handling clause. The clause is removed from the I-O-CONTROL paragraph.
APPLY CYL-INDEX clause	R	1,2	The clause is removed from the I-O-CONTROL paragraph.
APPLY CYL-OVERFLOW clause	R	1,2	The clause is removed from the I-O-CONTROL paragraph.
APPLY EXTENDED-SEARCH clause	R	1,2	The clause is removed from the I-O-CONTROL paragraph.
APPLY MASTER-INDEX clause	R	1,2	The clause is removed from the I-O-CONTROL paragraph.
APPLY RECORD-OVERFLOW clause	R	3,4	The clause is removed from the I-O-CONTROL paragraph.
APPLY REORG-CRITERIA clause	R	3,4	This is an ISAM file handling clause. The clause is removed from the I-O-CONTROL paragraph.
APPLY WRITE-VERIFY clause	R	1,2	The clause is removed from the I-O-CONTROL paragraph.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
ASSIGN clause organization parameter	C F	1-4	<p>The assignment name is modified under the following conditions:</p> <ul style="list-style-type: none"> If the target language level is 2 (COBOL for VSE/ESA) and the device type is "UR", the assignment name is set to the system logical device (SYSnnn). For example, <code>SYSnnn-UR-device-S-<NAME></code> is changed to <code>SYSnnn</code> If the device class is not "UR" and the external file name is missing (only applies to DOS/VS programs), then the system logical device (SYSnnn) is added as the external file name. For example, <code>SYSnnn-UT-device-C-<-nn></code> is changed to <code>SYSnnn-UT-device-C-<-nn>-SYSnnn</code> <p>Files that have an organizational parameter equal to D, W, A, U, or R should be converted to VSAM/RRDS. An ORGANIZATION IS RELATIVE clause and a FILE STATUS IS LCP-STATUS-<i>nn</i> clause is added to the SELECT entries.</p> <p>Files that have an organizational parameter equal to I should be converted to VSAM/KSDS. An ORGANIZATION IS INDEXED clause and a FILE STATUS IS LCP-STATUS-<i>nn</i> clause is added to the SELECT entries.</p> <p>When the target language is COBOL for VSE/ESA, if the file is a tape device and both the programmer logical device (SYSnnn) and an external file name are included in the file assignment name, CCCA displays message ABJ6027.</p>
ASSIGN integer system-name	C	1-4	The integer is removed from the clause.
ASSIGN...OR	C	1-4	The OR is removed.
AUTHOR paragraph	C	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the AUTHOR paragraph in the Identification Division is commented out.
BDAM files	C F	1-4	<p>The target languages do not support the processing of BDAM files.</p> <p>You should convert BDAM files into VSAM/RRDS files. CCCA converts the file definitions but you must add the key algorithms manually.</p> <p>See also in this table the other BDAM file processing language elements:</p> <ul style="list-style-type: none"> ACTUAL KEY clause. APPLY RECORD-OVERFLOW clause. SEEK statement. TRACK-LIMIT clause.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
BLANK WHEN ZERO clause	R	1-4	If the data description entry has a BLANK WHEN ZERO clause and a PICTURE string with an * (zero suppression) symbol in it, the BLANK WHEN ZERO clause is removed.
BLOCK CONTAINS clause	R	1-11	The concept of blocking has no meaning for VSAM files. If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the clause will be removed from VSAM file descriptions.
CALL statement	C	1,2	If the program-name in the CALL statement is not enclosed in quotation marks or apostrophes, then quotation marks (if the literal delimiter is the quotation mark) or apostrophes (if the literal delimiter is the apostrophe) are placed around the program-name.
CALL identifier statement	F	3-5	The statement is flagged if the identifier has a PICTURE string consisting of A's and B's only. The COBOL 74 Standard classes these fields as alphabetic, whilst the COBOL 85 Standard classes them as alphanumeric-edited. You will have to make a change to the program as alphanumeric-edited identifiers are not permitted in the CALL statement.
CALL...ON OVERFLOW statement	F	1-5	Under the COBOL 85 Standard the ON OVERFLOW phrase executes under more conditions than it does under the COBOL 68 and COBOL 74 Standards.
	F	1-7	The ON OVERFLOW phrase in a DOS/VS COBOL, OS/VS COBOL or VS COBOL II program is not invoked, if the program is running under CICS. When an overflow condition occurs in a COBOL/VSE program running under CICS, the ON OVERFLOW phrase will be invoked, if it is specified. The statement is flagged if the target language is not VS COBOL II.
CALL...ON EXCEPTION statement	F	6,7	The ON EXCEPTION phrase in a VS COBOL II program is not invoked, if the program is running under CICS. When an exception condition occurs in a COBOL/VSE program running under CICS, the ON EXCEPTION phrase will be invoked, if it is specified. The statement is flagged if the target language is not VS COBOL II.
CALL...USING statement	F	1-4	If identifiers following USING are VSAM file names then the statement is flagged. If identifiers following USING are procedure names and the Check procedure names option on Conversion Options panel 2 is set to Y, then the statement is flagged.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
CANCEL statement	F	3-5	The statement is flagged if there is an identifier in the statement with a PICTURE string consisting of A's and B's only. The COBOL 74 Standard classes these fields as alphabetic, whilst the COBOL 85 standard classes them as alphanumeric-edited. You will have to make a change to the program as alphanumeric-edited identifiers are not permitted in the CANCEL statement.
CBL statement	C F	1-4	The following options are obsolete and are replaced with new compile options: BUF, CLIST, DMAP, CATALR, LINECNT, LOAD, PMAP, SYST, SYSx, STATE, SYNTAX, CSYNTAX, SUPMAP, SXREF, VBSUM.
	R F	1-4	The following options are removed: BATCH, COUNT, ENDJOB, FLOW, LANGLVL1/2, SYMDMP, CDECK, FDECK, LCOL1/2, LSTONLY, LSTCOMP, L120, L132, OSDECK. The following option is removed if the target language is not COBOL II: RESIDENT.
	C F	5-7	The following option is replaced if the target language is not COBOL II: FDUMP.
	R F	5-7	The following option is removed if the target language is not COBOL II: RESIDENT.
			5-11
CLOSE...WITH DISP CLOSE...WITH POSITIONING statements	R	3,4	The WITH DISP phrase and the WITH POSITIONING phrase are removed.
CLOSE...REEL/UNIT FOR REMOVAL statement	F	3,4	CLOSE...REEL/UNIT FOR REMOVAL statements are flagged because in the target languages the FOR REMOVAL option is treated as a comment.
COM-REG special register	F	1,2	The COM-REG special register is not supported by the target languages. You should remove all references to it from the program.
COMMUNICATION SECTION	F	3,4	The Communication module is not supported by the target languages and there is nothing with which it can be replaced. See also in this table the other Communication module language elements: ACCEPT MESSAGE COUNT statement. DISABLE statement. ENABLE statement. RECEIVE statement. SEND statement.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
CONFIGURATION SECTION header	C F	1-4	The CONFIGURATION SECTION header is added, if it is missing and a SOURCE-COMPUTER, an OBJECT-COMPUTER, or a SPECIAL-NAMES paragraph is present. If the CONFIGURATION SECTION header is coded out of sequence, then attempts are made to put it in its correct place. If this cannot be done, then the CONFIGURATION SECTION header is flagged.
COPY statement	C	1,3	<p>COPY statements with associated names are not supported by the target languages.</p> <p>The following example shows how these COPY statements are converted:</p> <pre>01 RECORD1 COPY MBR-A.</pre> <p>Copy member (MBR-A) before and after conversion:</p> <pre>01 RECORD-A. 05 FIELD-A... 05 FIELD-B...</pre> <p>Statement after conversion:</p> <pre>01 RECORD1 COPY MBR-A REPLACING ==01 RECORD-A.== BY == ==.</pre>
		1-5	<p>Under the COBOL 68 and COBOL 74 Standard, National extension characters @, # and \$ are allowed in the text-name and library-name. The COBOL 85 Standard allows these characters in the text-name and library-name, if they are in the form of a nonnumeric literal.</p> <p>If the text-name or library-name contains these National characters and is not in the form of a numeric literal, CCCA encloses the name in quotation marks or apostrophes.</p>

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
COPY...REPLACING statement	F	1-5	<p>If there are lowercase alphabetic characters in operands of the REPLACING phrase, that are not in nonnumeric literals, the statement is flagged. Under the COBOL 68 and COBOL 74 standards the REPLACING phrase is case sensitive. Under the COBOL 85 standard, lowercase characters are treated as their uppercase equivalent. You should check to see if this change will result in different text being copied into your program.</p> <p>If the operands of the REPLACING phrase contain a colon (:) character, that is not in a nonnumeric literal, the statement is flagged. Under the COBOL 68 and COBOL 74 Standards the colon (:) is a non-COBOL character. Under the COBOL 85 Standard the colon character is treated as a separator. You should check to see if this change will result in different text being copied into your program.</p> <p>If the operands of the REPLACING phrase contain an COBOL 85 Standard non-COBOL character that is not in a nonnumeric literal, the statement is flagged. Under the COBOL 68 and COBOL 74 Standards non-COBOL characters are permitted in the REPLACING option. Under the COBOL 85 standard non-COBOL characters in the REPLACING phrase are diagnosed. You should remove all non-COBOL characters from the REPLACING phrase and from the copy book.</p>
CURRENCY SIGN clause	F	1,3	<p>The target languages do not accept the / (slash) character or the = (equal) character in the CURRENCY SIGN clause.</p>
CURRENT-DATE special register	C	1-4	<p>The CURRENT-DATE register is not supported by the target languages. Wherever CURRENT-DATE is referenced in the program, it is replaced by code that obtains the date from the system and puts it in the format of the CURRENT-DATE register. The fields required for the reformatting are generated in the WORKING-STORAGE section.</p> <p>For CICS programs converting to VS COBOL II, the date is retrieved from the system using an EXEC CICS ASKTIME statement. (CICS Release 1.7 or later is required.)</p> <p>For non-CICS programs converting to VS COBOL II, the ACCEPT...FROM DATE statement is used to obtain the date.</p> <p>For programs converting to a non-VS COBOL II level, the Intrinsic Function CURRENT-DATE is used to obtain the date. The fields required for reformatting are generated in the WORKING-STORAGE SECTION.</p> <p>For DOS/VS COBOL, there are two different formats for the CURRENT-DATE register. You must specify in the VSE system date format field on Conversion Options panel 1, the date format that is used at your installation. If you specify the wrong one CCCA will not convert this language element correctly.</p>

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
DATA RECORDS clause	R	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the DATA RECORDS clause is removed from the FD entry. The word RECORDS is added if missing when the clause is not removed.
DATE COMPILED/ DATE WRITTEN headers	C	1-4	If the hyphen after DATE is missing, it is added.
DATE-COMPILED/ DATE-WRITTEN paragraphs	C	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, these paragraphs in the Identification Division are commented out.
DATE-COMPILED header	C	1-4	If you specify N for the Remove obsolete elements option on Conversion Options panel 2 and there is no period after the header, a period is added.
DEBUG card and packet	R	1-4	These are commented out.
DISABLE statement Communication feature	F	3,4	This is a Communication statement. The Communication module is not supported by the target languages and there is nothing with which it can be replaced.
DIVIDE...ON SIZE ERROR statement	F	1-5	DIVIDE...ON SIZE ERROR statements with multiple receiving fields are flagged because the ON SIZE ERROR phrase will not be executed for intermediate results under the COBOL 85 Standard.
ENABLE statement Communication feature	F	3,4	This is a Communication statement. The Communication module is not supported by the target languages and there is nothing with which it can be replaced.
ENTER statement	R	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the ENTER statement is removed.
ERROR declaratives	C	1-4	An ERROR declarative SECTION is generated for each file that is to be converted to VSAM, as long as there does not exist a global file declarative (such as INPUT, OUTPUT, I-O, EXTEND) or a declarative for the file in question. The code in the SECTION includes a DISPLAY of the returned file status and a GOBACK.
ERROR declaratives GIVING option	R I	1-4	The GIVING option is removed from the program.
EXAMINE	C	1-4	The EXAMINE statement is replaced by an INSPECT statement and the statement MOVE ZERO TO TALLY is added in front of it.
EXHIBIT statement	C	1-4	The EXHIBIT statement is replaced by a DISPLAY statement.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
EXIT PROGRAM statement GOBACK statement STOP RUN statement	C	1-5	<p>Under the COBOL 85 Standard, control can not flow beyond the last line of a called subprogram. The compiler generates an implicit EXIT PROGRAM at the end of each program.</p> <p>Under the COBOL 68 and COBOL 74 Standard control can flow beyond the last line of a called program. When this happens the program ABENDs.</p> <p>The COBOL 68 and COBOL 74 Standard behavior can be preserved under the COBOL 85 Standard by adding, at the end of the program, a section with a call to an abend module.</p> <p>If you specify Y for the Negate implicit EXIT PROGRAM option on Conversion Options panel 2, and EXIT PROGRAM, STOP RUN, or GOBACK is not the last physical statement in the program, a section will be added to the end of the program.</p> <p>If the program being converted is a batch program, the section will include a CALL to one of the following modules:</p> <ul style="list-style-type: none"> • ILBOABN0—if you are converting to VS/COBOL II • CEE5ABD—if you are converting to COBOL for VSE/ESA • CEE3ABD—if you are converting to COBOL for MVS & VM or COBOL for OS/390 & VM <p>If the program being converted is a CICS program, the section will include an EXEC CICS ABEND('CCCA') statement.</p>
FILE-LIMIT/ FILE-LIMITS clauses	R	1-4	The clause is removed from the FILE-CONTROL paragraph.
FILE STATUS clause	C	1-4	<p>A FILE STATUS clause:</p> <pre>FILE-STATUS IS LCP-FILE-STATUS-nn</pre> <p>is added to the FILE-CONTROL paragraph for each file that is to be converted to VSAM. The status key data item LCP-FILE-STATUS-nn referred to in the clause is added to the WORKING-STORAGE section. nn is a sequence number.</p>
FILE STATUS codes	F	1-5	<p>The file status codes returned under the COBOL 85 Standard are different from those returned under the COBOL 68 and COBOL 74 Standard.</p> <p>You should check all references to the file status key in the program and update the values of the file status codes where it is required.</p>
GOBACK statement	C	1-5	See the EXIT PROGRAM statement entry in this table.
GREATER THEN relational operator	C	1-4	THEN is changed to THAN.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
IF statement	C F	1-4	<p>Brackets immediately prior to relational operators are moved, but you should inspect the conversion.</p> <p>For example: IF A (= B)</p> <p>is converted to: IF (A = B)</p> <p>The target languages do not accept the following statements: IF dataname ZEROS... IF dataname ZEROES...</p> <p>They are converted to: IF dataname zero...</p> <p>Superfluous IFs are removed.</p>
Indexes (qualified)	F	3,4	<p>Qualified indexes are no longer permitted. Any reference to one will be flagged.</p>
INITIALIZE...REPLACING ALPHABETIC/ALPHANUMERIC-EDITED statement	F	5	<p>The statement is flagged if there are receiving fields with PICTURE strings that consist of A's and B's only. The COBOL 74 Standard classes these fields as alphabetic, whilst the COBOL 85 Standard classes them as alphanumeric-edited.</p> <p>In most cases you will have to change this statement if you want it to exhibit the same behavior as before.</p>
INSPECT statement	F	3-5	<p>The statement is flagged if the PROGRAM COLLATING SEQUENCE established in the OBJECT COMPUTER paragraph identifies an alphabet that was defined with the ALSO clause.</p> <p>Under these circumstances the statement will behave differently under the COBOL 85 Standard.</p>
INSTALLATION paragraph	C	1-11	<p>If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the INSTALLATION paragraph in the Identification Division is commented out.</p>
ISAM files	C	1-4	<p>The target languages do not support the processing of ISAM files.</p> <p>You should convert ISAM files into VSAM/KSDS files. CCCA will convert the file definition and I/O statements for ISAM files.</p> <p>See also in this table the other ISAM file processing language elements: APPLY CORE-INDEX clause. APPLY REORG-CRITERIA clause NOMINAL KEY clause. TRACK AREA clause. START...USING KEY statement</p>

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
JUSTIFIED JUST RIGHT JUSTIFIED RIGHT clause	C F	1,3	<p>Under the COBOL 68 Standard, if a JUSTIFIED clause is specified together with a VALUE clause for a data description entry, the initial data is right justified. Under the COBOL 85 Standard the initial data is not right justified.</p> <p>To preserve the COBOL 68 Standard behavior of this language element, CCCA makes the following conversion.</p> <p>If the length of the nonnumeric literal in the VALUE clause is less than the length of the field as specified in the PICTURE clause, spaces are added to the front of the literal string until there lengths are equal.</p> <p>The clause will be flagged, instead of converted, if the literal has more than 28 characters.</p>
LABEL RECORDS clause	R	1-11	<p>If you specify Y for the Remove obsolete elements option on the Optional Processing Panel, this clause is removed.</p> <p>The word RECORDS is added, if missing, when the clause is not removed.</p>
LABEL RECORDS... TOTALING/TOTALED AREA option	R I	1-4	This option is removed from the program. The data-name associated with this option is listed at the end of the diagnostic listing.
LESS THEN relational operator	C	1-4	THEN is changed to THAN.
Literals - Nonnumeric	C F	1-4	<p>If the continuation of a nonnumeric literal begins in Area A, it is shifted to the right until its whole length lies within Area B.</p> <p>If the continuation is too long to fit in Area B, it is flagged.</p> <p>If the continuation does not start with a delimiter, then one is added.</p>
MEMORY SIZE clause	R	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the MEMORY SIZE clause of the OBJECT-COMPUTER paragraph is removed.
MOVE statement	R	1-4	Superfluous TOs are removed.
MOVE ALL literal	F	1,3	MOVE ALL literal TO numeric will be flagged with a warning.
MOVE CORR/ CORRESPONDING statement	C	1-4	<p>The target languages do not allow multiple receiving fields in the MOVE CORRESPONDING statement.</p> <p>If the statement has multiple receiving fields, it is replaced by separate MOVE CORRESPONDING statements for each of the receiving fields.</p>
FOR MULTIPLE REEL/UNIT clause	R	1-4	The clause is removed from the program.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
MULTIPLE FILE TAPE clause	R	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, this clause is removed from the I-O-CONTROL paragraph.
MULTIPLY...ON SIZE ERROR statement	F	1-5	MULTIPLY...ON SIZE ERROR statements with multiple receiving fields are flagged because the ON SIZE ERROR phrase will not be executed for intermediate results under the COBOL 85 Standard.
NOMINAL KEY clause	C R	1-4	<p>You should convert this file to VSAM.</p> <p>If the new organization for the file is INDEXED the NOMINAL KEY clause is removed. Before every I/O statement for these file, the following statement is added prior to the I/O statement: MOVE nominal-key-name TO record-key-name</p> <p>After the I/O statement, the statement MOVE record-key-name TO nominal-key-name</p> <p>If the new organization for the file is RELATIVE NOMINAL KEY is replaced by RELATIVE KEY.</p>
NOT	C F	1,3	<p>C: NOT in an abbreviated combined relation will be changed into an unabbreviated relation condition.</p> <p>F: If more than one NOT is involved, the expression is flagged. You will have to update the expression manually.</p>
NOTE statement	C	1-4	<p>The NOTE statement is used to write comments in the source program. It is not supported by the target languages.</p> <p>CCCA fully converts this statement by commenting it out.</p> <p>If the NOTE sentence is the first sentence of a paragraph, an asterisk is placed in column 7 of each line in the paragraph.</p> <p>If the Note sentence is not the first sentence of the paragraph, an asterisk is placed in column 7 of all lines up to the first period. If other language elements, not part of the NOTE statement, are on the first or last line of the NOTE statement, the line is split in order to isolate the NOTE.</p>
NSTD-REELS special register	F	1,2	The NSTD-REELS special register is not supported in the target languages. You should remove all references to it from the program.
OCCURS clause	C	1-4	<p>OS/VIS COBOL and DOS/VIS COBOL allow a non-standard order for phrases in the OCCURS clause. They allow the DEPENDING ON phrase after or among the ASCENDING/DESCENDING phrases. They also allow the DEPENDING ON phrase after the INDEXED BY phrase. The target languages only allow phrases in the standard order.</p> <p>Phrases in the OCCURS clause are put in the standard order.</p>

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
OCCURS DEPENDING ON clause (variable length record)	C	1-5	<p>COBOL statements that result in data transfer to a variable length receiver that contains its own OCCURS DEPENDING ON (ODO) object behave differently under the COBOL 85 Standard.</p> <p>Under the COBOL 68 and COBOL 74 Standards all ODO objects in sending and receiving fields must be set before the statement is executed. The actual lengths of the sender and receiver are calculated just before the execution of the data movement statement.</p> <p>Under the COBOL 85 Standard, in some circumstances, the maximum length of the variable length group is used when it is a receiver, whereas the COBOL 68 and COBOL 74 Standard always use the actual length.</p> <p>CCCA preserves the COBOL 68 and COBOL 74 behavior in the following way.</p> <p>For the following statements</p> <ul style="list-style-type: none"> • MOVE...TO identifier • READ...INTO identifier • RETURN...INTO identifier • UNSTRING...INTO identifier DELIMITER IN identifier <p>If the identifier is a variable length data item that contains its own ODO object, then reference modification is added to it.</p> <p>For example:</p> <pre>MOVE...TO identifier</pre> <p>is changed to</p> <pre>MOVE...TO identifier (1:LENGTH OF identifier)</pre> <p>For the following statements</p> <ul style="list-style-type: none"> • RELEASE record-name FROM identifier • REWRITE record-name FROM identifier • WRITE record-name FROM identifier <p>if the identifier is a variable length data item that contains its own ODO object, the FROM phrase is removed from the statement and a MOVE statement with reference modification is added before the statement:</p> <p>For example,</p> <pre>WRITE record-name FROM identifier</pre> <p>is changed to</p> <pre>MOVE identifier TO record-name (1:LENGTH OF record-name) WRITE record-name</pre> <p>MOVE CORRESPONDING statements are flagged as reference modification is not allowed when the CORRESPONDING phrase is specified.</p>

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
OCCURS DEPENDING ON clause (reference modification)	C	6	<p>VS COBOL II Release 3 (COBOL 85 Standard) and the target languages differ in the length used for the data transfer of a reference modified variable length group receiver that contains its own OCCURS DEPENDING ON (ODO) object. If no length is specified in the reference modifier, VS COBOL II Release 3 uses the current length of the group as defined by the ODO object.</p> <p>The target languages use the maximum length of the ODO object regardless of the value in the ODO.</p> <p>To preserve the behavior of this language element, the converter inserts the length of the receiver into the receiver. For example:</p> <pre>MOVE ODO-SENDER TO ODO-RECEIVER (1:)</pre> <p>where ODO-RECEIVER is a variable length field that contains its own ODO object is converted to:</p> <pre>MOVE ODO-SENDER TO ODO-RECEIVER (1:LENGTH OF RECEIVER)</pre>
ON statement	C F	1-4	<p>The ON statement is not supported by the target languages.</p> <p>The statement:</p> <pre>ON integer imperative statement</pre> <p>is converted to:</p> <pre>ADD 1 TO LCP-ONCTR-nn IF LCP-ONCTR-nn = integer imperative statement</pre> <p>The statement:</p> <pre>ON integer-1 until integer-2 imperative statement</pre> <p>is converted to:</p> <pre>ADD 1 TO LCP-ONCTR-nn IF LCP-ONCTR-nn > (integer-1 - 1) & < integer-2 imperative statement</pre> <p>A data item with the dataname LCP-ONCTR-nn (where nn is a sequence number) is added into the WORKING-STORAGE section with an initial value of zero.</p> <p>More complex ON statements are flagged.</p>
OPEN...DISP OPEN...LEAVE OPEN...REREAD statements	C	3,4	<p>The DISP option, LEAVE option and REREAD option are removed.</p>

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
OPEN...REVERSED statement	F	3,4	You should check whether the file in the OPEN statement has multiple reels. If it does, you will have to make a change to the program, because for the target languages this option is only valid for single reel files. OS/VS COBOL handles single reel files and in an undocumented extension multireel files.
ORGANIZATION clause	C	1-4	For VSAM files, this clause is removed.
OTHERWISE	C	1-4	OTHERWISE is replaced by ELSE.
PERFORM/ALTER	F	1-4	The section is checked for a priority number less than 49 and for the presence of ALTER. If this is not the case, manual changes may be required if this independent section is performed from outside the section.
PERFORM...VARYING... AFTER statement	F	1-5	Under the COBOL 85 Standard the rules for augmenting variables have changed. If there are dependencies between variables of the statement, then the statement may behave differently PERFORM...VARYING...AFTER statements are flagged if the conversion process detects a possible dependency. You should check to see if there are any dependencies between the variables of the statement that will result in different behavior. If there are you should modify the statement.
Periods	C	1-4	If there is no period immediately before or immediately after paragraph names or section headers in the PROCEDURE DIVISION, one is inserted.
PICTURE clause scaled integers	F I	1,3	Scaled integers (that is, data items that have a P as the rightmost symbol in their PICTURE strings) are flagged. If the scaled integer is the sending field in a MOVE statement and the receiving field is alphanumeric or numeric edited, you will have to convert this statement. If the scaled integer is compared with an alphanumeric or numeric edited field, you will have to convert this statement.
	F	2,4,5	Scaled integers are flagged. If the scaled integer is compared with a nonnumeric field, you will have to convert this statement.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
PROCESS statement	C F	1-4	The following options are obsolete and are replaced with new compile options: BUF, CLIST, DMAP, CATALR, LINECNT, LOAD, PMAP, SYST, SYSx, STATE, SYNTAX, CSYNTAX, SUPMAP, SXREF, VBSUM.
	R F	1-4	The following options are removed: BATCH, COUNT, ENDJOB, FLOW, LANGLVL1/2, SYMDMP, CDECK, FDECK, LCOL1/2, LSTONLY, LSTCOMP, L120, L132, OSDECK. The following option is removed if the target language is not COBOL II: RESIDENT.
	C F	5-7	The following option is replaced if the target language is not COBOL II: FDUMP.
	R F	5-7	The following option is removed if the target language is not COBOL II: RESIDENT.
		5-11	All compiler options that the target language does not support are removed from the statement and, where possible, are replaced with the target language equivalents.
PROCESSING MODE clause	R	1-4	The PROCESSING MODE clause is removed.
Program name	C	1-4	The target languages do not allow a data item to have a data-name that is the same as the program name. If there is one in the program, the dataname will be suffixed, in the same manner as datanames that are reserved words.
PROGRAM-ID header	C	1-4	If the PROGRAM-ID header begins in Area B, it is moved to the left so that it begins in Area A.
READ statement ISAM files	C	1-4	For randomly accessed indexed (ISAM) files, the following statement is added prior to the READ statement: MOVE nominal-key-name TO record-key-name After the READ statement, the statement MOVE record-key-name TO nominal-key-name is added. You should convert the file to VSAM.
READY TRACE statement	R	1-4	The statement is removed.
RECEIVE statement Communication feature	F	3,4	This is a Communication statement. The Communication module is not supported by the target languages and there is nothing with which it can be replaced.
RECORD CONTAINS	R	1-11	The clause is removed from the program, except for RECORD CONTAINS 0, which is left in place.
RECORDING MODE clause	R	1-11	The target language compilers ignore this clause, if it is specified for a VSAM file. If the clause is in a file description entry for a VSAM file or a file that is to be converted to VSAM, it is removed.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
REDEFINES clause in FD or SD entry	C	1-4	The target languages do not permit REDEFINES clauses in FD or SD entries. As they are superfluous, they are removed.
Picture P in RELATIVE KEY	F	1-5	This is flagged.
REMARKS paragraph	C	1-4	This is converted to comments with an asterisk (*) inserted in column 7 of the paragraph header and all succeeding lines of the paragraph.
REPLACE statement	F	6	The REPLACE statement is flagged because blank lines and comment lines in text that match pseudo-text are treated differently in the target languages. This could affect the semantics of the resulting program since the line numbers could be different. (For example if the program uses the USE FOR DEBUGGING declarative, the contents of the DEBUG-ITEM may be different). You should check that the semantics of the program is not altered.
REPORT SECTION & REPORT WRITER statements	F	1-4	These statements are not supported by the target languages: GENERATE INITIATE REPORT TERMINATE USE BEFORE REPORTING If you specify Y for the Flag Report Writer Statements option on Conversion Options panel 2, they will be flagged. If you want to keep these statements, you will require the COBOL Report Writer Pre-compiler.
RESERVE ALTERNATE AREAS	C	1-4	The following changes are performed: from RESERVE NO/n ALTERNATE AREA/AREAS. to RESERVE 1/n + 1 AREA/AREAS.
RESERVE AREAS	C	1-4	The following changes are performed: from ANS68 RESERVE n AREA/AREAS. to ANS74 RESERVE n+1 AREA/AREAS.
Reserved word	C	1-9	A suffix is appended to all user defined words that are reserved words in the target language. You specify the suffix that you want appended in the Reserved word suffix field of the Conversion Parameters Panel. -74 is the default suffix.
RESET TRACE statement	R	1-4	The statement is removed.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
REWRITE statement ISAM files	C	1-4	For randomly accessed indexed (ISAM) files, the following statement is added prior to the REWRITE statement: MOVE nominal-key-name TO record-key-name After the REWRITE statement, the statement MOVE record-key-name TO nominal-key-name is added. You should convert the file to VSAM.
SAME AREA clause	C	1-4	SAME AREA is changed to SAME RECORD AREA.
SEARCH ALL	F	1,3	The statement is flagged.
SEARCH...WHEN	C	1-4	In DOS/VS COBOL and OS/VS COBOL the ASCENDING/DESCENDING KEY data item may be specified as either the subject or the object of the WHEN relation condition. In the target languages it must be specified as the subject. If the key is not the subject, the condition is reversed, so that the subject becomes the object. NEXT SENTENCE is added if no statement is found.
SECURITY paragraph	C	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the SECURITY paragraph in the Identification Division is commented out.
SEEK	R	1-4	This is a BDAM file handling statement. The statement is removed from the program.
SELECT OPTIONAL	R	1,3	The OPTIONAL phrase is removed from the program.
SEND statement Communication feature	F	3,4	This is a Communication statement. The Communication module is not supported by the target languages and there is nothing with which it can be replaced.
SET...TO TRUE statement	F	5	Under the COBOL 74 Standard, the SET...TO TRUE statement is performed according to the rules of the MOVE statement. Under the COBOL 85 Standard, SET...TO TRUE follows the rules of the VALUE clause. There are three instances in which different behavior arises: <ul style="list-style-type: none"> • when the conditional variable is described by a JUSTIFIED clause and the condition name value is not justified. • when the conditional variable is described by a BLANK WHEN ZERO clause and the condition name value is zero. • when the conditional variable has editing symbols in its PICTURE string. <p>CCCA will flag all occurrences of such condition names when they appear in a SET...TO TRUE statement.</p>
SORT-OPTION clause	R	1,2	The clause is removed from the SD entry.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
START...USING KEY statement	C	1-4	The USING KEY clause of the START statement is not supported by the target languages. START statements that specify this clause are converted to START...KEY statements.
STOP RUN statement	C	1-5	See the EXIT PROGRAM statement entry in this table.
STRING statement	F	3-5	The statement is flagged if it has a receiving field with a PICTURE string that consist of A's and B's only. The COBOL 74 Standard classes these fields as alphabetic, whilst the COBOL 85 Standard classes them as alphanumeric-edited. You will have to make a change to the program as alphanumeric-edited receiving fields in the STRING statement are not permitted.
	F	3-5	The statement is flagged if the PROGRAM COLLATING SEQUENCE established in the OBJECT COMPUTER paragraph identifies an alphabet that was defined with the ALSO clause. Under these circumstances the statement will behave differently under the COBOL 85 Standard.
	F	3,4	String statements: <pre>STRING identifier-1 DELIMITED BY identifier-2 INTO identifier-3 WITH POINTER identifier-4...</pre> <p>where identifier-1 or identifier-2 is the same as identifier-3 or identifier-4 or where identifier-3 is the same as identifier-4 are flagged.</p>
> THAN relational operator	C	1-4	THAN is removed.
< THAN relational operator	C	1-4	THAN is removed.
> THEN relational operator	C	1-4	THEN is removed.
< THEN relational operator	C	1-4	THEN is removed.
THEN	R	1-4	THEN used between statements is removed.

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
TIME-OF-DAY special register	C	1-4	<p>The TIME-OF-DAY register is not supported by the target languages. Wherever TIME-OF-DAY is referenced in the program, it is replaced by code that obtains the time from the system and puts it in the format of the TIME-OF-DAY register. The fields required for the reformatting are generated in the WORKING-STORAGE section.</p> <p>For CICS programs converting to VS COBOL II, the time is retrieved from the system using an EXEC CICS ASKTIME statement. (CICS Release 1.7 or later is required.)</p> <p>For non-CICS programs converting to VS COBOL II, the ACCEPT...FROM TIME statement is used to obtain the time.</p> <p>For programs converting to a non-VS COBOL II level, the Intrinsic Function CURRENT-DATE is used to obtain the time. The fields required for reformatting are generated in the WORKING-STORAGE SECTION.</p>
= TO relational operator	C	1-4	TO is removed.
TOTALING/ TOTALED AREA	R I	3,4	This option is removed from the program. The data-name associated with this option is listed at the end of the diagnostic listing.
TRACE	R	1-4	The clause is removed from the program.
TRACK-AREA	R	1-4	The clause is removed from the program.
TRACK-LIMIT clause	R	3,4	The clause is removed from the program.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
TRANSFORM statement	C F	1-4	<p>CCCA converts a TRANSFORM to an INSPECT statement when a TRANSFORM statement is converting:</p> <ul style="list-style-type: none"> • From a figurative constant to another figurative constant • From a figurative constant to a nonnumeric literal • From a figurative constant to the value of an identifier • From a nonnumeric literal to a figurative constant • From the value of an identifier to a figurative constant • From the value of one identifier to the value of another <p>CCCA may convert a TRANSFORM to an INSPECT statement when a TRANSFORM statement is converting:</p> <ul style="list-style-type: none"> • A nonnumeric literal to another nonnumeric literal: <ul style="list-style-type: none"> – The “from” and the “to” literals must be the same size. If they are not, it is assumed the “to” literal is a single character. – If the “from” literal is 28 characters or less, the TRANSFORM is converted to an INSPECT statement. – If the “from” literal is more than 28 characters in length, manual intervention is required due to an internal limitation within CCCA. • the value of an identifier to a nonnumeric literal: <ul style="list-style-type: none"> – The “to” literal must be longer than a single character. If this is the case, it is assumed the literal is the same size as the identifier and conversion to the INSPECT statement occurs. – If the literal is a single character, manual intervention is required as CCCA cannot determine if this matches the size of the “from” identifier. <p>Manual intervention is required whenever a TRANSFORM statement is converting a nonnumeric literal to the value of an identifier.</p> <p>CCCA flags any INSPECT statements which it is unable to convert to INSPECT statements.</p>
UNSTRING statement	F	1,3	The UNSTRING statement is flagged if an ALL is specified in the DELIMITED BY phrase and the DELIMITER IN phrase is also specified.
	C	1-4	<p>Insert the word OR between identifiers in the DELIMITED BY phrase if it is missing and remove any commas or semicolons.</p> <p>Remove the word IS if it appears in the POINTER phrase.</p>
	F	1-4	The UNSTRING statement is flagged if subscripted data items are found following the DELIMITED BY/INTO/DELIMITER IN/COUNT IN phrases.
	F	3-5	<p>The statement is flagged if the PROGRAM COLLATING SEQUENCE established in the OBJECT COMPUTER paragraph identifies an alphabet that was defined with the ALSO clause.</p> <p>Under these circumstances the statement will behave differently under the COBOL 85 Standard.</p>

Converted COBOL

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
UPSI name	C F	1-5	Condition names are added. UPSI-n is replaced by condition-name. For example: LCP-ON-UPSI-n and LCP-OFF-UPSI-n Where n is a number from 0 to 7. Note: When the target language is COBOL/VSE, CCCA generates a warning message to highlight the need to possibly alter the program's JCL.
USE Procedures (precedences of)	F	6	In VS COBOL II Release 3, a GLOBAL file specific USE procedure always takes precedence even if an applicable mode specific USE procedure exists in the current program or if a mode specific USE procedure with the GLOBAL attribute in an outer program is nearer than the (GLOBAL) file-specific procedure. CCCA will flag all GLOBAL file-specific USE procedures. Mode-specific declaratives in contained programs will now take precedence. You should check the consequences of this.
USE AFTER STANDARD .. ON .. GIVING	R I	1-4	The GIVING option is removed from the program. A list of affected data-names is printed in the conversion listing.
 USE ... AFTER ... LABEL PROCEDURE ... 	R	1-4	Removed if the target language 5 is selected.
USE BEFORE STANDARD	R	1-4	USE BEFORE STANDARD is removed from the program.
USE FOR DEBUGGING	F	1-4	If an identifier following DEBUGGING is a file name, then the statement is flagged. If an identifier following DEBUGGING is not a procedure name and the Check procedure names option on Conversion Options panel 2 is set to Y, then the statement is flagged.
VALUE in 88 level	C	1-4	If the value of a level 88 refers to a variable defined with a PICTURE X and the value is not enclosed between quotes or apostrophes, quotes or apostrophes will be added.
VALUE in non-88 level	C	1-4	If the data item PICTURE string is numeric and the contents of the VALUE literal is also numeric, then the VALUE literal is changed to a numeric literal. That is, the quotes or apostrophes are removed.
Signed VALUE	C	1-4	The sign is removed from the value if PIC is unsigned.
VALUES	C	1-4	If not used in 88 level, VALUES is changed to VALUE.

Table 6. Language elements converted to specified target language (continued)

Language element	Conversion status	Language level	Notes
VALUE OF clause	R	1-11	If you specify Y for the Remove obsolete elements option on Conversion Options panel 2, the VALUE OF clause is removed from the FD entry.
WHEN-COMPILED	C	1,2	Programs converting to VS COBOL II with a date format of DD/MM/YY obtain the date and time from the WHEN-COMPILED special register.
	C	3,4	Programs converting to VS COBOL II obtain the date and time from the WHEN-COMPILED special register. Note that the original format of the WHEN-COMPILED special register included a 4-digit year. The century value is not available from the current special register and, if required, must be manually added to the converted source program.
	C	1-4	Programs converting to a non-VS COBOL II target level obtain the date and time information from the Intrinsic Function WHEN-COMPILED. The fields required for reformatting are generated in the WORKING-STORAGE SECTION.
WRITE statement ISAM files	C	1-4	For randomly accessed indexed (ISAM) files, the following statement is added prior to the WRITE statement: MOVE nominal-key-name TO record-key-name After the WRITE statement, the statement MOVE record-key-name TO nominal-key-name is added. You should convert the file to VSAM.
WRITE...BEFORE/AFTER ADVANCING mnemonic-name LINE/LINES	C	1-4	The target languages do not accept LINE or LINES in this statement. They are removed.
WRITE ... AFTER POSITIONING n lines	C	1-4	If n is a literal, this is changed to WRITE ... AFTER ADVANCING n LINES. If n is an identifier, SPECIAL-NAMES are generated and a section is added at the end of the program. Note: When compiling the converted program with the target language compiler, use the NOADV option. If POSITIONING and ADVANCING are used in the old program, you should review the ADV option.

Appendix B. Converted CICS commands

CCCA converts CICS Command Level statements from the syntax of the source language level to the target language level.

The Base Locator for Linkage sections (BLLs) are classified as either primary or secondary. Primary BLLs are associated with the portion of the record that is equal to or less than 4Kb (4096 bytes), and secondary BLLs correspond to record portions greater than 4Kb (4096 bytes).

Linkage section

If the CICS option on the Conversion (Selection) panel (see Figure 12 on page 28), is set to Y, the BLL definitions are removed. If the entire BLL structure is redefined, the redefined structure is removed. If the BLLs are not defined with a length of 4 bytes, the CICS conversion cannot be performed.

Note: If the level 01 of the BLL structure is FILLER, the BLL definitions are not removed from the Linkage Section, but all of the references to BLLs in the Procedure Division are processed.

Working-Storage Section

If needed by the conversion of statements involving primary BLLs, the following code is generated in the Working-Storage Section for use with the POINTER facility.

```
77 LCP-WS-ADDR-COMP PIC S9(8) COMP.  
77 LCP-WS-ADDR-PNTR REDEFINES LCP-WS-ADDR-COMP USAGE POINTER.
```

Table 7 on page 142 identifies statements that deal with primary BLLs.

Converted CICS

Table 7. Converted CICS commands

Element	Conversion status	Notes
ADD	C	<p>These primary BLL references are changed to ADDRESS OF special registers and POINTER facilities. For example:</p> <ul style="list-style-type: none"> • ADD ID1, ... TO BLL <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD ID1, ... TO LCP-WS-ADDR-COMP SET ADDRESS OF REC TO LCP-WS-ADDR-PNTR</p> <ul style="list-style-type: none"> • ADD BLL TO ID1, ID2 <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC ADD LCP-WS-ADDR-COMP TO ID1, ID2</p> <ul style="list-style-type: none"> • ADD ID1, ID2 GIVING BLL <p>is changed to</p> <p>ADD ID1, ID2 GIVING LCP-WS-ADDR-COMP SET ADDRESS OF REC TO LCP-WS-ADDR-PNTR</p> <ul style="list-style-type: none"> • ADD ID1, BLL1 GIVING BLL2 BLL3 <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC1 ADD ID1, LCP-WS-ADDR-COMP GIVING LCP-WS-ADDR-COMP SET ADDRESS OF REC2 TO LCP-WS-ADDR-PNTR SET ADDRESS OF REC3 TO LCP-WS-ADDR-PNTR</p> <ul style="list-style-type: none"> • ADD ID1, BLL1 GIVING ID2 ID3 <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC1 ADD ID1, LCP-WS-ADDR-COMP GIVING ID2 ID3</p>
COMPUTE	C	<p>The primary BLL references are changed to ADDRESS OF special register and POINTER facilities. For example:</p> <ul style="list-style-type: none"> • COMPUTE BLL = exp (BLL) <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC COMPUTE LCP-WS-ADDR-COMP = EXP (LCP-WS-ADDR-COMP)</p> <ul style="list-style-type: none"> • COMPUTE ID = exp (BLL) <p>is changed to</p> <p>SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC COMPUTE ID = exp (LCP-WS-ADDR-COMP)</p> <ul style="list-style-type: none"> • COMPUTE BLL = exp (BLL) <p>is changed to</p> <p>COMPUTE LCP-WS-ADDR-COMP = exp (BLL)</p>

Table 7. Converted CICS commands (continued)

Element	Conversion status	Notes
END-EXEC	F	An error message is generated if an EXEC CICS statement does not finish with an END-EXEC statement.
EXEC CICS used with SET option	C	<p>The primary BLLs are replaced by corresponding ADDRESS OF special register. For example: EXEC CICS READ ... SET(BLL1)...</p> <p>is replaced by EXEC CICS READ ... SET(ADDRESS OF REC1)...</p> <p>The statements affected are: GETMAIN, READ, READNEXT, READPREV, READQ, RECEIVE, RETRIEVE, SEND CONTROL, SEND PAGE, SEND TEXT, LOAD, CONVERSE, ISSUE RECEIVE, and POST.</p>
EXEC CICS used with CICS ADDRESS statements	C	<p>The primary BLL is replaced by corresponding ADDRESS OF special register. The options affected are CSA, CWA, EIB, TCTUA, and TWA. For example: EXEC CICS ADDRESS TWA(BLL)</p> <p>is replaced by EXEC CICS ADDRESS TWA(ADDRESS OF TWA).</p>
MOVE	C	<p>The primary BLL references are changed to ADDRESS OF special register and POINTER facility. For example:</p> <ul style="list-style-type: none"> • MOVE BLL1 TO BLL2 <p>is changed to SET ADDRESS OF REC2 TO ADDRESS OF REC1</p> <ul style="list-style-type: none"> • MOVE ID TO BLL <p>is changed to MOVE ID TO LCP-WS-ADDR-COMP SET ADDRESS OF REC TO LCP-WS-ADDR-PNTR</p> <ul style="list-style-type: none"> • MOVE BLL TO ID <p>is changed to SET LCP-WS-ADDR-PNTR TO ADDRESS OF REC MOVE LCP-WS-ADDR-COMP TO ID</p>
SERVICE RELOAD	C	SERVICE RELOAD is replaced by CONTINUE.
SUBTRACT	C	The primary BLL references are changed to ADDRESS OF special register and POINTER facility. This conversion is the same as the conversion for ADD.

C=Converted R=Removed F=Flagged I=Information

Note: For secondary BLLs, LCP892 replaces any statement associated with these BLLS by CONTINUE. For example: ADD 4096 T0 BLL is replaced by CONTINUE.

Appendix C. Messages

There are four categories of *ABJnnnn* messages:

0000-0100

Converter error messages

1002-3999,

9001

LCP compiler error messages

4000-5999

Tokenization diagnostics

6000-6999

Conversion diagnostics from the supplied LCPs

Note: Your LCPs should use message numbers outside these ranges.

Converter error messages

ABJ0000 00

PROCESSING &1

ABJ0012 16

ADD MEMBER &1 FAILED IN FILE &2.&3

ABJ0014 16

RESERVED WORD TABLE NOT FOUND

ABJ0015 16

TABLE FILE NOT CREATED

ABJ0020 16

FILE FULL &1.&2

The referenced file has to be redefined with more space.

ABJ0021 16

SOURCE OR COPY OUTPUT LIBRARY FULL

ABJ0022 16

INVALID KEY FOR FILE DRWORK OR FILE DRWORK FULL

ABJ0023 16

INVALID FILE/RECORD NAME OR DRWORK FULL

ABJ0025 16

OPTION RECORD MISSING IN CONTROL FILE

ABJ0026 16

SOURCE FILE EMPTY - MEMBER &1

ABJ0027 16

NO TOKEN FILE GENERATED FROM PHASE 1

ABJ0029 16

ERRORS OCCURRED DURING PREVIOUS CONVERSION

ABJ0030 16

PROGRAM &1 NOT CONVERTED

Messages

- ABJ0031 16**
SEQUENCE ERROR DURING EXECUTION OF PHASE 3
- ABJ0040 16**
TERMINAL ERROR FOUND DURING CONVERSION OF &1
- ABJ0041 16**
INVALID READ OF TOKEN FILE
- This message occurs when an LCP is checking the syntax of a COBOL statement. This message is followed by message LCP0046, that indicates which LCP issues the message and on which statement. Report these 2 messages to your support organization.
- ABJ0042 16**
INVALID READ OF &1 FILE - KEY &2 - RC &3
- ABJ0043 16**
INVALID UPDATE OF &1. FILE - KEY &2 - RC &3
- ABJ0044 16**
INVALID ADD-TEXT LENGTH - MUST BE GREATER THAN ZERO
- ABJ0045 16**
INVALID TOKEN LENGTH - TOKEN SEQUENCE IS &1 - ADD-TEXT IS &2
- ABJ0046 16**
LCP PROGRAM IS &1 - STATEMENT NUMBER IS &2
- ABJ0047 16**
INVALID RECORD NAME
- ABJ0048 16**
TOKEN SEQUENCE IS &1 - RECORD NAME IS &2 - RC &3
- ABJ0064 16**
NO INFORMATION FOR THIS LCP
- ABJ0070 16**
READ/WRITE ERROR ON FILE &2 FILE STATUS IS &1
- ABJ0071 16**
WRITE ERROR ON FILE &2 FILE STATUS IS &1
- ABJ0072 16**
READ ERROR ON FILE &2 FILE STATUS IS &1
- ABJ0073 16**
OPEN ERROR ON FILE &2 FILE STATUS IS &1
- ABJ0074 16**
I/O ERROR ON FILE SOURCE FILE STATUS IS &1
- ABJ0075 16**
INVALID KEY IN CONTROL FILE
- ABJ0076 16**
NO MAP PRODUCED
- When CICS option is set to Y (YES), the Linkage Section of the program is compiled. If no map is produced by the compiler, no conversion is performed.
- ABJ0077 16**
ERROR WHILE COMPILING LINKAGE SECTION - CHECK ERRORS WITHIN LINKAGE SECTION

When CICS option is set to Y (YES), the Linkage Section of the program is compiled. If there are compilation errors, they are listed. The CICS commands are not converted.

ABJ0078 16

PROGRAM NOT CONVERTED

ABJ0079 12

CICS STATEMENTS NOT CONVERTED

ABJ0080 12

BLLS DEFINED IN LINKAGE SECT BUT NO RECORD DEFINED IN LINKAGE SECTION

When CICS option is set to Y (YES), the Linkage Section of the program is compiled. BLLs are defined but no records are defined. CICS statements are not converted.

ABJ0081 08

MORE THAN 200 REDEFINED BLLS IN LINKAGE SECTION - REDEFINED BLLS NOT FLAGGED

More than 200 BLLs are redefined. The program is not converted.

ABJ0082 16

BLL LENGTH NOT = 4 UNABLE TO PROCESS

A BLL is defined and the length is not four bytes. The program is not converted.

ABJ0083 16

MORE THAN 200 BLLS DEFINED IN LINKAGE SECTION

More than 200 BLLs are defined in Linkage Section. The program is not converted. This is a limit of the converter.

ABJ0084 16

PICTURE IS NOT 0CLX OR X..C

A picture of a BLL is not 9(4) or X(4). The program is not converted.

ABJ0085 12

NO BLLS DEFINED IN LINKAGE SECTION

There is a Linkage Section, but no BLLs are defined. The CICS statements are not converted.

ABJ0090 16

XXX ERROR - FILE=XXX - FILE STATUS = nn <KEY=XXXXXX>

Note: The meaning of the file status is given in the *IBM COBOL Programming Guide Release 2* for your platform.

ABJ0092 16

ERROR WRITING COPY MEMBER &1 - DDNAME &2 - RC = &3

RC = 8

WRITE I/O Error

RC = 32

RECORD length different from 80

RC = 48

OPEN Error.

Messages

ABJ0093 16

STOW ERROR COPY MEMBER &1 - DDNAME &2 - RC = &3

RC = 4

STOW (without R) for member already in the library

RC = 12

File not opened.

Note: Parameters &1, &2, &3, and &4 are replaced by the correct values once the message is printed.

LCP compiler error messages

ABJ1002 08

NAME OR LITERAL EXCEEDS 30 CHARACTERS

ABJ1003 08

MORE THAN 10 WORDS SPECIFIED ON THE SAME LINE

ABJ2001 08

DATA NAME OR PARAGRAPH NAME IS A RESERVED WORD

ABJ2004 08

INVALID SYNTAX IN DATA NAME OR PARAGRAPH NAME

ABJ2005 08

INVALID SYNTAX IN *CONVER

ABJ2006 08

INVALID DATA NAME OR RESERVED WORD

ABJ2007 08

PERIOD OR SPACE NOT FOUND

ABJ2008 08

INVALID PARAGRAPH NAME IN 'PERFORM' STATEMENT

ABJ2009 08

ELEMENT NOT IN AREA B

ABJ2010 08

DUPLICATE PARAGRAPH NAME

ABJ2011 08

TOO MANY PARAGRAPH NAMES IN PROGRAM - MAXIMUM PARAGRAPH NAMES IS 100

ABJ2012 08

NO PARAGRAPH NAME BEFORE FIRST LCP STATEMENT

ABJ2013 08

INVALID STATEMENT

ABJ2014 08

INVALID STATEMENT AFTER PERIOD

ABJ2015 08

PERIOD REQUIRED OR SYNTAX ERROR

ABJ2016 08

INVALID DATA NAME OR LITERAL

ABJ2017 08

DUPLICATE DATA NAME

ABJ2018 08
TOO MANY DATA NAMES - MAXIMUM 50 PER LCP PROGRAM

ABJ2019 08
NO PICTURE CLAUSE FOR THIS DATA NAME

ABJ2020 08
INVALID PICTURE CLAUSE FOR THIS DATA NAME

ABJ2021 08
'AND' RELATION INVALID IN THIS CONTEXT

ABJ2022 08
PARAGRAPH NAME NOT IN AREA A

ABJ2023 08
FACTOR 1 MUST BE A DATA NAME

ABJ2024 08
INVALID LITERAL OR DATA NAME USED BUT NEVER DEFINED

ABJ2025 08
'UNTIL' OPTION INVALID IN THIS CONTEXT

ABJ2026 08
INVALID CONDITION IN IF, UNTIL, OR, AND STATEMENT

ABJ2027 08
INVALID CLASS OPERAND IN CONDITION STATEMENT

ABJ2028 08
INVALID WORD AFTER OPERAND 2 IN CONDITION

ABJ2030 08
INVALID SYNTAX IN MOVE, ADD, SUBTRACT STATEMENT

ABJ2031 08
FACTOR 2 MUST BE A DATA NAME

ABJ2034 08
'IF' STATEMENT INVALID IN THIS CONTEXT

ABJ2035 08
'OR' RELATION INVALID IN THIS CONTEXT

ABJ2036 08
'ELSE' IS UNMATCHED BY 'IF'

ABJ2037 08
PARAGRAPH NAME ALREADY DEFINED AS DATA NAME

ABJ2038 08
FACTORS 1, 2 MUST BE NUMERIC IN 'ADD' OR 'SUBTRACT' STATEMENT

ABJ2039 08
MOVE ALPHABETIC TO NUMERIC IS INVALID IN 'MOVE'

ABJ2040 08
*CONVER MUST BE FIRST LCP STATEMENT - SYNTAX IS: * IN COLUMN 7,
CONVER OR CONVERQ IN COLUMN 12 TO 18

ABJ2044 08
CORRECT *CONVER AND RETRY

ABJ2045 08
TERMINAL ERROR FOUND IN LCP PROGRAM &1

Messages

ABJ3029 08
PARAGRAPH NAME USED BUT NEVER DEFINED

ABJ3032 08
LCP PROGRAM TOO BIG - MAXIMUM 18000 BYTES PER PROGRAM

ABJ3033 08
INVALID PARAGRAPH NAME 2 IN PERFORM STATEMENT

Tokenization diagnostics

ABJ4001 16
BLL REQUEST AND NO LINKAGE SECTION.

ABJ4002 16
BLL REQUEST AND NO PROCEDURE DIVISION.

ABJ4003 16
SYSTEM PARAMETERS COULD NOT BE SET.

ABJ4004 16
I/O ERROR OF BLL FILE.

ABJ4005 16
MLE FILE COULD NOT BE OPENED.

ABJ4006 16
PROGRAM ID &1 NOT IN MLE FILE.

ABJ4007 16
RSW TABLE EXPANSION EXCEEDED.

ABJ4008 16
UNEXPECTED END OF DATA ON INPUT PROGRAM SOURCE.

ABJ4009 16
AN I/O ERROR OCCURRED WHILE READING A COPY LIBRARY MEMBER.

ABJ4010 16
INSUFFICIENT STORAGE AVAILABLE BELOW THE 16M LINE.

ABJ4011 16
INSUFFICIENT STORAGE AVAILABLE FOR CCCA PROCESSING.

ABJ4012 16
OPEN FAILURE ON INPUT SOURCE FILE.

ABJ4013 16
NO INVOCATION OPTIONS SPECIFIED.

ABJ4014 16
INSUFFICIENT STORAGE AVAILABLE FOR CCCA PROCESSING.

ABJ4015 16
I/O ERROR ON DRWORK VSAM FILE.

ABJ4016 16
ERRORS IN DATE IDENTIFICATION FILE INPUT. CONVERSION PROCESS
TERMINATED.

ABJ4017 16
&1 DATASET COULD NOT BE OPENED.

ABJ4018 16
&1 DATASET I/O ERROR.

ABJ4019 16
&1 DATASET COULD NOT BE CLOSED.

ABJ4020 16
CRITICAL DATASET COULD NOT BE CLOSED.

ABJ4024 16
AN ERROR OCCURRED WHILE ATTEMPTING TO LOAD MODULE &1.

ABJ4025 16
AN ERROR OCCURRED WHILE ATTEMPTING TO DELETE MODULE &1.

ABJ4026 16
LISTING HEADING OR LISTING ANNOTATION LINE(S) ID "&1" WAS NOT FOUND
IN THE LISTING HEADER DATA FILE.

ABJ4027 16
AN ERROR OCCURRED DURING RETRIEVAL OF DATA FROM THE LISTING HEADER
DATA FILE.

ABJ4028 16
A REQUEST WAS ISSUED TO EXPAND A STATIC TABLE.

ABJ4029 16
THERE WAS AN ATTEMPT TO PRIME A TABLE THAT WAS PREVIOUSLY PRIMED.

ABJ4030 16
THERE WAS AN ATTEMPT TO FREE A TABLE THAT WAS PREVIOUSLY FREED.

ABJ4031 16
LANGUAGE TABLE &&&&&&& COULD NOT BE DYNAMICALLY LOADED.

ABJ4032 16
INSUFFICIENT STORAGE TO DYNAMICALLY LOAD LANGUAGE TABLE.

ABJ5000 00
DATE IDENTIFICATION FILE ERROR IN RECORD &1 COLUMN \$\$.

ABJ5001 00
UNEXPECTED DATA FOUND BEFORE PROGRAM NAME. SKIPPED TO THE NEXT
PROGRAM NAME.

ABJ5002 00
UNEXPECTED DATA FOUND BEFORE LINE NUMBER. SKIPPED TO THE NEXT LINE
NUMBER.

ABJ5003 00
RESERVED WORD "OF" USED IMPROPERLY.

ABJ5004 00
INVALID COBOL USER WORD IN DATA NAME.

ABJ5005 00
UNRECOGNIZED DATE FORMAT SPECIFICATION.

ABJ5006 00
UNEXPECTED DATA FOUND WHERE "OF" EXPECTED.

Conversion diagnostics from LCPs

ABJ6000 08
***** MANUAL UPDATE REQUIRED

ABJ6001 00
'THEN' IS REMOVED

Messages

ABJ6002 00
LCP-xxx DATE/TIME DATA ITEMS GENERATED IN WORKING-STORAGE FOR
CURRENT-DATE/TIME-OF-DAY/WHEN-COMPILED CONVERSIONS

ABJ6003 00
NEW CODE GENERATED FOR CURRENT-DATE

ABJ6004 08
UNABLE TO SUCCESSFULLY CONVERT TRANSFORM TO INSPECT ***** MANUAL
UPDATE REQUIRED

ABJ6005 00
NEW CODE GENERATED FOR TIME-OF-DAY

ABJ6006 08
DUPLICATE CHARACTERS FOUND IN "FROM" LITERAL ***** MANUAL UPDATE
REQUIRED

ABJ6007 00
NEW CODE GENERATED FOR WHEN-COMPILED

ABJ6008 04
RELATIVE KEY DEFINED AS GROUP *MANUAL UPDATE MAY BE REQUIRED

ABJ6009 00
MULTIPLE MOVE CORRESPONDING CHANGED TO SEPARATE MOVES

ABJ6010 00
REDEFINES CLAUSE IN FD REMOVED

ABJ6011 00
REMARKS CHANGED TO COMMENT

ABJ6012 00
VALUE CLAUSE IS CHANGED

ABJ6013 08
DATA EXCEEDS 28 CHARACTERS DATA NOT RIGHT JUSTIFIED ***** MANUAL
UPDATE REQUIRED

ABJ6014 00
COMBINED EXPRESSION IS CHANGED

ABJ6015 04
NEW CODE GENERATED FOR WHEN-COMPILED ** WARNING CENTURY VALUE NOT
SET, MANUAL UPDATE MAYBE REQD

ABJ6016 00
HYPHEN ADDED TO DATE

ABJ6017 00
EJECT REPLACED BY /

ABJ6018 00
TALLY IS INITIALIZED

ABJ6019 00
EXAMINE REPLACED BY INSPECT

ABJ6020 04
MOVE ALL STATEMENT FOUND 68 STANDARD INTERPRETATION *MANUAL UPDATE
MAY BE REQUIRED

ABJ6021 00
OTHERWISE REPLACED BY ELSE

ABJ6022 00
NOTE CHANGED TO COMMENT

ABJ6023 08
CURRENCY SIGN CLAUSE FOUND ***** MANUAL UPDATE REQUIRED

ABJ6024 00
OPTIONAL IS REMOVED

ABJ6025 08
UNSTRING ... DEL. BY ALL FOUND 68 STANDARD INTERPRETATION *****
MANUAL UPDATE REQUIRED

ABJ6026 04
SCALED VARIABLE FOUND 68 STANDARD INTERPRETATION *MANUAL UPDATE MAY
BE REQUIRED

ABJ6027 04
IF TAPE IS 'UNLABELLED', CHECK JCL FOR A MATCHING TLBL STATEMENT AND
REMOVE

ABJ6028 04
UPSI SWITCHES MAY ONLY BE SET USING THE LE/VSE UPSI RUN-TIME OPTION
*****JCL UPDATE MAY BE REQUIRED

ABJ6030 08
ASCII FILE TO BE CHECKED

ABJ6031 00
SPECIAL-NAMES IS GENERATED

ABJ6032 04
MNEMONIC NAME FOUND *MANUAL UPDATE MAY BE REQUIRED

ABJ6033 00
INTEGER IS REMOVED

ABJ6034 08
FOR MULTIPLE REEL/UNIT IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6035 00
NOMINAL IS CHANGED TO RELATIVE FOR VSAM RRDS

ABJ6036 00
PERIOD ADDED AT THE END OF THE PARAGRAPH

ABJ6037 00
RESERVE AREA IS CHANGED

ABJ6038 00
FILE-LIMIT CLAUSE IS REMOVED

ABJ6039 00
PROCESSING MODE CLAUSE IS REMOVED

ABJ6040 00
APPLY CLAUSE IS REMOVED

ABJ6041 08
TOTALING/TOTALED AREA REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6042 00
DISP/POSITIONING OPTION IN CLOSE IS REMOVED

ABJ6043 00
LEAVE/REREAD/DISP OPTION IN OPEN IS REMOVED

Messages

ABJ6044 08
GIVING OPTION IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6045 08
USE BEFORE IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6046 00
WRITE...AFTER POSITIONING...CHANGED TO WRITE...AFTER ADVANCING

ABJ6047 00
LCP-ASA DATA NAME IS GENERATED

ABJ6048 00
LCP-WRITE-... SECTION IS ADDED

ABJ6049 08
FILE TO BE CONVERTED TO VSAM RRDS *****

ABJ6050 00
PERIOD ADDED

ABJ6051 00
TO IS REMOVED

ABJ6052 00
ACTUAL IS CHANGED TO RELATIVE FOR VSAM RRDS

ABJ6053 00
SAME AREA CHANGED TO SAME RECORD AREA

ABJ6054 08
LABEL RECORDS CHANGED TO STANDARD

ABJ6055 00
RECORDING MODE IS REMOVED

ABJ6056 00
SEEK IS REMOVED

ABJ6057 08
TRACK-LIMIT CLAUSE IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6058 08
FILE TO BE CONVERTED TO VSAM KSDS *****

ABJ6059 00
THAN IS REMOVED

ABJ6060 00
TRACK AREA CLAUSE IS REMOVED

ABJ6061 08
USING IS REMOVED ***** CHECK IF GENERIC KEY

ABJ6062 00
LCP-EOP DATA NAME IS GENERATED

ABJ6063 00
SUPERFLUOUS 'INTO' REMOVED

ABJ6064 00
ADDITIONAL ASSIGNMENT NAMES REMOVED

ABJ6065 08
REPORT WRITER STATEMENT FOUND ***** MANUAL UPDATE REQUIRED

ABJ6066 08
USE FOR DEBUGGING ONLY ALLOWED FOR PROCEDURE NAME ***** MANUAL
UPDATE REQUIRED

ABJ6067 08
ON STATEMENT FOUND ***** MANUAL UPDATE REQUIRED

ABJ6068 04
READY/RESET TRACE IS REMOVED

ABJ6069 00
EXHIBIT CHANGED TO DISPLAY

ABJ6070 00
EXHIBIT CHANGED TO DISPLAY TREATED AS EXHIBIT NAMED

ABJ6071 08
DEBUG IS NOT SUPPORTED

ABJ6072 08
DATA EXCEEDS 28 CHARACTERS DATA SHOULD BE BETWEEN QUOTES *****
MANUAL UPDATE REQUIRED

ABJ6073 08
COMMUNICATIONS NOT SUPPORTED ***** MANUAL UPDATE REQUIRED

ABJ6074 00
NOMINAL KEY FIELD MOVED TO RECORD KEY FIELD

ABJ6075 08
***** ERROR FILE NAME *****

Check if the SELECT statement, ASSIGN clause, and FD definition match.
This takes place when the converter writes a record in DRWORK file with
a duplicate key. If the program compiles without error at the source
language level, report the problem, providing the list of the Input-Output
Section in the Environment Division and all the File Descriptions (FD) in
the Data Division. Also provide a printout of the DRWORK.ABJ file.

ABJ6076 00
PROGRAM-ID PARAGRAPH IS ADDED

ABJ6077 08
QUALIFIED KEY NOT SUPPORTED ***** MANUAL UPDATE REQUIRED

ABJ6078 00
NOMINAL KEY IS REMOVED

ABJ6079 04
** WARNING POSSIBLE SUBSCRIPT EVALUATION DIFFERENCES

ABJ6080 00
FILE STATUS CLAUSE IS ADDED

ABJ6081 00
RECORD KEY FIELD MOVED BACK TO NOMINAL KEY

ABJ6082 00
NEW ORGANIZATION IS ADDED

ABJ6084 04
"NOT" IN ABBREVIATED COMBINED RELATION CONDITION. CONDITION IS NOW
EXPANDED DIFFERENTLY. *MANUAL UPDATE REQUIRED

Messages

ABJ6085 00
DECLARATIVE IS ADDED

ABJ6086 08
NO FILE STATUS TEST ***** MANUAL UPDATE REQUIRED

ABJ6087 00
CODE-SET CLAUSE IS ADDED

ABJ6088 00
LANGLEVEL 1 COPY IS CHANGED

ABJ6089 00
UPSI CHANGED TO CONDITION NAME

ABJ6090 08
ONLY CONDITION NAME IS ALLOWED ***** MANUAL UPDATE REQUIRED

ABJ6091 00
TRANSFORM REPLACED BY INSPECT

ABJ6092 04
MANUAL CHANGE MAY BE REQUIRED IF THIS INDEPENDENT SECTION IS
PERFORMED OUTSIDE THE SECTION

ABJ6093 00
DATA ITEM LCP-FILE-STATUS IS GENERATED

ABJ6094 00
FILE STATUS TEST IS ADDED

ABJ6095 00
LABEL CLAUSE IS REMOVED

ABJ6096 04
MULTIPLE "NOT" FOUND *****MANUAL UPDATE MAY BE REQUIRED

ABJ6097 00
KEY DATA ITEM CHANGED TO BE THE SUBJECT

ABJ6098 00
ASSIGNMENT NAME IS CHANGED

ABJ6099 08
PERFORM KEYCALC IS ADDED USER SHOULD PROVIDE KEYCALC SECTION

ABJ6103 99

** DATA NAMES TO BE CHECKED **

ABJ6104 99
* USED IN LABEL CLAUSE *

ABJ6105 99
* USED IN TOTALING CLAUSE *

ABJ6106 99
* USED IN TOTALED CLAUSE *

ABJ6107 99
* USED IN GIVING OPTION *

ABJ6109 99
* USED AS UPSI *

ABJ6110 99
 * USED AS SCALED VARIABLE *

ABJ6111 00
 PICTURE CHANGED FOR RELATIVE KEY

ABJ6112 08
 PROC/FILE NAME NOT ALLOWED ***** MANUAL UPDATE REQUIRED

ABJ6114 08
 INVALID PICTURE FOR RELATIVE KEY ***** MANUAL UPDATE REQUIRED

ABJ6115 00
 SYSTEM NAME CHANGED TO IBM-370

ABJ6116 00
 ON STATEMENT CHANGED TO IF

ABJ6117 00
 ON COUNTER GENERATED IN WORKING STORAGE

ABJ6118 08
 TOO MANY QUALIFIERS ***** MANUAL UPDATE REQUIRED

ABJ6119 00
 RECORDING MODE CLAUSE REMOVED

ABJ6122 08
 RELATIVE KEY NOT FOUND RELATIVE FILE NAME IS :

ABJ6124 04
 EXEC STATEMENT FOUND *MANUAL UPDATE MAY BE REQUIRED

ABJ6125 00
 USER-DEFINED WORD IS RESERVED WORD IN TARGET LANGUAGE SUFFIX HAS BEEN ADDED.

ABJ6126 99

 * END OF COBOL CONVERSION *
 * 5648-B05 COBOL CONVERSION *

ABJ6127 08
 RELATIVE KEY NAME NOT DEFINED IN WORKING-STORAGE SECTION KEY IS :

ABJ6128 00
 RECORD KEY IS ADDED

ABJ6132 00
 THEN REPLACED BY THAN

ABJ6133 00
 WORKING-STORAGE SECTION ADDED

ABJ6134 08
 ILLEGAL USE OF CURRENT-DATE ***** MANUAL UPDATE REQUIRED

ABJ6135 08
 ILLEGAL USE OF TIME-OF-DAY ***** MANUAL UPDATE REQUIRED

ABJ6136 00
 NEXT SENTENCE ADDED

ABJ6142 00
 IDENTIFIER CHANGED TO LITERAL

Messages

ABJ6144 08
COM-REG SPECIAL REGISTER FOUND ***** MANUAL UPDATE REQUIRED

ABJ6145 08
NSTD-REELS SPECIAL REG FOUND ***** MANUAL UPDATE REQUIRED

ABJ6146 08
SORT-OPTION CLAUSE IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6147 00
ENTER STATEMENT IS REMOVED

ABJ6148 00
LCP NOT FOUND - RECOMPILE LCP

ABJ6151 00
RECORDS WORD IS ADDED

ABJ6152 00
PARAGRAPH CHANGED TO COMMENT

ABJ6153 08
ERROR WRITING CONTROL FILE FILE CONVERSION MAY BE WRONG *****
CHECK CONTROL FILE

ABJ6160 00
CONFIGURATION SECTION ADDED

ABJ6161 08
SORT-OPTION CLAUSE IS REMOVED ***** MANUAL UPDATE REQUIRED

ABJ6162 08
NSTD-REELS SPECIAL REG FOUND ***** MANUAL UPDATE REQUIRED

ABJ6170 00
ALPHABET WORD IS ADDED

ABJ6171 00
ALPHABETIC CHANGED TO ALPHABETIC-UPPER

ABJ6172 00
EXIT PROGRAM IS ADDED

ABJ6173 00
ALPHABET CLAUSE ADDED FOR ASCII FILE

ABJ6174 00
ACTUAL LENGTH ADDED TO VARIABLE LENGTH RECEIVING ITEM

ABJ6175 08
MAXIMUM LENGTH USED FOR VARIABLE LENGTH RECEIVING ITEM *****
MANUAL UPDATE REQUIRED

ABJ6176 00
ABEND CODE GENERATED IN WS

ABJ6177 00
RECORD CLAUSE IS REMOVED

ABJ6178 08
UPSI NOT ALLOWED AS QUALIFIER ***** MANUAL UPDATE REQUIRED

ABJ6179 00
CONTINUE STATEMENT IS ADDED

ABJ6180 08
 SYMBOL P IN PIC NOT ALLOWED FOR RELATIVE KEY ***** MANUAL UPDATE
 REQUIRED

ABJ6181 00
 OBSOLETE ELEMENT IS REMOVED

ABJ6182 00
 VALUES CHANGED TO VALUE

ABJ6183 00
 LINE/LINES IS REMOVED

ABJ6184 00
 SIGN IS REMOVED IN VALUE

ABJ6185 08
 COPY FOUND IN NOTE END OF NOTE NOT PROCESSED ***** MANUAL UPDATE
 REQUIRED

ABJ6186 04
 OBSOLETE ELEMENT IS REMOVED ***** MANUAL UPDATE MAYBE REQUIRED

ABJ6200 08
 LEVEL 01 BLL FOUND ***** MANUAL UPDATE REQUIRED

ABJ6201 00
 POINTER OPTION IN EXEC CICS CHANGED TO ADDRESS OF ...

ABJ6202 00
 SERVICE RELOAD REPLACED BY CONTINUE

ABJ6203 00
 BLL'S ARE REMOVED

ABJ6204 08
 UNIDENTIFIED BLL ***** MANUAL UPDATE REQUIRED

ABJ6205 08
 PRIMARY BLL FOUND NOT IN MOVE CALL ADD SUBTRACT ***** MANUAL
 UPDATE REQUIRED

ABJ6206 00
 SERVICE RELOAD IS REMOVED

ABJ6207 00
 BLL CONVERTED TO SET POINTER SET ADDRESS OF ...

ABJ6208 00
 STATEMENT WITH SECONDARY BLL REPLACED BY CONTINUE

ABJ6209 00
 BLL REPLACED BY ADDRESS OF ...

ABJ6210 08
 UNDEFINED/REDEFINED BLL FOUND ***** MANUAL UPDATE REQUIRED

ABJ6211 08
 BLL FOUND ***** MANUAL UPDATE REQUIRED

ABJ6212 00
 WORKING POINTER FOR CICS ADDED TO WORKING STORAGE

ABJ6213 08
 MULTIPLE MOVE NOT PROCESSED ***** MANUAL UPDATE REQUIRED

I
 I

Messages

ABJ6214 08
MOVE CORR NOT PROCESSED ***** MANUAL UPDATE REQUIRED

ABJ6215 08
UNDEFINED STATEMENT WITH BLL ***** MANUAL UPDATE REQUIRED

ABJ6216 08
BLL MIXED WITH IDENTIFIER(S) IN A MOVE, ADD OR SUBTRACT *****
MANUAL UPDATE REQUIRED

ABJ6217 08
MULTIPLE ADD NOT PROCESSED ***** MANUAL UPDATE REQUIRED

ABJ6218 00
PRIMARY BLL IN ADD SUBTRACT CHANGED TO ADDRESS OF ...

ABJ6219 08
MULTIPLE BLL BEFORE GIVING IN ADD OR SUBTRACT STATEMENT *****
MANUAL UPDATE REQUIRED

ABJ6220 00
PRIMARY BLL IN COMPUTE CHANGED TO ADDRESS OF ...

ABJ6221 08
ILLEGAL USE OF SECONDARY BLL ***** MANUAL UPDATE REQUIRED

ABJ6222 04
MORE THAN 3 LEVELS OF QUALIFICATION ON TABLE. *MANUAL UPDATE MAY BE
REQUIRED

ABJ6223 00
SUPERFLUOUS "TO" REMOVED

ABJ6224 04
COPY...REPLACING ENCOUNTERED *MANUAL UPDATE MAY BE REQUIRED

ABJ6225 08
BRACKETS MOVED *MANUAL UPDATE MAY BE REQUIRED

ABJ6226 00
ENVIRONMENT DIVISION MOVED.

ABJ6227 08
END-EXEC NOT FOUND *MANUAL UPDATE REQUIRED

ABJ6228 00
BLANK WHEN ZERO IS REMOVED

ABJ6229 08
ACTUAL KEY INCOMPATIBLE WITH FILE ORGANIZATION *MANUAL UPDATE
REQUIRED

ABJ6230 08
CONFIGURATION SECTION OUT OF ORDER. *MANUAL UPDATE REQUIRED

ABJ6231 08
VALUE SHOULD NOT START IN AREA A. *MANUAL UPDATE MAY BE REQUIRED

ABJ6233 00
ZEROS/ZEROES REPLACED.

ABJ6234 08
STRING INTO SAME AREA. *MANUAL UPDATE REQUIRED

ABJ6235 00
SUFFIX - DATANAME SAME AS PROGRAM NAME.

- ABJ6236 00**
LITERAL DELIMITER ADDED.
- ABJ6237 08**
LITERAL DELIMITER MISSING. *MANUAL UPDATE REQUIRED
- ABJ6238 08**
REFERENCE TO FIRST BLL CAN NOT BE CONVERTED. *MANUAL UPDATE REQUIRED
- ABJ6239 08**
COPYBOOK NAME MUST START WITH ALPHABETIC CHARACTER. *MANUAL UPDATE REQUIRED
- ABJ6240 08**
THE ON OVERFLOW PHRASE OF THE CALL STATEMENT WILL NOW EXECUTE UNDER MORE CONDITIONS. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6241 08**
COMPARISONS BETWEEN A SCALED INTEGER AND A NONNUMERIC WILL NOW BE PERFORMED DIFFERENTLY. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6242 08**
THIS STATEMENT WILL NO LONGER USE THE COLLATING SEQUENCE IN THE OBJECT-COMPUTER PARAGRAPH. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6243 08**
THE ON SIZE ERROR PHRASE WILL NO LONGER BE EXECUTED FOR INTERMEDIATE RESULTS. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6244 08**
BLL CELL DOES NOT REFERENCE A 01 LEVEL RECORD. VERIFY BLL CELL USAGE. *MANUAL UPDATE REQUIRED
- ABJ6245 08**
RECORD WITH INSUFFICIENT BLL CELLS AVAILABLE TO PROVIDE ADDRESSABILITY. *MANUAL UPDATE REQUIRED
- ABJ6246 08**
CONDITIONAL VARIABLE WILL NO LONGER BE RIGHT JUSTIFIED. *MANUAL UPDATE REQUIRED
- ABJ6247 08**
CONDITIONAL VARIABLE WILL NOW BE SET TO ZERO (NOT SPACES). *MANUAL UPDATE REQUIRED
- ABJ6248 08**
PICTURE CLAUSE OF CONDITIONAL VARIABLE HAS EDITING SYMBOLS. RESULTS WILL BE DIFFERENT. *MANUAL UPDATE REQUIRED
- ABJ6249 08**
THE COLON WILL NOW BE TREATED AS A SEPARATOR. RESULTS MAY BE DIFFERENT. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6250 08**
LOWERCASE CHARACTERS WILL NOW BE TREATED AS THEIR UPPERCASE EQUIVALENTS. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6251 08**
THE NON-COBOL CHARACTERS IN THE REPLACE CLAUSE WILL NOW BE DIAGNOSED. *MANUAL UPDATE REQUIRED
- ABJ6252 08**
DIFFERENT FILE STATUS VALUES WILL NOW BE RETURNED. *MANUAL UPDATE MAY BE REQUIRED

Messages

- ABJ6253 08**
RULES FOR AUGMENTING VARIABLES HAVE CHANGED. IF DEPENDENCIES BETWEEN VARIABLES EXIST, THEN *MANUAL UPDATE MAY BE REQUIRED
- ABJ6254 08**
PICTURE CLAUSE OF A RECEIVING FIELD CONSISTS OF A'S AND B'S - NO LONGER CLASSED ALPHABETIC *MANUAL UPDATE REQUIRED
- ABJ6255 08**
PICTURE CLAUSE OF A RECEIVING FIELD CONSISTS OF A'S AND B'S - NO LONGER PERMITTED. *MANUAL UPDATE REQUIRED
- ABJ6256 08**
CALL IDENTIFIER HAS A PICTURE CLAUSE CONSISTING OF A'S AND B'S - NO LONGER PERMITTED. *MANUAL UPDATE REQUIRED
- ABJ6257 08**
CANCEL IDENTIFIER HAS PICTURE CLAUSE CONSISTING OF A'S AND B'S - NO LONGER PERMITTED. *MANUAL UPDATE REQUIRED
- ABJ6258 08**
BLANK LINES AND COMMENT LINES IN TEXT THAT MATCH PSEUDO-TEXT ARE NOW TREATED DIFFERENTLY. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6259 08**
MODE SPECIFIC DECLARATIVES IN CONTAINED PROGRAMS NOW TAKE PRECEDENCE OVER THIS ONE. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6260 08**
ON OVERFLOW PHRASE CAN NOW BE INVOKED WHEN RUNNING UNDER CICS. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6261 08**
ON EXCEPTION PHRASE CAN NOW BE INVOKED WHEN RUNNING UNDER CICS. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6262 00**
'OR' HAS BEEN INSERTED BETWEEN THE DELIMITERS IN THE DELIMITED BY PHRASE.
- ABJ6263 00**
THE 'IS' HAS BEEN REMOVED FROM THE POINTER PHRASE.
- ABJ6264 08**
THE REVERSED OPTION IS NOW ONLY VALID FOR SINGLE REEL FILES. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6265 08**
PHRASES IN THE OCCURS CLAUSE ARE NOT IN THE CORRECT SEQUENCE. *MANUAL UPDATE REQUIRED
- ABJ6266 08**
THE FOR REMOVAL OPTION IS NOW TREATED AS A COMMENT. *MANUAL UPDATE MAY BE REQUIRED
- ABJ6267 08**
QUALIFIED INDEXES ARE NO LONGER PERMITTED. *MANUAL UPDATE IS REQUIRED.
- ABJ6269 00**
OLD OPTIONS REMOVED
- ABJ6270 00**
NEW COMPILER OPTION ADDED

ABJ6271 00

DATA NAME IS THE SAME AS THE PROGRAM NAME. SUFFIX HAS BEEN ADDED.

ABJ6272 04

DATE FORMAT CLAUSE NOT ADDED DATE FORMAT INCOMPATIBLE WITH DATA ITEM'S PICTURE CLAUSE

ABJ6273 04

DATE FORMAT CLAUSE NOT ADDED - NOT PERMITTED WITH BLANK WHEN ZERO CLAUSE

ABJ6274 04

DATE FORMAT CLAUSE NOT ADDED - NOT PERMITTED WITH JUSTIFIEDCLAUSE

ABJ6275 04

DATE FORMAT CLAUSE NOT ADDED - NOT PERMITTED WITH SIGN CLAUSE

ABJ6276 04

DATE FORMAT CLAUSE NOT ADDED - DATA ITEM ALREADY HAS ONE

ABJ6277 04

DATE FORMAT CLAUSE NOT ADDED - DATE FIELDS THAT ARE GROUP ITEMS MUST HAVE USAGE DISPLAY

ABJ6278 04

DATE FORMAT CLAUSE NOT ADDED - INCOMPATIBLE WITH SPECIFIED OR ASSUMED USAGE CLAUSE

ABJ6279 04

DATE FORMAT CLAUSE NOT ADDED - INCOMPATIBLE WITH EXTERNAL CLAUSE IN 01 ENTRY

ABJ6280 04

DATE FORMAT CLAUSE NOT ADDED - INCOMPATIBLE WITH EXTERNAL CLAUSE IN FD OR SD ENTRY

ABJ6281 04

DATE FORMAT CLAUSE ADDED

ABJ6284 04

* WARNING - COMPILER WARNING MESSAGES WILL BE GENERATED

ABJ6300 08

STATEMENT IS INVALID IN A CICS PROGRAM *** MANUAL UPDATE REQUIRED

ABJ6301 04

31 BIT ESA ADDRESSES WILL BE TREATED AS NEGATIVE NUMBERS: RESULTS MAY BE UNPREDICTABLE *** MANUAL UPDATE RECOMMENDED

ABJ6302 04

FIELD USED IN SET ADDRESS STATEMENT CHANGED TO USAGE IS POINTER

ABJ6304 00

COPYBOOK NAME IS NOW A LITERAL

ABJ6305 04

BACK-TO-BACK PARENTHESES REMOVED

ABJ6306 04

FILE SECTION ADDED

ABJ6307 08

CONFIGURATION SECTION OUT OF ORDER

Messages

ABJ6308 00
PERIOD REMOVED

ABJ6309 00
"IS" IS REMOVED

ABJ6310 08
END-OF-PAGE PHRASE NOT ALLOWED WITHOUT A LINAGE CLAUSE FILE
DESCRIPTION ENTRY *MANUAL UPDATE IS REQUIRED

ABJ6311 16
MORE THAN 999999 CHANGE RECORDS HAVE BEEN CREATED FOR A TOKEN -
POSSIBLE PROGRAM ERROR *** CONTACT IBM'S CCCA HELPLINE

ABJ6312 00
PERIOD ADDED AFTER DIVISION HEADER

ABJ6313 00
PERIOD ADDED AFTER SECTION HEADER

ABJ6317 00
SUPERFLUOUS "IF" REMOVED PARENTHESES ADDED

ABJ6401 08
UNEXPECTED END OF COPY STATEMENT. COPY STATEMENT NOT CONVERTED

ABJ6402 08
NESTED COPY STATEMENT WITHIN COPY WITH REPLACING PHRASE. COPY
STATEMENT NOT CONVERTED

ABJ6403 08
COPY STATEMENT WITH REPLACING PHRASE WITHIN A NESTED COPY. COPY
STATEMENT NOT CONVERTED

ABJ6404 08
COPY STATEMENT HAS INVALID SYNTAX. COPY STATEMENT NOT CONVERTED

ABJ6405 08
COPY STATEMENT HAS INVALID SYNTAX. MISSING "BY". COPY STATEMENT NOT
CONVERTED

ABJ6406 08
LIBRARY MEMBER WAS EMPTY.

ABJ6407 08
MEMBER NOT FOUND IN COPY LIBRARY.

ABJ6408 08
COPY STATEMENT HAS NULL OR INVALID PSEUDO-TEXT-1. COPY STATEMENT NOT
CONVERTED

ABJ6409 08
PSEUDO-TEXT ENDING DELIMITER "==" WAS MISSING. COPY STATEMENT NOT
CONVERTED

ABJ6410 08
A RIGHT PARENTHESIS MISSING IN AN IDENTIFIER SPECIFIED IN THE
REPLACING PHRASE. COPY STATEMENT NOT CONVERTED

ABJ6411 08
THE COPY LIBRARY WAS NOT FOUND

ABJ6412 08
COPY STATEMENT CAUSES RECURSION.

ABJ9001 00
&1 ERRORS FOUND DURING COMPILATION

Messages

Appendix D. LCP reserved words

With the exception of predefined data item names and LCP function names, this list identifies all reserved words of the LCP compiler. Only those in *bold italics* have a meaning to the LCP compiler. The other words in the list have no meaning to the LCP compiler, but if used they will elicit an error message from the compiler.

This appendix documents intended Programming Interfaces that allow the customer to write programs to obtain the services of CCCA.

See Appendix E, "Predefined data items," on page 175 for a list of predefined data items and Appendix F, "List of LCP functions," on page 187 for a list of LCP functions.

The following words are reserved for the LCP compiler. Do not confuse this list of words with the COBOL compiler's list of reserved words. For a complete list of COBOL reserved words, please refer to the appropriate COBOL *Language Reference* manual.

accept comp
access comp-3
acquire
comp-4
add computational
advancing
computational-3
after computational-4
all compute
alphabetic
configuration
also contains
alter control
alternate
controls
and control-area
apply copy
are core-index
area corr
areas corresponding
ascending
count
assign csp
at currency

LCP reserved words

attribute-data
c01

author
data
date

before date-compiled

blank date-written

block day

bottom
de

by debug-contents
debug-item

call debug-line

cancel debug-name

cd debug-sub-1

cf debug-sub-2

ch debug-sub-3

changed
debugging

character
decimal-point

characters
declaratives

clock-units
delete

close delimited

cobol delimiter

code depending

code-set
descending

collating
destination

column
detail

comma
disable

communication
display

divide indexed

division
indic

down indicate
drop indicator
duplicates
 indicators
dynamic
 initial
 initiate
egi input
else input-output
emi inspect
enable
 installation
end into
end-change
 invalid
end-of-page
 is
enter
environment
 just
eop justified
equal
error key
esi
every label
exception
 last
exclusive
 leading
exhibit
 left
exit length
extend
 less
 limit
fd limits
file lineage
file-control
 linage-counter
filler line
final lines

LCP reserved words

first line-counter
footing linkage
for local-data
format lock
from low-value
low-values
generate

giving memory
go merge
greater message
group mode
modules
heading
move
high-value multiple
high-values multiply
named
I-O native
I-O-control negative
identification next
if no
in *not*
index note
number reports
numeric requestor
rerun
object-computer reserve
occurs reset
of return
off reversed

omitted
 rewind

on rewrite

open rf

optional
 rh

or right

organization
 rolling

output
 rounded

overflow
 run

page same

page-counter
 sd

perform
 search

pf section

ph security

pic segment

picture
 segment-limit

plus select

pointer
 send

position
 sentence

positive
 separate

printing
 sequence

procedure
 sequential

procedures
 set

proceed
 sign

program
 size

program-id
 sort

LCP reserved words

sort-merge

queue source

quote source-computer

quotes
space
spaces

random
special-names

rd standard

read standard-1

receive
start

record starting

records
status

redefines
stop

reel string

references
subtract

relative
sub-queue-1

release
sub-queue-2

remainder
sub-queue-3

removal
sum

renames
suppress

replacing
symbolic

report sync

reporting
synchronized

system-console
upon

system-shutdown
upsi-0
upsi-1

table upsi-2

tallying
upsi-3

tape *upsi-4*
terminal
 upsi-5
terminate
 upsi-6
text *upsi-7*
than *usage*
then *use*
through
 using

thru
time *value*
times *values*
to *varying*
top
trace *when*
trailing
 with
transaction
 words
true *working-storage*
type *write*

unit *zero*
unstring
 zeroes
until *zeros*
up

LCP reserved words

Appendix E. Predefined data items

This appendix documents intended Programming Interfaces that allow the customer to write programs to obtain the services of CCCA.

The following list describes predefined data items you can use in LCPs.

Access to data contained in these data items is available:

- At all times
- As a result of using an LCP function

Note: Do not code Data Division statements in your LCPs for predefined data items.

Name	Description
ACCESS-FILE-MODE X(1) FILE record	Access-type: I Indexed S Sequential R Relative D Dynamic
ADD-GROUP X(30) CHANGE data set	Used to define the concatenation of data; ADD-LENGTH, ADD-TEXT.
ADD-LENGTH 9(2) CHANGE data set	Length of token to be added or modified.
ADD-TEXT X(30) CHANGE data set	Text to be added or modified.
ASCII-FILE X(1) FILE record	Y if the file has ASCII-data.
ASSOCIATE NAME X(30) COPY record	Data name defined in the original source program being copied.
BLL-NAME X(30) CICS file	Name of BLL found in the Linkage Section.
BYPASSED-REF-MOD X(1) Interpreter	Identifies when Reference modification has been bypassed as a result of a call to the function BYPASS-IDENTIFIER. Y Reference modification has been bypassed N Reference modification was not present Byte 3 of BYPASSED-REF-TYPES.
BYPASSED-REF-QUAL X(1) Interpreter	Identifies when qualification has been bypassed as a result of a call to the function BYPASS-IDENTIFIER. Y Qualification of the data item has been bypassed N Data Item was not qualified Byte 1 of BYPASSED-REF-TYPES.
BYPASSED-REF-TYPES X(3) Interpreter	A concatenation of BYPASSED-REF-QUAL, BYPASSED-REF-SUB and BYPASSED-REF-MOD. Identifies what has been bypassed by a call to the function BYPASS-IDENTIFIER.

Predefined data items

Name	Description
BYPASSED-REF-SUB X(1) Interpreter	Identifies when subscripting and indexing have been bypassed as a result of a call to the function BYPASS-IDENTIFIER. Y Subscripting or indexing of the data item has been bypassed N Data Item was not subscripted or indexed. Byte 2 of BYPASSED-REF-TYPES.
CALL-NAME X(30) CALL file	Name of program to be called.
CHARACTER-STRING X(1)	Reserved.
CICS-RECORD-NAME X(30) CICS file	Name of record pointed by BLL-NAME.
COBOL-STANDARD X(5) OPTION record	<p>The level of the COBOL to be converted: Note: <i>L/Level</i> refers to the Source Language Level that you specify on the Language Level panel (see Figure 8 on page 17).</p> <p>ANS68 ANS 68: DOS/VS COBOL LANGLVL(1), or OS/VS COBOL LANGLVL(1) (L/Levels 1 and 3)</p> <p>ANS74 ANS 74: DOS/VS COBOL LANGLVL(2), OS/VS COBOL LANGLVL(2), VS COBOL II Release 1.0, 1.1, 2.0, or any COBOL with the CMPR2 option (L/Levels 2, 4, and 5)</p> <p>ANS85 ANS 85: VS COBOL II NOCMR2 Release 3.0, 3.1, 3.2, VS COBOL II NOCMR2 Release 4.0, COBOL/370 NOCMR2, COBOL for VSE/ESA NOCMR2, COBOL for MVS & VM NOCMR2, or COBOL for OS/390 & VM NOCMR2 (L/Levels 6, 7, 8, 9, 10, 11) Enterprise COBOL for z/OS</p>

Name	Description
COBOL-TYPE X(6) OPTION record	<p>Indicates the type of source COBOL to be converted: Note: <i>L/Level</i> refers to the Source Language Level that you specify on the Language Level panel (see Figure 8 on page 17).</p> <p>DOS/VS DOS/VS COBOL (L/Levels 1 and 2)</p> <p>OS/VS OS/VS COBOL (L/Levels 3 and 4)</p> <p>COBII VS COBOL II (any release before 4.0) (L/Levels 5 and 6)</p> <p>COBII4 VS COBOL II (Release 4.0) (L/Level 7)</p> <p>COB370 COBOL/370 NOCMR2 (L/Level 8)</p> <p>COBVSE COBOL for VSE/ESA NOCMR2 (L/Level 9)</p> <p>COBMVS COBOL for MVS & VM NOCMR2 (L/Level 10)</p> <p>COB390 COBOL for OS/390 & VM NOCMR2 (L/Level 11)</p> <p>COBENT Enterprise COBOL for z/OS (pre version 5)</p>
CONSOLE-NAME X(6)	Reserved.
COPY-LIBRARY X(8)	Name of copy library.
COPY-LOCATION X(3) COPY record	<p>Indicates where the COPY member is used:</p> <p>EN Environment Division FS File Section LI Linkage Section WS Working-Storage Section PR Procedure Division.</p>
COPY-NAME X(10) COPY record	Name of COPY member.
COPY-POINTER 9(7)	Location of the last COPY statement.
DA-LOCATION 9(7) Interpreter	Location of token DATA in token file (Data Division).

Predefined data items

Name	Description
DATE-FORMAT X(8) OPTION record	Format of the date as generated in VSE for DOS/VS COBOL. It can be MM/DD/YY or DD/MM/YY.
DEVICE-FILE-NAME X(10)	Set to "DISK" or blank for REWRITE LCP.
DEVICE-OVERRIDE-01 X(2)	Reserved.
DEVICE-OVERRIDE-02 X(2)	Reserved.
EN-LOCATION 9(7) Interpreter	Location of token ENVIRONMENT in token file (Environment Division).
END-OF-COPY X(1) OPTION record	End-of-copy definition P First period after the word COPY L Source line end, containing the word COPY N Do not process.
EXTERNAL-FILE-NAME X(10) FILE record	Name of the external file name.
FIELD-SIZE 9(7)	Reserved.
FILE-CONVERSION X(1) FILE record	Y or N conversion required.
FILE-SEQUENCE-NO 9(2) Interpreter	+1 for each file then -1 if no file status added.
FILE-STATUS-NAME X(30) KEY record	Name of FILE-STATUS in COBOL program.
FIRST-TOKEN-POINTER 9(7) Interpreter	Location of the first token of the program.
IBM-SYSTEM X(2)	Reserved.
ID-LOCATION 9(7) Interpreter	Location of token IDENTIFICATION in token file. (Identification Division).
INPUT-FILE X(10)	Reserved.
INPUT-LIBRARY X(8)	Reserved.

Name	Description
INPUT-TEXT X(30) Interpreter	Data field used by the MOVE-LCP function from which characters are moved.
INTERNAL-FILE-NAME X(30) FILE record	File name in the COBOL program.
IO-LOCATION 9(7) Interpreter	Location of token INPUT-OUTPUT in token file (Input-Output Section).
LAST-TOKEN-POINTER 9(7) Interpreter	Location of the last token of the source program.
LCP-ALPHA X(10) Interpreter	Contains alphanumeric data for CONVERT-ALPHA-NUMERIC function.
LCP-NUMERIC 9(10) Interpreter	Contains numeric data after execution of CONVERT-ALPHA-NUMERIC function.
LENGTH-OF-MOVE 9(2) Interpreter	Number of characters to be moved from INPUT-TEXT to OUTPUT-TEXT by the MOVE-LCP function.
LINAGE X(1)	Linage clause found in File Section.
LI-LOCATION 9(7) Interpreter	Location of token LINKAGE in token file. (Linkage Section.)
LITERAL-SEPARATOR X(1) OPTION record	Separation character for nonnumeric literals: A Apostrophe Q Quotation mark.
MEMBER-NAME X(8) OPTION record	Source member name containing the program to be converted.
MESSAGE-ID X(7) Interpreter	Identifier for conversion messages.
NOMINAL-KEY-NAME X(30) KEY record	Name of field defining the NOMINAL KEY in the COBOL program.
NUMERIC- <i>nn</i> 9(10) Interpreter	Ten data elements used to save numeric values. Note: Do not use these data items for your LCPs; they are used by the supplied LCPs. For your LCPs, use USER-NUMERIC- <i>nn</i> .
OBJECT-COMPUTER-NAME X(30)	Reserved.
OLD-ORGANIZATION- -FILE-MODE X(1) FILE record	File organization: A: D: I: R: S: U: W:

Predefined data items

Name	Description
OLD-PROGRAM-NAME X(30) PROGRAM record	COBOL program name before conversion if name changed.
OPTION-CICS X(1) OPTION record	Indicator from panel for enabling CICS command conversion.
OPTION-nn X(1) OPTION record	Fifteen indicators used to control optional conversion processing. Values: Y, N.
ORGANIZATION-FILE-MODE X(1) FILE record	File-organization: I Indexed S Sequential R Relative
OUTPUT-FILE X(10)	Reserved.
OUTPUT-LIBRARY X(8)	Reserved.
OUTPUT-TEXT X(30) Interpreter	Data field used by the MOVE-LCP function to store characters moved from the INPUT-TEXT field.
PROGRAM-NAME X(10) PROGRAM record	Name of the converted COBOL program. This name will appear after PROGRAM-ID in the converted program.
PROGRAM-STATUS X(10) PROGRAM record	Save field for information about the conversion of the COBOL program. For example: COMPLETE, ERROR, and WARNING.
PR-LOCATION 9(7) Interpreter	Location of token PROCEDURE in token file. (Procedure Division).
RECEIVING-CHARACTER 9(2) Interpreter	Position of the first character in the OUTPUT-TEXT field to be replaced by the MOVE-LCP function.
RECORD-KEY-NAME X(30) KEY record	Name of field defining the RECORD KEY in the COBOL program.
RECORD-NAME X(30) RECORD record	Record name.
RELATIVE-KEY-NAME X(30) KEY record	Name of field defining the RELATIVE KEY in the COBOL program.

Name	Description
RETURN-CODE X(2) Interpreter	<p>Updated by the interpreter, giving the return code after each I/O operation on all the logical files:</p> <p>00 Successful operation</p> <p>23 Record not found (after READ or READ-NEXT)</p> <p>24 File full (after WRITE).</p> <p>For other return code values, refer to the STATUS KEY values listed in the COBOL <i>Language Reference</i> manual.</p>
SELECT-LOCATION 9(7)	Location of SELECT clause.
SEQUENCE-STATUS-NO 9(2) FILE record	Sequence number of the definition of the file described in the converted COBOL program.
SOURCE-COMPUTER-NAME X(30)	Reserved.
SP-LOCATION 9(7) Interpreter	Location of token SPECIAL-NAMES in token file.
STARTING-CHARACTER 9(2) Interpreter	Number of first character to be moved from the INPUT-TEXT field by the MOVE-LCP function.
STARTING-POSITION 9(2) CHANGE data set	<p>Start position of TOKEN in the converted statement.</p> <p>Position 1 is equal to column 8 in a COBOL statement.</p>
STRING-DELIMITER X(1) Interpreter	Character used by the STRING-LCP and UNSTRING-LCP functions to concatenate character strings or to separate character strings.
STRING-LENGTH 9(2) Interpreter	Length of string in STRING-TEXT after execution of the STRING-LCP function.
STRING-TEXT X(30) Interpreter	Field used by the STRING-LCP and UNSTRING-LCP functions.
STRING-WORD- <i>nn</i> X(30) Interpreter	Ten fields used by the UNSTRING-LCP function to store a character string extracted from STRING-TEXT and used by the STRING-LCP function to define STRING-TEXT.
STRING-WORDS	<p>The 01-level item for the STRING-WORDS-<i>nn</i> fields. Before using the STRING-LCP function, initialize the STRING-WORDS-<i>nn</i> fields by moving SPACES to STRING-WORDS.</p> <p>Reserved.</p>
SUBSCRIPT1-WORDS X(30)	Reserved.
SUBSCRIPT2-WORDS X(30)	Reserved.
SUBSCRIPT1- <i>nn</i> X(30) Interpreter	Ten fields used to save subscripts or indexes defined in the COBOL program

Predefined data items

Name	Description
SUBSCRIPT2- <i>nn</i> X(30) Interpreter	Ten fields used to save subscripts or indexes defined in the COBOL program
TARGET-LANGUAGE X(5) OPTION record	<p>Indicates the target COBOL language that the program is being converted to.</p> <p>COBII = VS COBOL II CBVSE = COBOL for VSE/ESA CBIBM = IBM COBOL (COBOL for MVS & VM, COBOL for OS/390 & VM) CBENT = Enterprise COBOL for z/OS and OS/390</p>
TEXT- <i>nn</i> X(30) Interpreter	<p>Ten fields used to save alphanumeric values. Note: Do not use these data items for your LCPs; they are used by the supplied LCPs. For your LCPs, use USER-TEXT-<i>nn</i>.</p>
TOKEN-CHANGE-CODE 9(3) TOKEN data set	<p>Indicates what LCP (if any) CCCA invokes when to convert the associated token:</p> <p>999 CCCA does not invoke an LCP. 990 CCCA invokes an LCP that has the token in its CONVER statement. <i>nnn</i> (other than 999 and 990) CCCA invokes an LCP that has LCP-<i>nnn</i> in its CONVER statement.</p> <p>The following list shows the change codes used by CCCA, and the change codes you can use for your own LCPs:</p> <p>000, 860-989, 992-998 Used by CCCA, or reserved for use</p> <p> These LCPs are invoked by internal CCCA programs, not by reserved words.</p> <p> You cannot enter these values in the Change code field.</p> <p>001-799 Available for your own LCPs.</p> <p>800-859 Used by supplied LCPs.</p> <p>991 Used by CCCA.</p> <p> LCP991 is invoked both by reserved words and internal CCCA programs.</p>
TOKEN-FLAG-01 X(1) TOKEN data set	<p>Word type defined in COBOL Reserved Word data set:</p> <p>1 Section or paragraph name 2 Start of a clause 3 Start of a statement 5 Start of a phrase 9 Reserved for compiler, no meaning.</p>
TOKEN-FLAG-02 X(1) TOKEN data set	<p>Used in the same manner as TOKEN-FLAG-01, for example, where the token is either a statement, a clause, or a section name.</p>

Name	Description
TOKEN-FLAG X(2) TOKEN data set	Concatenates TOKEN-FLAG-01 and TOKEN-FLAG-02 Example: TOKEN-FLAG-01 = 1 TOKEN-FLAG-02 = 3 then TOKEN-FLAG = 13.
TOKEN-LENGTH 9(3) TOKEN data set	Length of token.
TOKEN-LINE-CODE X(1)	Reserved.
TOKEN-MESSAGE-ID X(7)	Reserved.
TOKEN-POINTER 9(7) Interpreter	Position of token in token file.
TOKEN-POSITION 9(2) TOKEN data set	Location of the first character of the token in the source statement. Note: Position 1 is equal to column 8 in the COBOL source statement.
TOKEN-SEQUENCE X(6) TOKEN data set	Source statement number, containing token.
TOKEN-SOURCE X(1) TOKEN data set	Source of token: C Token contained in a COPY member P Token contained in a program statement.
TOKEN-TEXT X(30) TOKEN data set	Character string containing the token. With a literal of more than 30 characters, the value of TOKEN-TEXT in the token file is blank.
TOKEN-TYPE-CODE X(1) TOKEN data set	Token-type code: C COPY statement (Element) L Nonnumeric literal (Token) N Numeric literal (Token) P Data-description (PICTURE) (Token) W Word (Token) / Command (Element) * Comment or element. (Element)
UPDATE X(08) Interpreter	The date of the conversion. The format is MM/DD/YY.
UPDATE-FILE-FLAG X(1) FILE record	Flag used when the file is open in input/output mode.
USER-NUMERIC-nn 9(10) Interpreter	Ten fields available to user written LCPs for saving numeric values.

Predefined data items

Name	Description
USER-TEXT-nn X(30) Interpreter	Ten fields available to user written LCP for saving alphanumeric values.
UTIME X(08) Interpreter	The time of the conversion. The format is HH:MM:SS.
VSAM-ORGANIZATION X(1) FILE record	Y or N if VSAM.
WHERE-USED X(3) Interpreter	Used to save the location of the token in the COBOL program: EN Environment Division FS File Section ID Identification Division IO Input-Output Section LI Linkage Section. PR Procedure Division RP Report Section WS Working-Storage Section
WORD-SUFFIX X(02) OPTION record	TWO numeric characters used to change reserved word used as data name.
WORD-SUFFIX-COUNT 9(4)	Reserved.
WORK-KEY-nn X(30) WORK file	Name of field containing the access key for WORK-nn file.
WORK-NUMERIC-nn 9(7) WORK file	Name of field containing a numeric work value for WORK-nn file.
WORK-NUMERIC2-nn 9(7) WORK file	Name of field containing a numeric work value for WORK-nn file.
WORK-TEXT-nn X(30) WORK file	Name of field containing a work value for WORK-nn file.
WORK-TEXT2-nn X(30) WORK file	Name of field containing a work value for WORK-nn file.
WORK-TYPE-nn X(3) WORK file	Name of field containing a work value for WORK-nn file.
WORK-TYPE2-nn X(3) WORK file	Name of field containing a work value for WORK-nn file.

Name	Description
WS-LOCATION 9(7) Interpreter	Position of the token WORKING-STORAGE in the token file.

Predefined data items

Appendix F. List of LCP functions

This appendix documents intended Programming Interfaces that allow the customer to write programs to obtain the services of CCCA.

The following list describes functions you can use in LCPs.

Note: If debugging for an LCP is activated (see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71), the **Op codes** in the following list appears in the LCP OPCODE column of the debug listing.

Name	Op code	Description
ADD-CALL	AD-CL	Write a CALL record to the logical CALL file.
ADD-COPY	AD-CY	Write a COPY record to the logical COPY file.
ADD-FILE	AD-FL	Write a FILE record to the logical FILE file.
ADD-KEY	AD-KY	Write or update a KEY record to the logical Key file.
ADD-PROGRAM	AD-PR	Write a PROGRAM record to the logical PROGRAM file.
ADD-RECORD	AD-RC	Write a RECORD record to the logical RECORD file.
ADD-WORK-nn	AD-nn	Write a WORK-nn record to the logical WORK-nn file.
BYPASS-IDENTIFIER	BYID	Bypass the identification of a data name (that is, qualifier, subscript, index, and reference modifier)
BYPASS-POINTER	BYPN	Bypass conversion process associated with the token currently in storage.
COMMENT	CM	Put an * in column 7 in the source statement of the token.
CONVERT-ALPHA-NUMERIC	ALNUM	Convert alphanumeric character string to numeric.
DETERMINE-LENGTH	DTLN	Calculate the length of data in ADD-TEXT. Result in ADD-LENGTH.
DIAGNOSTIC	DG	Place message text contained in ADD-TEXT into the message summary which appears at the end of the conversion diagnostics listing.
EDIT-MESSAGE	EDMSG	Used to write informational and error messages to the conversion listing.
EJECT	EP	Put a / in column 7 of the generated source statement.
GET-FIRST GET-FIRST-TOKEN	GTFRT	Retrieve first token from the TOKEN data set.
GET-LAST GET-LAST-TOKEN	GTLST	Retrieve last token from the TOKEN data set.
GET-NEXT GET-NEXT-TOKEN	GTNXT	Retrieve next token from the TOKEN data set.
GET-PREVIOUS GET-PREVIOUS-TOKEN	GTPRT	Retrieve previous token form the TOKEN data set.

LCP functions

Name	Op code	Description
GET-TOKEN	GTTKN	Retrieve specified token or element from the TOKEN data set.
GET-NEXT-ELEMENT	GTNXE	Retrieve next token or element from the TOKEN data set.
GET-PREVIOUS-ELEMENT	GTPRE	Retrieve previous token or element from the TOKEN data set.
GET-ELEMENT	GTELM	Retrieve specified token or element from the TOKEN data set.
INSERT-AFTER INSERT-AFTER-TOKEN	INAF	Insert a token or element after current token or element.
INSERT-BEFORE INSERT-BEFORE-TOKEN	INBF	Insert a token or element before current token or element.
MAINTAIN-LINE-POSITION	MNLNP	If possible, maintain the physical position of the token or element in the line of generated source.
MOVE-LCP	MVLCP	Move characters within predefined fields.
READ-CICS	RD-CI	Read a CICS record.
READ-FILE	RD-FL	Read a FILE record.
READ-KEY	RD-KY	Read a KEY record.
READ-WORK-nn	RD-nn	Read a WORK-nn record.
READ-NEXT-FILE	RN-FL	Read next FILE record.
READ-NEXT-RECORD	RN-RC	Read the next RECORD record for the current FILE record.
READ-NEXT-WORK-nn	RN-nn	Read the next WORK-nn record.
REMOVE REMOVE-TOKEN	RM	Remove current token.
REMOVE-CLAUSE	RMCL	Remove token currently in storage and the tokens following that are part of the same clause.
REMOVE-NEXT REMOVE-NEXT-TOKEN	RMNXT	Remove token following the token currently in storage.
REMOVE-SUFFIX	RMSUF	Remove the reserved word suffix added in pass 1.
REMOVE-STATEMENT	RMST	Remove token currently in storage and the tokens following that are part of the same statement.
REPLACE REPLACE-TOKEN	RP	Replace the current token with the value held in ADD-TEXT.
RETRIEVE-FILE	RT-FL	Set and read a FILE record according to the RECORD record currently in storage.
SETLL-FILE	ST-FL	Set FILE file to the first record.
SETLL-RECORD	ST-RC	Set logical RECORD file to first RECORD record for the current FILE record.
SETLL-WORK-nn	ST-nn	Set logical WORK-nn file to the first WORK-nn record.
SPLIT-LINE	SPLN	Start a new line after the token currently in storage.
STRING-LCP	STLC	Concatenate fields into a single character string.

Name	Op code	Description
SUFFIX SUFFIX-TOKEN	SF	Insert after current token, leaving no blank between the current token and the token being added.
UNSTRING-LCP	UNLC	Separate a character string into parts delimited by a specified delimiter.
UPDATE-FILE	UP-FL	Update logical FILE file for FILE record currently in storage.
UPDATE-WORK- <i>nn</i>	UP- <i>nn</i>	Update logical WORK- <i>nn</i> record currently in storage.

LCP functions

Appendix G. LCP directory

This appendix lists the supplied Language Conversion Programs (LCPs), with a brief description of the processing performed by each one.

LCPs fall into one of five categories:

1. LCPs that convert CICS commands
2. LCPs that convert COBOL statements
3. LCPs that partially convert COBOL statements
4. LCPs that flag COBOL statements
5. LCPs that set information for other LCPs

For a more complete description of the conversion and flagging of the language elements performed by the LCPs see Appendix A, "Converted COBOL language elements," on page 117.

Converted CICS commands

EXEC BLL references changed to ADDRESS OF

SERVICE RELOAD

Replaced by CONTINUE

ADD (851)

BLL references changed to POINTER facilities

COMPUTE

BLL references changed to POINTER facilities

MOVE

BLL references changed to POINTER facilities

SUBTRACT

BLL references changed to POINTER facilities.

Completely converted COBOL statements

ACTUAL

ACTUAL KEY clause; replaced by RELATIVE

ALPHABETIC

Changed to ALPHABETIC-UPPER

APPLY

Remove APPLY clause in I-O-CONTROL paragraph

ASSIGN

Change ASSIGN clause syntax

CBL Modify compiler options for COBOL/370

COPY Converts COBOL Standard 68 syntax and adds COPY information

CORR/CORRESP

Multiple MOVE changed to separate MOVES

- CURRENT-DATE**
Replaced by DATE special register and reformatted or by EXEC CICS ASKTIME in a CICS program
- DATE** Add hyphen if missing in DATE COMPILED and DATE WRITTEN
- DATE-COMPILED**
Add period after header if missing.
- DISP** In OPEN and CLOSE statements option deleted
- ENTER**
Obsolete element removed.
- ENVIRONMENT**
Add Configuration Section header if it is needed; relocate it if it is in the wrong place.
- EXAMINE**
Change EXAMINE to INSPECT
- EXHIBIT**
EXHIBIT statement changed to DISPLAY
- FD** Convert FD entry, check LABEL clause
- FILE-LIMIT**
Delete FILE-LIMIT clause (COBOL 68 Standard)
- FILE-LIMITS**
Delete FILE-LIMITS clause (COBOL 68 Standard)
- JUST** Value literal is changed for COBOL 68 Standard syntax
- JUSTIFIED**
Value literal changed for COBOL 68 Standard syntax
- LEAVE**
In OPEN statement option deleted
- LINE/LINES**
Word removed in WRITE BEFORE/AFTER ADVANCING mnemonic
- LVL88** Put 88 level value string in quotes, if missing
- MEMORY**
Remove MEMORY SIZE clause if **Remove obsolete elements** field on Conversion Options panel 2 is set to Y
- MOVE (851)**
Add reference modification to variable length receivers that contain their ODO object
- MULTIPLE**
For multiple reel/unit COBOL 68 Standard CLAUSE deleted
- NATIVE**
Add ALPHABET word in SPECIAL-NAMES
- NOMINAL**
Replaced by RELATIVE or clause deleted
- NOTE** Change to comment
- OPEN** Add FILE STATUS test for VSAM files that have had FILE STATUS clauses added

OTHERWISE

Clause of IF statement replaced by ELSE

POSITIONING

AFTER POSITIONING clause of WRITE statement replaced by AFTER
ADVANCING clause

PROCEDURE

An Error Declaratives Section is added for each file that is to be converted
to VSAM.

PROCESSING

Delete PROCESSING MODE clause (COBOL 68 Standard)

READ MOVE NOMINAL TO RECORD KEY for ISAM files

Add reference modification to variable length receivers
that contain their own ODO object

REDEFINES

Remove clause in FD

RELEASE

Add reference modification to variable length receivers that contain their
own ODO object

REMARKS

Change to a comment

REREAD

In OPEN statement option deleted

RESERVE

Change RESERVE syntax COBOL 68 Standard

RETURN (851)

Add reference modification to variable length receivers that contain their
own ODO object

REWRITE

MOVE NOMINAL KEY TO RECORD KEY for ISAM files
Add reference modification to variable length receivers
that contain their own ODO object

SAME Change SAME AREA to SAME RECORD AREA**SD** Conver SD ENTRY, LABEL clause**SEARCH**

SEARCH WHEN KEY

SEEK Statement deleted**SEQUENCE**

Add ALPHABET word in SPECIAL-NAMES

SPECIAL-NAMES

Add SPECIAL NAMES

STANDARD-1

Add ALPHABET word in SPECIAL-NAMES

START

MOVE NOMINAL TO RECORD KEY

THAN

Removed after > or < relational operators

LCP directory

THEN Delete THEN between statements

TIME-OF-DAY

Replaced by TIME special register or by an EXEC CICS ASKTIME (in a CICS program) and reformatted.

TRACK-AREA

TRACK-AREA removed

TRANSFORM

Replaced by INSPECT statement

UNSTRING

Add reference modification to variable length receivers that contain their own ODO object.

UPSI-0 (850)

Replace UPSI switch by condition name

UPSI-1 (850)

Replace UPSI switch by condition name

UPSI-2 (850)

Replace UPSI switch by condition name

UPSI-3 (850)

Replace UPSI switch by condition name

UPSI-4 (850)

Replace UPSI switch by condition name

UPSI-5 (850)

Replace UPSI switch by condition name

UPSI-6 (850)

Replace UPSI switch by condition name

UPSI-7 (850)

Replace UPSI switch by condition name

USING

START...USING KEY USING word deleted

VALUE

Remove sign if PICTURE unsigned

VALUES

Changed to VALUE if not used in level 88

WHEN-COMPILED

WHEN-COMPILED special register output reformatted (OS/VS COBOL only)

WRITE

MOVE NOMINAL KEY TO RECORD KEY for ISAM files;
add reference modification to variable length receivers
that contain their own ODO.

COBOL statements converted with warning

NOT Change abbreviated relation condition COBOL 68 Standard syntax

ON ON integer changed to IF
ON integer UNTIL integer changed to IF
other cases flagged

COBOL statements flagged

The flagged COBOL statements may be put in several categories:

1. Language elements from functions of the source language that are no longer supported in the target languages and have no replacement or equivalent in the target languages. Therefore, a conversion cannot be performed.
 - a. Communication Facility (OS/VS COBOL only)

COMMUNICATION

Communication Section header flagged

COUNT

ACCEPT MESSAGE COUNT statement flagged

DISABLE

DISABLE statement flagged

ENABLE

ENABLE statement flagged

RECEIVE

Receive statement flagged

SEND Send statement flagged

- b. Report Writer section (flagged if **Flag Report Writer statements** field on Conversion Options panel 2 is set to Y)

GENERATE

Generate statement flagged

INITIATE

Initiate statement flagged

LINE-COUNTER (855)

Flagged

PAGE-COUNTER (855)

Flagged

PRINT-SWITCH (855)

Flagged

REPORT

Flagged

REPORTS

Flagged

TERMINATE

Statement flagged

USE Flagged USE BEFORE REPORTING

2. Other cases

ALL Flag MOVE ALL (if COBOL 68 Standard syntax)

ALTER

SEGMENTATION - flag

CALL Flagged if the identifier has a PICTURE string that consists of A's and B's only; CALL...USING procedure name/VSAM file name statements are flagged;

- CANCEL**
Flagged if there is an identifier in the statement with a PICTURE string that consists of A's and B's only; procedure name/VSAM file name statements are flagged
- CURRENCY**
Flag COBOL 68 Standard CURRENCY SIGN clause
- DEBUGGING**
USE FOR DEBUGGING flag if not procedure name
- DIVIDE**
Flag ON SIZE ERROR when multiple receivers
- IN** Flag qualified indexes
- INITIALIZE**
Flag INITIALIZE...REPLACING ALPHABETIC/ALPHANUMERIC-EDITED if there are receiving fields with PICTURE strings that consist of A's and B's only.
- INSPECT**
Flagged if the PROGRAM COLLATING SEQUENCE established in the OBJECT COMPUTER paragraph identifies an alphabet defined with the ALSO clause
- LABEL RECORD**
Data name changed to STANDARD
- MULTIPLY**
Flag ON SIZE ERROR when there are multiple receivers
- NSTD-REELS**
Flag references to NSTD-REELS special register
- OCCURS**
Flags if phrases of OCCURS clause are in non-standard order
- OF** Flag qualified indexes
- PIC** Check scaled variables
- PICTURE**
Check scaled variables
- REPLACE**
Flagged if COBOL 85 Standard source
- RELATIVE**
Check if PICTURE of relative key has scaling position
- STRING**
Statement flagged if the receiver has a PICTURE string that consists of A's and B's only;
- TOTALING/ TOTALED AREA**
In LABEL clause deleted
- TRACE**
READY/RESET TRACE statement deleted
- TRACK-LIMIT**
TRACK-LIMIT removed
- TRUE** SET...TO TRUE statement flagged if COBOL Standard 85 standard behavior is different.

UNSTRING
UNSTRING DELIMITED BY ALL flag (if COBOL 68 Standard)

USE GIVING phrase removed in USE AFTER STANDARD

LCPs corresponding to information

ACCESS
Update FILE information in CONTROL file

ASCENDING
Save key ID for SEARCH ... WHEN

DECLARATIVES
Check section end

DEPENDING
Save name of object of ODO

DESCENDING
Save key ID for SEARCH ... WHEN

END-OF-CONVERSION-1
Add WRITE ... AFTER ADVANCING section

END-OF-CONVERSION-2
Add data items in WS

END-OF-CONVERSION-3
Add data items in WS

END-OF-CONVERSION-4
Add SPECIAL NAMES

END-OF-CONVERSION-5
List data names to be checked

ENVIRONMENT
Set flag when entering Environment Division

ID Set flag when entering ID Division

IDENTIFICATION
Set flag when entering ID Division

INDEXED
Store Indexes on Work file; used by the IN and the OF LCP

INPUT-OUTPUT
Set flag when entering I/O Section

LINKAGE
Set flag when entering Linkage Section

PROGRAM-ID
Update PROGRAM FILE

RECORD
Update key record

SECTION
Set flag when entering a section

SELECT
Update CONTROL file

LCP directory

WORKING-STORAGE

Set flag entering WS SECTION

01 Save RECORD name of FD

Appendix H. Sample output

This appendix contains sample output generated by CCCA.

Program/File report

```
5648-B05 V2R1      - IBM COBOL CONVERSION AID -  SAMPLE RUN          17 APR 1998 18:45:39   Page   1
..... PROGRAM -- FILE REPORT .....
C
---COBOL---        D L I  -----OPTIONS-----          ----- FILES DEFINED -----
PGM.NAME REV  PBR SUFF E V C      1 11111 MEMBER    STATUS      OLD NEW CNV SYSTEM  COBOL
CNV WORD  L L S  12345 67890 12345 NAME      DATE/TIME  ORG ORG REQ NAME   NAME

ABJIVP01   03  213   0  Q 1  N  YYYYY YNNNN NNNNN ABJIVP01  COMPLETE
                                           98/04/16 18:26
                                           COMPIL RC=00
                                           98/04/16 18:26
                                           MANUAL COMPLETION
                                           / / :
                                           S  S  N  DDPRINT  PRINT-FILE

ABJIVP02   04  208   2  Q 1  N  YYYYY YNNNN NNNNN ABJIVP02  COMPLETE
                                           98/04/16 18:12
                                           COMPIL RC=04
                                           98/04/16 18:12
                                           MANUAL COMPLETION
                                           / / :
                                           S  S  N  PRINT   PRINT-OUT

ABJIVP03   02  875   0  Q 1  Y  YYYYY NNNNN NNNNN ABJIVP03  COMPLETE
                                           98/04/16 18:04
                                           MANUAL COMPLETION
                                           / / :
                                           S  S  N  PRINT   PRINT-OUT

DIRECT1    02   42   0  A 2  N  YYYYY NNNNN NNNNN DIRECT1  WARNING
                                           98/04/16 18:32
                                           MANUAL COMPLETION
                                           / / :
                                           R  R  Y  MASTER1  DA-FIL1
                                           S  S  N  MASTER2  DA-FIL2
                                           D  R  Y  MASTER3  DA-FIL3
                                           W  R  Y  MASTER4  DA-FIL4
                                           A  R  Y  MASTER5  DA-FIL5
                                           U  R  Y  MASTER6  DA-FIL6
                                           I  I  Y  MASTER7  DA-FIL7
                                           U  R  Y  TAT1     B1
                                           R  R  Y  TAT2     B2
                                           S  S  N  TAT3     B3
                                           D  R  Y  TAT4     B4
                                           W  R  Y  TAT5     B5
                                           A  R  Y  TAT6     B6

.....          E N D   O F   R E P O R T          .....
```

File/Program report

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE RUN          17 APR 1998 18:46:21   Page   1
.....  F I L E  -- P R O G R A M  R E P O R T  .....
SYSTEM      PROGRAM      ORG  CONVERSION  COBOL
NAME        NAME          REQUIRED  NAME

ANSWERS     LCPTST03                NO    PRINT-OUT
APRINTER    LCPI0101                NO    PRINT-FILE
            LCPI0105                S    NO    OUTPUT-DEVICE
            LCPTST05                S    NO    PRINT-FILE
            LCPTST10                S    NO    PRINT-FILE
            LCPTST11                NO    OUTPUT-LINE
            LCPTST12                S    NO    PRINT-LINE
            LCPTST13                S    NO    PRINT-LINE
            LCPTST14                S    NO    PRINT-FILE
            LCPTST15                S    NO    A-LINE
            LCPTST16                S    NO    A-LINE
CPNN0001    CPGM0001                S    NO    CADD010T1
CPNN0002    CPGM0002                S    NO    CSD-ONLINE-RECORD-SORT-FILE
CPNN0003    CPGM0003                S    NO    CSD-DATABASE-CONTROL-FILE
CPNN0004    CPGM0004                S    NO    CSD-ONLINE-MASTER-FILE
DDPRINT     ABJIVP01                S    NO    PRINT-FILE
DUM         BLGSA01                 S    NO    SORTFILE
EIPARM      EI030BPF                I    YES   EIPARM
IBDAM       LCPI0105                R    YES   BDAM-IN
            LCPI0107                R    YES   BDAM-IN
INFILE      INDEX                   S    NO    CARD-FILE
INFPRI     INFF0101                S    NO    REPORT-FILE
IOBDAM      LCPI0105                R    YES   BDAM-IO
            LCPI0107                R    YES   BDAM-IO
ISAM01A     LCPI0101                I    YES   QISM-OUT
            LCPI0101                I    YES   QISM-IN
ISAM07A     LCPI0106                I    YES   QUISAM
ISAM08A     LCPI0106                I    YES   QUISAMX
ISAM09A     LCPI0106                I    YES   BYSAM
MASTER      INDEX                   I    YES   IS-FILE
            LCPTST04                I    YES   IS-FILE
            LCPTST07                I    YES   IS-FILE
MASTER1     DIRECT1                 R    YES   DA-FIL1
MASTER2     DIRECT1                 S    NO    DA-FIL2
MASTER3     DIRECT1                 R    YES   DA-FIL3
PRINT       ABJIVP02                S    NO    PRINT-OUT
.....          E N D    O F    R E P O R T    .....

```

Copy/Program report

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE RUN          17 APR 1998 18:47:38   Page   1
.....  C O P Y  -- P R O G R A M  R E P O R T  .....
COPY       PROGRAM      LOCATION  ASSOCIATED
NAME       NAME          NAME

ABJCIOUT   ABJIVP03                LINKAGE SECTION MAP13I
ABJCQIN    ABJIVP03                WORKING-STORAGE MAP1I
ABJCQOUT   ABJIVP03                LINKAGE SECTION MAP11I
ABJERRMP   ABJIVP03                LINKAGE SECTION MAP12I
ABJL901    ABJIVP02                FILE SECTION  OUTPUT-RECORD
ABJL902    ABJIVP02                FILE SECTION
ABJL903    ABJIVP02                WORKING-STORAGE NUM-OF-ITEMS
ABJL903A   ABJIVP02                WORKING-STORAGE
ABJL904    ABJIVP02                WORKING-STORAGE
DFHAID     ABJIVP03                WORKING-STORAGE DFHAID
DFHBL LDS  ABJIVP03                LINKAGE SECTION DFHBL LDS
DFHBMSCA   ABJIVP03                WORKING-STORAGE DFHBMSCA
DFHCSADS   ABJIVP03                LINKAGE SECTION DFHCSADS
DFHTCADS   ABJIVP03                LINKAGE SECTION DFHTCADS
.....          E N D    O F    R E P O R T    .....

```

Call/Program report

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID - SAMPLE RUN          17 APR 1998 18:48:49   Page   1
..... C A L L  -- P R O G R A M   R E P O R T .....
PROGRAM   N OF CALL
NAME      CALLS  NAME

ABJIVP03 00006 "CBLTDLI"
AMPM2AA  00010 'CBLTDLI'
BLGA201  00005 'CBLBTS'
BLGF200  00001 'BLGT20A'
.....      E N D   O F   R E P O R T   .....

```

LCP directory

```

5648-B05 V2R1          - IBM COBOL CONVERSION AID -                      17 APR 1998 17:16:33   Page   1
..... L C P   D I R E C T O R Y .....
RESERVED WORD          PROCESSING DESCRIPTION          DATE          TIME          CORE          DBG
                                                                SIZE          OPT
-----
ACCEPT                flag ACCEPT used in CICS programs                20/APR/1998  07:56:10    185
ACCESS                Update Control file with FILE information          21/APR/1998  16:38:59    525
ACTUAL                ACTUAL KEY... replaced by RELATIVE KEY...        21/APR/1998  16:39:15    670
ADD                   ADD WITH BLL'S                                    20/APR/1998  07:56:44    8480
ALL                   MOVE ALL ...                                       20/APR/1998  07:57:17    815
ALPHABETIC            ALPHABETIC changed to ALPHABETIC-UPPER          21/APR/1998  16:39:29    295
ALTER                 SEGMENTATION - FLAG                              20/APR/1998  07:57:27    530
APPLY                 Remove APPLY clause from I-O-CONTROL para        21/APR/1998  16:41:51    1065
ASCENDING             Save KEY data-name for SEARCH...WHEN            21/APR/1998  16:51:15    690
ASSIGN                Change ASSIGN clause syntax                      21/APR/1998  16:42:20    9405
ASSIGN/DOS            Change ASSIGN clause syntax                      21/APR/1998  16:42:44    8680
BLANK                 Save 88's with VALUE zero for SET...TO TRUE      21/APR/1998  16:43:06    3940
BLOCK                 If VSAM file, remove BLOCK CONTAINS clause      21/APR/1998  16:43:21    245
CALL                  CALL statement update and flagging              21/APR/1998  16:43:30    5870
CANCEL                Flag identifiers with A and B only PICTURE      21/APR/1998  16:43:48    3270
CBL                   Update compiler options                          21/APR/1998  16:44:03    2595
CLOSE                 Remove WITH POSITIONING phrase                    21/APR/1998  16:44:16    1380
COM-REG              Flag reference to COM-REG special register       21/APR/1998  16:44:28    205
COMMUNICATION         COMMUNICATION SECTION FLAG                      20/APR/1998  08:00:06    655
COMPUTE              CICS - CHANGE BLL TO ADDRESS OF                  20/APR/1998  08:00:16    4965
CONFIGURATION         CHECK IF ENVIRONMENT DIVISION                    20/APR/1998  08:00:44    785
COPY                  COPY statement update and flagging              21/APR/1998  16:44:38    3240
CORR                  REPLACED BY SEPARATE MOVES                      20/APR/1998  08:01:28    5520
CORRESPONDING         REPLACED BY SEPARATE MOVES                      20/APR/1998  08:01:46    5520
COUNT                COMMUNICATION SECTION - FLAG                    20/APR/1998  08:02:05    345
CURRENCY              FLAG ANS 68 CURRENCY SIGN CLAUSE                20/APR/1998  08:02:41    560
CURRENT-DATE          CHANGE DATE FORMAT                              20/APR/1998  08:02:26    3400
DATE                  ADD - TO DATE COMPILED AND DATE WRITTEN         20/APR/1998  08:03:02    1555
DATE-COMPILED         COMMENT OUT DATE-COMPILED PARAGRAPH             20/APR/1998  08:02:52    300
DEBUG                 CHANGE TO COMMENT THE PACKET                    20/APR/1998  08:03:13    755
DEBUGGING             USE FOR DEBUGGING FLAG IF ~ PROCNAME           20/APR/1998  08:03:24    1230
DECLARATIVES          CHECK SECTION END (ALTER-PERFORM)               20/APR/1998  08:03:35    460
DELETE                flag DELETE used in CICS programs               20/APR/1998  08:03:45    185
DELIMITED             CHECK IF STRING INTO SAME AREA                  20/APR/1998  08:03:54    1565
DEPENDING             STORE ODO ON WORK FILE                          20/APR/1998  08:04:06    4690
DESCENDING            SAVE KEY ID FOR SEARCH ... WHEN                 20/APR/1998  08:04:24    690
DISABLE               COMMUNICATION SECTION FLAG                      20/APR/1998  08:04:34    645
DISP                  OPEN/CLOSE..DISP ..TAPE                        20/APR/1998  08:04:44    845
DISPLAY               flag DISPLAY used in CICS programs              20/APR/1998  08:04:55    185
DIVIDE                FLAG SIZE ERROR WHEN MULTIPLE RECEIVERS        20/APR/1998  08:05:04    1820
DIVISION              ENSURE PERIOD FOLLOWS DIVISION HDR             20/APR/1998  08:05:16    980
ENABLE               COMMUNICATION SECTION FLAG                      20/APR/1998  08:05:26    645
END-OF-CONVERSION-1  CHECK 00 ADD WRITE                              20/APR/1998  08:05:36    8805
END-OF-CONVERSION-2  ADD DATA ITEMS IN WS                           20/APR/1998  08:06:40    8960
END-OF-CONVERSION-2/DOS  ADD DATA ITEMS IN WS                           20/APR/1998  08:05:57    8820
END-OF-CONVERSION-3  ADD DATA ITEMS IN WS                           20/APR/1998  08:07:01    8765
END-OF-CONVERSION-3/DOS  ADD DATA ITEMS IN WS                           20/APR/1998  08:07:23    4115
END-OF-CONVERSION-4  ADD SPECIAL NAMES                               20/APR/1998  08:07:38    7920
END-OF-CONVERSION-5  LIST DATA NAMES                                20/APR/1998  08:07:59    3305

```

RESERVED WORD	PROCESSING DESCRIPTION	DATE	TIME	CORE SIZE	DBG OPT
ENTER	REMOVE ENTER STATEMENT	20/APR/1998	08:08:14	605	
ENVIRONMENT	CHECK IF CONFIGURATION-SECTION	20/APR/1998	08:08:24	1570	
EXAMINE	REPLACE EXAMINE WITH INSPECT	20/APR/1998	08:08:36	3765	
EXEC	REPLACE POINTER OPTION BY ADDRESS OF ...	20/APR/1998	08:08:52	2990	
EXHIBIT	CHANGE EXHIBIT TO DISPLAY	20/APR/1998	08:09:06	7895	
FD	CONVER FD ENTRY,CHECK LABEL CLAUSE	20/APR/1998	08:09:28	7750	
FILE-LIMIT	DELETE FILE-LIMIT ANS 68 CLAUSE	20/APR/1998	08:09:51	535	
FILE-LIMITS	DELETE FILE-LIMITS ANS 68 CLAUSE	20/APR/1998	08:10:06	535	
GENERATE	STATEMENT FLAGGED RPWT	20/APR/1998	08:10:17	365	
ID	SET FLAG WHEN ENTERING ID DIVISION	20/APR/1998	08:10:27	160	
IDENTIFICATION	SET FLAG WHEN ENTERING ID DIVISION	20/APR/1998	08:10:37	160	
IN	ISSUE MESSAGE FOR QUALIFIED INDEXES	20/APR/1998	08:10:46	2940	
INDEXED	STORE INDEX NAME ON WORK FILE	20/APR/1998	08:11:00	2550	
INITIALIZE	FLAG REPLACING ALPHABETIC/ALPHANUMERIC	20/APR/1998	08:11:14	3930	
INITIATE	STATEMENT FLAGGED RPWT	20/APR/1998	08:11:30	365	
INPUT-OUTPUT	SET FLAG WHEN ENTERING I-O SECTION	20/APR/1998	08:11:50	165	
INSPECT	FLAG IF COLLATING SEQUENCE HAS AN ALSO	20/APR/1998	08:11:40	540	
JUST	ANSI 68 - RIGHT JUSTIFY PICTURE VALUE	20/APR/1998	08:11:59	6225	
JUSTIFIED	ANSI 68 - RIGHT JUSTIFY PICTURE VALUE	20/APR/1998	08:12:18	6225	
LABEL	CHECK LABEL CLAUSE	20/APR/1998	08:12:38	3690	
LEAVE	OPEN...LEAVE TAPE	20/APR/1998	08:18:57	620	
LINE	REMOVE LINE AFTER MNEMONIC NAME	20/APR/1998	08:19:09	605	
LINES	REMOVE LINE AFTER MNEMONIC NAME	20/APR/1998	08:19:19	605	
LINKAGE	SET FLAG WHEN ENTERING IN LINKAGE SECTION	20/APR/1998	08:19:29	425	
MEMORY	REMOVE MEMORY SIZE CLAUSE	20/APR/1998	08:20:48	310	
MERGE	flag MERGE used in CICS programs	20/APR/1998	08:20:58	185	
MULTIPLE	DELETE MULTIPLE FILE TAPE CLAUSE	20/APR/1998	08:21:07	770	
MULTIPLY	FLAG SIZE ERROR WHEN MULTIPLE RECEIVERS	20/APR/1998	08:21:18	1800	
NATIVE	ADD ALPHABET WORD IN SPECIAL-NAMES	20/APR/1998	08:21:29	615	
NOMINAL	DELETE NOMINAL KEY CLAUSE	20/APR/1998	08:21:40	840	
NOT	FLAG ANSI 68 ABBREV. RELATION CONDITIONS	20/APR/1998	08:21:50	5620	
NOTE	COMMENT OUT NOTE STATEMENT	20/APR/1998	08:22:08	650	
NSTD-REELS	FLAG NSTD-REELS SPECIAL REGISTER	20/APR/1998	08:22:18	205	
OBJECT-COMPUTER	MOVE INTO AREA A.	20/APR/1998	08:22:28	320	
OCCURS	Correct order of phrases in OCCURS clause	21/APR/1998	16:45:21	1885	
OF	ISSUE MESSAGE FOR QUALIFIED INDEXES	20/APR/1998	08:22:52	2940	
ON	FLAG ON DEBUGGING	20/APR/1998	08:23:06	4545	
OPEN	REVERSED OPTION - MULTIREEL FILE	20/APR/1998	08:23:23	915	
OTHERWISE	REPLACE OTHERWISE BY ELSE	20/APR/1998	08:23:34	290	
PERFORM	FLAG PERFORM...VARYING...AFTER	20/APR/1998	08:23:44	815	
PIC	FLAG SCALED VARIABLES	20/APR/1998	08:23:54	7195	
PICTURE	FLAG SCALED VARIABLES	20/APR/1998	08:24:15	7195	
POSITIONING	CHANGE POSITIONING TO ADVANCING	20/APR/1998	08:24:35	9580	
PROCEDURE	GENERATE ERROR DECLARATIVES	20/APR/1998	08:25:01	4870	
PROCESS	MODIFY COMPILER OPTIONS FOR COBOL 370	20/APR/1998	08:25:17	2595	
PROCESSING	DELETE 68 STANDARD CLAUSE	20/APR/1998	08:25:31	255	
PROGRAM-ID	UPDATE PROGRAM FILE	20/APR/1998	08:25:40	2495	
READ	MOVE NOMINAL TO RECORD KEY	20/APR/1998	08:25:55	525	
RECEIVE	COMMUNICATION SECTION FLAG	20/APR/1998	08:26:22	640	

RESERVED WORD	PROCESSING DESCRIPTION	DATE	TIME	CORE SIZE	DBG OPT
RECORD	UPDATE KEY RECORD OF WORK FILE	20/APR/1998	08:26:32	240	
REDEFINES	REMOVE CLAUSE IN FD	20/APR/1998	08:26:42	425	
RELATIVE	KEEP RELATIVE KEY	20/APR/1998	08:26:52	480	
RELEASE	ADD LENGTH FOR VARIABLE LENGTH RECEIVER	20/APR/1998	08:27:02	4350	
REMARKS	COMMENT OUT REMARKS PARAGRAPH	20/APR/1998	08:27:19	340	
REPLACE	ANSI 85 FLAG OTHERWISE ADD SUFFIX	20/APR/1998	08:27:29	620	
REPORT	STATEMENT FLAGGED RPWT	20/APR/1998	08:27:39	730	
REPORTS	STATEMENT FLAGGED RPWT	20/APR/1998	08:27:50	635	
REREAD	OPEN...REREAD TAPE	20/APR/1998	08:28:00	605	
RERUN	CHANGE RERUN CLAUSE SYNTAX	20/APR/1998	08:28:11	2545	
RERUN/DOS	CHANGE ASSIGN NAME SYNTAX	20/APR/1998	08:28:23	4220	
RESERVE	CHANGE RESERVE SYNTAX ANS 68 TO ANS 74	20/APR/1998	08:28:39	1495	
REWRITE	MOVE NOMINAL KEY TO RECORD KEY	20/APR/1998	08:28:51	8215	
SAME	CHANGE SAME AREA TO SAME RECORD AREA	20/APR/1998	08:29:28	400	
SD	CONVER SD ENTRY , LABEL CLAUSE	20/APR/1998	08:29:42	3550	
SEARCH	SEARCH WHEN KEY	20/APR/1998	08:29:58	9060	
SECTION	SET FLAG WHEN ENTERING A SECTION	20/APR/1998	08:30:48	1165	
SEEK	DELETE STANDARD 68 CLAUSE	20/APR/1998	08:31:08	680	
SELECT	UPDATE CONTROL FILE	20/APR/1998	08:31:26	2875	
SEND	COMMUNICATION SECTION FLAG	20/APR/1998	08:31:42	640	
SEQUENCE	ADD ALPHABET WORD IN SPECIAL-NAMES	20/APR/1998	08:31:53	2260	
SERVICE	REPLACE SERVICE RELOAD BY CONTINUE	20/APR/1998	08:32:06	1065	
SORT-OPTION	REMOVE SORT-OPTION SPECIAL REGISTER	20/APR/1998	08:32:18	290	
SOURCE-COMPUTER	MOVE INTO AREA A	20/APR/1998	08:32:28	320	
SPECIAL-NAMES	ADD SPECIAL NAMES	20/APR/1998	08:32:38	680	
STANDARD-1	ADD ALPHABET WORD IN SPECIAL-NAMES	20/APR/1998	08:32:48	610	
START	MOVE NOMINAL TO RECORD KEY	20/APR/1998	08:32:59	5800	
STOP	flag STOP used in CICS programs	20/APR/1998	08:33:18	280	
STRING	FLAG ALPHANUMERIC-EDITED RECEIVERS	20/APR/1998	08:33:28	3410	
SUBTRACT	BLL SUBTRACT ...	20/APR/1998	08:33:43	8615	
TERMINATE	CHANGE TO A COMMENT RPWT	20/APR/1998	08:34:07	365	
THAN	REMOVE THAN IF > THAN OR < THAN	20/APR/1998	08:34:17	340	
THEN	DELETE THEN BETWEEN STATEMENTS	20/APR/1998	08:34:26	965	
TIME-OF-DAY	CHANGE TIME-OF-DAY FORMAT	20/APR/1998	08:34:49	2810	
TRACE	REMOVE TRACE STATEMENT	20/APR/1998	08:35:22	1295	
TRACK-AREA	TRACK-AREA REMOVED	20/APR/1998	08:35:03	250	
TRACK-LIMIT	TRACK-LIMIT REMOVED	20/APR/1998	08:35:13	210	
TRANSFORM	REPLACE TRANSFORM BY INSPECT	20/APR/1998	08:35:34	1310	
TRUE	SET...TO TRUE - REL. CONDITIONS FLAGGED	20/APR/1998	08:35:46	4620	
UNSTRING	UNSTRING DELIMITED BY ALL FLAG	20/APR/1998	08:36:03	8520	
USE	REMOVE USE FOR DEBUGGING/REPORTING SECTION	20/APR/1998	08:36:27	3510	
USING	START ...USING KEY	20/APR/1998	08:36:46	1555	
VALUE	REMOVE SIGN IF PICTURE UNSIGNED	20/APR/1998	08:37:01	2185	
VALUES	CHANGED TO VALUE	20/APR/1998	08:37:14	2640	
WHEN-COMPILED	CHANGE WHEN-COMPILED FORMAT	20/APR/1998	08:37:27	2610	
WORKING-STORAGE	SET FLAG ENTERING WS SECTION	20/APR/1998	08:38:04	365	
WRITE	MOVE NOMINAL KEY TO RECORD KEY	20/APR/1998	08:37:40	8890	
ZEROES	REPLACE ZEROES WITH ZERO IN IF	20/APR/1998	08:38:16	1030	
ZEROS	REPLACE ZEROS WITH ZERO IN IF	20/APR/1998	08:38:28	1030	

RESERVED WORD	PROCESSING DESCRIPTION	DATE	TIME	CORE SIZE	DBG OPT
01	MODULE STANDARD : LEVEL 01	20/APR/1998	08:19:39	5885	
1	MODULE STANDARD : LEVEL 1	20/APR/1998	08:19:57	5965	
77	MODULE STANDARD : LEVEL 77	20/APR/1998	08:20:15	3645	
848	Add suffix to user-defined words	21/APR/1998	16:45:11	710	
849	CHECK END-OF-PAGE AGAINST LINAGE	20/APR/1998	08:13:03	585	
850	MODIFY UPSI SWITCH	20/APR/1998	08:13:17	6655	
851	add LENGTH for variable length receiver	20/APR/1998	08:13:41	6815	
852	ADD SUFFIX TO DOS & OS user-defined word	20/APR/1998	08:14:02	470	
853	ADD ALPHABET WORD IN SPECIAL-NAMES	20/APR/1998	08:14:12	925	
854	SAVE MNEMONIC NAMES FOR ADVANCING .. LINE	20/APR/1998	08:14:23	280	
855	STATEMENT FLAGGED RPWT	20/APR/1998	08:14:33	290	
856	COMMENT OUT COMMENT PARAGRAPH	20/APR/1998	08:14:43	300	
857	ADD SUFFIX FOR RESERVED WORD (DOS)	20/APR/1998	08:14:53	610	
858	ADD SUFFIX to ANSI68/74 user-defined word	20/APR/1998	08:15:03	465	
859	ADD SUFFIX TO DOS, OS & VS COBOL II WORD	20/APR/1998	08:15:13	595	
860	CHECK PERIODS BEFORE/BEHIND LABELS.	20/APR/1998	08:15:23	1760	
861	ADD SUFFIX TO PROGRAM NAME	20/APR/1998	08:15:35	970	
862	ADD QUOTE/APOST MSG ON CONTINUED LITRL	20/APR/1998	08:15:46	625	
863	REMOVE BACK-TO-BACK PARENTHESES	20/APR/1998	08:15:56	610	
864	CHECK LITERAL HAS SPACES FOR AND AFT	20/APR/1998	08:16:07	1435	
865	REMOVE CONSECUTIVE PERIODS	20/APR/1998	08:16:19	110	
867	FLAG FILE-STATUS	20/APR/1998	08:16:28	1000	
870	Add DATE FORMAT clause	20/APR/1998	08:16:39	9900	
88	PUT VALUE BETWEEN QUOTE IF NEEDED	20/APR/1998	08:20:29	5855	
890	REMOVE BLL CELLS IN LINKAGE SECTION	20/APR/1998	08:17:05	2390	
891	CHANGE BLL CELLS TO ADDRESS OF ...	20/APR/1998	08:17:18	6855	
892	REMOVE STATEMENT WITH SECONDARY BLL	20/APR/1998	08:17:40	1300	
893	FLAG STATEMENT WITH REDEFINED BLL	20/APR/1998	08:17:51	465	
894	FLAG STATEMENT WHICH REFERENCES 1ST BLL	20/APR/1998	08:18:02	455	
895	FLAG 01 LEVEL RECORDS WITHOUT BLL CELLS	20/APR/1998	08:18:12	190	
896	FLAG BLL CELLS THAT DO NOT HAVE 01 RECORDS	20/APR/1998	08:18:21	190	
991	REMOVE BRACKETS AROUND OPERATORS	20/APR/1998	08:18:30	2335	
997	REMOVE TO AFTER =	20/APR/1998	08:18:43	3435	
.....	E N D O F D I R E C T O R Y				

Compilation of an LCP

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN
STMT SEQNBR A 1 B.. ... 2 LCP SOURCE STATEMENTS ... 6 7 .IDENTFCN

17 APR 1998 03:06:46 PAGE 1

```

/*****00001000
*
*          CONVERA OBJECT-COMPUTER          'MOVE INTO AREA A.'          *00002000
*
*          *00003001
*          *00004000
*****00005000
* Licensed Materials - Property of IBM          *00006000
*          *00007000
* 5785-CCC 5785-ABJ 5648-B05 5686-A07          *00008000
*          *00009000
* (c) Copyright IBM Corp. 1982, 1998. All Rights Reserved.          *00009100
*          *00009200
* US Government Users Restricted Rights - Use,          *00009300
* duplication or disclosure restricted by GSA ADP          *00009400
* Schedule Contract with IBM Corp.          *00009500
*          *00009600
*****00010200
00011000
2 OBJECT-010.          00012000
3 IF COBOL-TYPE NOT = 'DOS/VS'          00013000
4 AND COBOL-TYPE NOT = 'OS/VS'          00014000
5 GO TO END-CHANGE.          00015000
6 IF WHERE-USED IS NOT EQUAL TO 'EN'          00016000
7 GO TO END-CHANGE.          00017000
8 IF TOKEN-POSITION NOT < 5          00017100
9 MOVE 01 TO STARTING-POSITION          00017300
10 MOVE TOKEN-TEXT TO ADD-TEXT          00017400
11 PERFORM DETERMINE-LENGTH          00017500
12 PERFORM REPLACE.          00017600
13 GO TO END-CHANGE .          00029000
TEXT DESCRIPTION - MOVE INTO AREA A.
LCP PROGRAM NAME - OBJECT-COMPUTER
TABLE DRIVEN CORE SIZE - 320
```

```

/*****00001000
*
* 1  CONVER EXAMINE      'REPLACE EXAMINE WITH INSPECT'      *00002000
*
*  REPLACE THE EXAMINE STATEMENT WITH THE INSPECT STATEMENT *00003000
*
*  ----- SYNTAX DESCRIPTION ----- *00005000
*
*  *00005300
*  *00005600
*  *00006000
*  *00007000
*  FORMAT 1 COBOL ANS 68 :
*  ----- *00008000
*  *00009000
*  -- <EXAMINE> <IDENTIFIER-1> *00010000
*  -- <TALLYING> *00011000
*  -- <UNTIL FIRST> <LITERAL-1> *00012000
*  -- <ALL> <LITERAL-1> *00013000
*  -- <LEADING> <LITERAL-1> *00014000
*  -- *00015000
*  ++ <REPLACING> <BY> <LITERAL-2> *00016000
*  *00017000
*  *00018000
*  FORMAT 1 COBOL ANS 74 :
*  ----- *00019000
*  -- <MOVE> <ZEROS> <TO> <TALLY> *00020000
*  -- <INSPECT> <IDENTIFIER-1> *00021000
*  -- <TALLYING> <TALLY> <FOR> *00022000
*  -- <CHARACTERS > <BEFORE> <ALPHA-LITERAL-1> *00023000
*  -- <ALL> <ALPHA-LITERAL-1> *00024000
*  -- <LEADING> <ALPHA-LITERAL-1> *00025000
*  ++ <REPLACING> *00026000
*  ++ <CHARACTERS > <BY> *00027000
*  ++ <ALPHA-LITERAL-2> *00028000
*  ++ <BEFORE> <ALPHA-LITERAL-1> *00029000
*  ++ <ALL> <ALPHA-LITERAL-1> *00030000
*  ++ <BY> <ALPHA-LITERAL-2> *00031000
*  ++ <LEADING> <ALPHA-LITERAL-1> *00032000
*  ++ <BY> <ALPHA-LITERAL-2> *00033000
*  *00034000
*  *00035000
*  *00036000
*  FORMAT 2 COBOL ANS 68 :
*  ----- *00037000
*  -- <EXAMINE> <IDENTIFIER-1> *00038000
*  -- <REPLACING> *00039000
*  -- <UNTIL FIRST> <LITERAL-3> <BY> <LITERAL-4> *00040000
*  -- <ALL> <LITERAL-3> <BY> <LITERAL-4> *00041000
*  -- <LEADING> <LITERAL-3> <BY> <LITERAL-4> *00042000
*  -- <FIRST> <LITERAL-3> <BY> <LITERAL-4> *00043000
*  *00044000
*  *00045000
*  *00046000
*  FORMAT 2 COBOL ANS 74 :
*  ----- *00047000
*  -- <INSPECT> <IDENTIFIER-1> *00048000
*  -- <REPLACING> *00049000
*  -- <CHARACTERS > <BY> *00050000
*  -- <ALPHA-LITERAL-4> *00051000
*  -- <BEFORE> <ALPHA-LITERAL-3> *00052000
*  -- <ALL> <ALPHA-LITERAL-3> *00053000
*  -- <BY> <ALPHA-LITERAL-4> *00054000
*  -- <LEADING> <ALPHA-LITERAL-3> *00055000
*  -- <BY> <ALPHA-LITERAL-4> *00056000
*  -- <FIRST> ALPHA-LITERAL-3 *00056000

```

```

*      --                <BY> <ALPHA-LITERAL-4>          *00057000
*                                                                *00058000
*****00059000
* Licensed Materials - Property of IBM                    *00060000
*                                                                *00061000
* 5785-CCC 5785-ABJ 5648-B05 5686-A07                    *00062000
*                                                                *00063000
* (c) Copyright IBM Corp. 1982, 1998. All Rights Reserved. *00064000
*                                                                *00064100
* US Government Users Restricted Rights - Use,            *00064200
* duplication or disclosure restricted by GSA ADP          *00064300
* Schedule Contract with IBM Corp.                        *00064400
*                                                                *00064500
*****00064600
00065000
2  * 05 POSITION-SAVE          PIC 9(2) .                    00066000
3  * 05 HOLD-TOKEN          PIC X(30) .                     00067000
4  * 05 HOLD-LENGTH        PIC 9(3) .                       00068000
5  * 05 HOLD-LITERAL       PIC X(30) .                       00069000
6  * 05 LITERAL-LENGTH     PIC 9(3) .                       00070000
7  * 05 TOKEN-POINTER-SAVE PIC 9(7) .                       00071000
8  * 05 UNTIL-FLAG         PIC X(1) .                        00072000
                                                                00073000
9  SKIP-WS .                                                00074000
                                                                00075000
10 IF COBOL-TYPE NOT = 'DOS/VS'                             00076000
11 AND COBOL-TYPE NOT = 'OS/VS'                             00076500
12 GO TO END-CHANGE.                                        00077000
13 IF WHERE-USED IS NOT EQUAL TO 'PR'                       00078000
14 GO TO END-CHANGE.                                        00079000
15 MOVE 'N' TO UNTIL-FLAG.                                   00080000
16 MOVE TOKEN-POSITION TO POSITION-SAVE.                      00081000
17 MOVE TOKEN-POINTER TO TOKEN-POINTER-SAVE.               00082000
                                                                00083000
18 MOVE POSITION-SAVE TO STARTING-POSITION.                  00086000
                                                                00087000
19 PERFORM GET-NEXT-TOKEN.                                   00088000
20 PERFORM BYPASS-IDENTIFIER.                               00089000
21 IF TOKEN-TEXT IS EQUAL TO 'TALLYING'                     00090000
22 MOVE TOKEN-POINTER-SAVE TO TOKEN-POINTER                00091000
                                                                00091100
23 PERFORM GET-TOKEN                                        00092000
24 PERFORM REMOVE                                          00092103
25 MOVE '18MOVE ZERO TO TALLY' TO ADD-GROUP                00093000
26 PERFORM SUFFIX                                          00094000
27 PERFORM SPLIT-LINE                                     00094100
28 MOVE '07INSPECT' TO ADD-GROUP                           00094402
29 PERFORM SUFFIX                                          00094602
30 MOVE 'ABJ6018' TO MESSAGE-ID                            00095000
31 PERFORM EDIT-MESSAGE                                    00096000
32 MOVE 'A' TO INPUT-TEXT                                   00097000
33 MOVE TEXT-08 TO OUTPUT-TEXT                             00098000
34 MOVE 7 TO RECEIVING-CHARACTER                           00099000
35 MOVE 1 TO STARTING-CHARACTER                            00100000
36 MOVE 1 TO LENGTH-OF-MOVE                                00101000
37 PERFORM MOVE-LCP                                        00102000
38 MOVE OUTPUT-TEXT TO TEXT-08                             00103000
39 ELSE                                                    00104000
40 MOVE TOKEN-POINTER-SAVE TO TOKEN-POINTER                00105000
41 PERFORM GET-TOKEN                                        00106002
42 MOVE '07INSPECT' TO ADD-GROUP                           00108002
43 PERFORM REPLACE.                                        00109002
44 MOVE 'ABJ6019' TO MESSAGE-ID.                            00110000

```

45	PERFORM EDIT-MESSAGE.	00111000
		00112000
46	PERFORM GET-NEXT-TOKEN .	00113000
47	PERFORM BYPASS-IDENTIFIER.	00114000
		00115000
48	IF TOKEN-TEXT = 'TALLYING'	00116000
49	PERFORM TALLYING-010 THRU TALLYING-END	00117000
50	ELSE	00118000
51	PERFORM REPLACING-010 THRU REPLACING-END.	00119000
52	GO TO END-CHANGE.	00120000
		00121000
		00123000
*	CONVERSION OF FORMAT 1 .	00124000
*	-----	00125000
		00126000
53	TALLYING-010.	00127000
		00128000
54	MOVE '05TALLY' TO ADD-GROUP.	00129000
55	PERFORM INSERT-AFTER.	00130000
56	MOVE '03FOR' TO ADD-GROUP.	00131000
57	PERFORM INSERT-AFTER.	00132000
		00133000
		00134000
58	PERFORM GET-NEXT-TOKEN.	00135000
*	TOKEN-TEXT IS NOW : UNTIL OR ALL OR LEADING .	00136000
*	HOLD TOKEN-TEXT .	00137000
59	MOVE TOKEN-TEXT TO HOLD-TOKEN.	00138000
60	MOVE TOKEN-LENGTH TO HOLD-LENGTH.	00139000
		00140000
61	IF TOKEN-TEXT = 'UNTIL'	00141000
62	MOVE '10CHARACTERS' TO ADD-GROUP	00142000
63	PERFORM REPLACE	00143000
64	PERFORM GET-NEXT-TOKEN	00144000
65	MOVE '06BEFORE' TO ADD-GROUP	00145000
66	PERFORM REPLACE.	00146000
		00147000
		00148000
67	PERFORM GET-NEXT-TOKEN.	00149000
*	TOKEN-TEXT IS NOW LITERAL-1 .	00150000
*	TRANSFORM LITERAL-1 IN ALPHA-LITERAL-1 .	00151000
68	PERFORM BLD-LITERAL THRU BLD-LITERAL-END .	00152000
69	MOVE TOKEN-TEXT TO HOLD-LITERAL.	00153000
70	MOVE TOKEN-LENGTH TO LITERAL-LENGTH.	00154000
		00155000
		00156000
71	PERFORM GET-NEXT-TOKEN.	00157000
*	TOKEN-TEXT IS NOW ON THE REPLACING OPTION OF FORMAT 1 .	00158000
72	IF TOKEN-TEXT NOT = 'REPLACING'	00159000
73	GO TO TALLYING-END.	00160000
		00161000
74	IF HOLD-TOKEN IS EQUAL TO 'UNTIL'	00162000
75	MOVE '10CHARACTERS' TO ADD-GROUP	00163000
76	PERFORM INSERT-AFTER	00164000
77	PERFORM GET-NEXT-TOKEN 2 TIMES	00165000
78	PERFORM BLD-LITERAL THRU BLD-LITERAL-END	00166000
79	MOVE '06BEFORE' TO ADD-GROUP	00167000
80	PERFORM INSERT-AFTER	00168000

81	MOVE HOLD-LITERAL TO ADD-TEXT	00169000
82	MOVE LITERAL-LENGTH TO ADD-LENGTH	00170000
83	PERFORM INSERT-AFTER .	00171000
		00172000
84	IF HOLD-TOKEN NOT = 'UNTIL'	00173000
85	MOVE HOLD-TOKEN TO ADD-TEXT	00174000
86	MOVE HOLD-LENGTH TO ADD-LENGTH	00175000
87	PERFORM INSERT-AFTER	00176000
88	MOVE HOLD-LITERAL TO ADD-TEXT	00177000
89	MOVE LITERAL-LENGTH TO ADD-LENGTH	00178000
90	PERFORM INSERT-AFTER	00179000
91	PERFORM GET-NEXT-TOKEN 2 TIMES	00182000
92	PERFORM BLD-LITERAL THRU BLD-LITERAL-END.	00183000
		00184000
93	TALLYING-END.	00185000
94	EXIT.	00186000
		00187000
		00188000
	* CONVERSION OF FORMAT 2 :	00189000
	* -----	00190000
		00191000
95	REPLACING-010.	00192000
		00193000
96	PERFORM GET-NEXT-TOKEN.	00194000
	* TOKEN-TEXT IS NOW : UNTIL OR ALL OR LEADING .	00195000
97	IF TOKEN-TEXT NOT = 'UNTIL'	00196000
98	PERFORM GET-NEXT-TOKEN	00197000
99	PERFORM BLD-LITERAL THRU BLD-LITERAL-END	00198000
100	PERFORM GET-NEXT-TOKEN 2 TIMES	00199000
101	PERFORM BLD-LITERAL THRU BLD-LITERAL-END	00200000
102	GO TO REPLACING-END.	00201000
		00202000
		00203000
	* PROCESS THE UNTIL FIRST OPTION .	00204000
		00205000
	* REPLACE : UNTIL FIRST BY CHARACTERS BY	00206000
103	PERFORM REMOVE.	00207000
104	PERFORM REMOVE-NEXT-TOKEN.	00208000
105	MOVE '10CHARACTERS' TO ADD-GROUP.	00209000
106	PERFORM INSERT-AFTER.	00210000
		00211000
107	PERFORM GET-NEXT-TOKEN.	00212000
	* TOKEN-TEXT IS NOW LITERAL-3 .	00213000
	* TRANSFORM LITERAL-3 IN ALPHA-LITERAL-3 .	00214000
	* AND REMOVE LITERAL-3 .	00215000
108	MOVE 'Y' TO UNTIL-FLAG.	00216000
109	PERFORM BLD-LITERAL THRU BLD-LITERAL-END .	00217000
110	MOVE TOKEN-TEXT TO HOLD-LITERAL	00218000
111	MOVE TOKEN-LENGTH TO LITERAL-LENGTH.	00219000
112	PERFORM REMOVE.	00220000
113	MOVE 'N' TO UNTIL-FLAG.	00221000
		00222000
114	PERFORM GET-NEXT-TOKEN 2 TIMES .	00223000
	* MAINTAIN : BY LITERAL-4	00224000
115	PERFORM BLD-LITERAL THRU BLD-LITERAL-END .	00225000
		00226000
	* GENERATE : BEFORE LITERAL-3	00228000

116	MOVE '06BEFORE' TO ADD-GROUP.	00229000
117	PERFORM INSERT-AFTER.	00230000
118	MOVE HOLD-LITERAL TO ADD-TEXT.	00231000
119	MOVE LITERAL-LENGTH TO ADD-LENGTH.	00232000
120	PERFORM INSERT-AFTER.	00233000
		00234000
121	REPLACING-END.	00236000
122	EXIT.	00237000
		00238000
123	BLD-LITERAL .	00241000
124	IF TOKEN-TYPE-CODE IS EQUAL TO 'L'	00242000
125	OR TOKEN-TEXT IS EQUAL TO 'SPACE'	00243000
126	OR TOKEN-TEXT IS EQUAL TO 'SPACES'	00244000
127	OR TOKEN-TEXT IS EQUAL TO 'ZERO'	00245000
128	OR TOKEN-TEXT IS EQUAL TO 'ZEROS'	00246000
129	OR TOKEN-TEXT IS EQUAL TO 'ZEROS'	00247000
130	OR TOKEN-TEXT IS EQUAL TO 'LOW-VALUE'	00248000
131	OR TOKEN-TEXT IS EQUAL TO 'LOW-VALUES'	00249000
132	OR TOKEN-TEXT IS EQUAL TO 'HIGH-VALUE'	00250000
133	OR TOKEN-TEXT IS EQUAL TO 'HIGH-VALUES'	00251000
134	OR TOKEN-TEXT IS EQUAL TO 'QUOTE'	00252000
135	OR TOKEN-TEXT IS EQUAL TO 'QUOTES'	00253000
136	GO TO BLD-LITERAL-END .	00254000
137	MOVE SPACES TO STRING-TEXT .	00255000
138	MOVE SPACES TO STRING-DELIMITER .	00256000
139	PERFORM UNSTRING-LCP .	00257000
140	IF LITERAL-SEPARATOR IS EQUAL TO 'A'	00258000
141	MOVE '''' TO STRING-WORD-01	00259000
142	MOVE '''' TO STRING-WORD-03	00260000
143	ELSE	00261000
144	MOVE ''' TO STRING-WORD-01	00262000
145	MOVE ''' TO STRING-WORD-03.	00263000
146	MOVE TOKEN-TEXT TO STRING-WORD-02 .	00264000
147	PERFORM STRING-LCP .	00265000
148	MOVE STRING-TEXT TO TOKEN-TEXT .	00266000
149	MOVE STRING-LENGTH TO TOKEN-LENGTH .	00267000
150	MOVE TOKEN-TEXT TO ADD-TEXT .	00268000
151	MOVE TOKEN-LENGTH TO ADD-LENGTH .	00269000
152	IF UNTIL-FLAG IS EQUAL TO 'N'	00270000
153	PERFORM REPLACE .	00271000
154	BLD-LITERAL-END .	00272000
155	EXIT .	00273000

TEXT DESCRIPTION - REPLACE EXAMINE WITH INSPECT
 LCP PROGRAM NAME - EXAMINE
 TABLE DRIVEN CORE SIZE - 3765

COBOL conversion

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN ABJIVP01 15 APR 1998 15:59:39 PAGE 1
 LINEID SEQNBR-A 1 B.. ... 2 COBOL SOURCE STATEMENTS ... 6 7 .IDENTFCN MSGID SEV --- D I A G N O S T I C S ---

```

000001 IDENTIFICATION DIVISION. 00001000
000002 PROGRAM-ID. ABJIVP01. 00002000
000003 * PROGRAM CONVERTED BY
000004 * CCCA FOR VSE/ESA 5686-A07
000005 * CONVERSION DATE 04/20/98 17:34:42.
000006 * ----- *00003000
000007 * LICENSED MATERIALS - PROPERTY OF IBM *00004000
000008 * *00005000
000009 * 5785-CCC 5785-ABJ 5648-B05 5686-A07 *00006000
000010 * *00007000
000011 * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00008000
000012 * *00009000
000013 * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *00010000
000014 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP *00011000
000015 * SCHEDULE CONTRACT WITH IBM CORP. *00012000
000016 * *00013000
000017 * ----- *00014000
000018 *OLD** REMARKS. 00015000 ABJ6011 00 REMARKS CHANGED TO COMMENT
000019 *REMARKS. 00015000
000020 *OLD** THIS PROGRAM IS BEING WRITTEN TO TEST THE PROPER CONVERSION 00016000
000021 * THIS PROGRAM IS BEING WRITTEN TO TEST THE PROPER CONVERSION 00016000
000022 *OLD** FROM OS/VS COBOL SOURCE LANGUAGE TO IBM SOURCE LANGUAGE. 00017000
000023 * FROM OS/VS COBOL SOURCE LANGUAGE TO IBM SOURCE LANGUAGE. 00017000
000024 *OLD** AUTHOR. XXXXXX. 00018000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000025 *AUTHOR. XXXXXX. 00018000
000026 *OLD** DATE-WRITTEN. JANUARY 24, 1983. 00019000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000027 *DATE-WRITTEN. JANUARY 24, 1983. 00019000
000028 00020000
000029 *OLD** NOTE - THE FOLLOWING AREAS ARE ADDRESSED 00021000
000030 * NOTE - THE FOLLOWING AREAS ARE ADDRESSED 00021000
000031 *OLD** 1 REMARKS 00022000
000032 * 1 REMARKS 00022000
000033 *OLD** 2 THEN 00023000
000034 * 2 THEN 00023000
000035 *OLD** 3 OTHERWISE 00024000
000036 * 3 OTHERWISE 00024000
000037 *OLD** 4 CURRENT-DATE 00025000
000038 * 4 CURRENT-DATE 00025000
000039 *OLD** 5 TIME-OF-DAY 00026000
000040 * 5 TIME-OF-DAY 00026000
000041 *OLD** 6 NOTE 00027000
000042 * 6 NOTE 00027000
000043 *OLD** 7 EXAMINE...TALLYING...REPLACING 00028000
000044 * 7 EXAMINE...TALLYING...REPLACING 00028000
000045 *OLD** 8 JUSTIFIED. 00029000
000046 * 8 JUSTIFIED. 00029000
000047 00030000
000048 *OLD** DATE-COMPILED. TODAYS DATE. 00031000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000049 *DATE-COMPILED. TODAYS DATE. 00031000
000050 EJECT 00032000
000051 ENVIRONMENT DIVISION. 00033000
000052 INPUT-OUTPUT SECTION. 00034000
000053 FILE-CONTROL. 00035000

```

```

000054          SELECT PRINT-FILE                                00036000
000055          ASSIGN TO UT-3330-S-DDPRINT.                    00037000
000056          DATA DIVISION.                                  00038000
000057          FILE SECTION.                                    00039000
000058          FD PRINT-FILE                                    00040000
000059          *OLD** RECORDING MODE IS F                       00041000 ABJ6119 00 RECORDING MODE CLAUSE REMOVED
000060          *OLD** LABEL RECORDS ARE STANDARD                00042000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000061          *OLD** DATA RECORD IS OUT-LINE.                00043000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000062          .                                                00042000
000063          01 OUT-LINE PIC X(80).                              00044000
000064          WORKING-STORAGE SECTION.                          00045000 ABJ6004 00 LCP-TIME-OF-DAY-68 GENERATED
000065          01 LCP-TIME-OF-DAY-68 PIC 9(6).                    IN WORKING-STORAGE
000066          01 LCP-TIME-OF-DAY-74.                              ABJ6002 00 LCP-CURRENT-DATE-68 GENERATED
000067             05 LCP-TIME-74 PIC 9(6).                        IN WORKING-STORAGE
000068             05 FILLER PIC 9(2).
000069          01 LCP-CURRENT-DATE-68.
000070             05 LCP-MONTH PIC X(2).
000071             05 FILLER PIC X VALUE "/".
000072             05 LCP-DAY1 PIC X(2).
000073             05 FILLER PIC X VALUE "/".
000074             05 LCP-YEAR PIC X(2).
000075          01 LCP-DATE-NEW-74.
000076             05 LCP-YEAR PIC X(2).
000077             05 LCP-MONTH PIC X(2).
000078             05 LCP-DAY1 PIC X(2).
000079          77 MY-COUNTER PIC 9(5) VALUE 0.                    00046000
000080          77 TRIPSWCH PIC 9 VALUE 0.                          00047000
000081          77 PASSWCH PIC 9 VALUE 0.                          00048000
000082          77 FAILSWCH PIC 9 VALUE 1.                        00049000
000083          77 CURRFLAG PIC 9 VALUE 0.                        00050000
000084          77 TOFDFLAG PIC 9 VALUE 0.                        00051000
000085          77 I PIC 9 VALUE 0.                                00052000
000086          77 DATE1 PIC X(8) VALUE SPACES.                   00053000
000087          77 DATE2 PIC X(8) VALUE SPACES.                   00054000
000088          77 DATE3 PIC X(8) VALUE SPACES.                   00055000
000089          77 TIME1 PIC X(6) VALUE SPACES.                   00056000
000090          77 TIME2 PIC X(6) VALUE SPACES.                   00057000
000091          77 TIME3 PIC X(6) VALUE SPACES.                   00058000
000092          .                                                00059000
000093          01 ORIGINAL-NUMBER.                                00060000
000094             05 FILLER PIC 9(18) VALUE 0.                   00061000
000095             05 FILLER PIC 9(18) VALUE 0.                   00062000
000096             05 FILLER PIC 9(18) VALUE 000000009099843576. 00063000
000097             05 FILLER PIC 9(18) VALUE 1212121212121290. 00064000
000098          .                                                00065000
000099          01 THIS-DEF REDEFINES ORIGINAL-NUMBER.          00066000
000100             03 A-NUMBER OCCURS 2 TIMES.                    00067000
000101                 05 LINE1 PIC 9(18).                      00068000
000102                 05 LINE2 PIC 9(18).                      00069000
000103          .                                                00070000
000104          01 A-POEM.                                         00071000
000105             03 LINE1.                                       00072000
000106                 05 FILLER PIC X(20) VALUE "ROSES ARE RED VIOLET". 00073000
    
```

```

000107      05 FILLER      PIC X(20) VALUE "S ARE BLUE,      ". 00074000
000108      03 LINE2.      PIC X(20) VALUE "SUGAR IS SWEET AND S". 00075000
000109      05 FILLER      PIC X(20) VALUE "O ARE YOU.      ". 00076000
000110      05 FILLER      PIC X(20) VALUE "O ARE YOU.      ". 00077000
000111      05 FILLER      PIC X(20) VALUE "O ARE YOU.      ". 00078000
000112      05 FILLER      PIC X(20) VALUE "O ARE YOU.      ". 00079000
000113      01 FAIL1CON2.   PIC XX VALUE SPACES. 00080000
000114      03 FILLER      PIC X(20) VALUE SPACES. 00081000
000115      03 CPLACE     PIC X(20) VALUE SPACES. 00082000
000116      03 CPLACE     PIC X(20) VALUE SPACES. 00083000
000117      01 FAIL2CON.   PIC XX VALUE SPACES. 00084000
000118      03 FILLER      PIC X(20) VALUE "ALL THREE READINGS O". 00085000
000119      03 FILLER      PIC X(20) VALUE "F 'CURRENT-DATE' SHO". 00086000
000120      03 FILLER      PIC X(20) VALUE "ULD BE THE SAME, BUT". 00087000
000121      03 FILLER      PIC X(20) VALUE " THEY ARE:      ". 00088000
000122      03 FILLER      PIC X(20) VALUE " THEY ARE:      ". 00089000
000123      01 FAIL2CON2.   PIC XX VALUE SPACES. 00090000
000124      03 FILLER      PIC X(8) VALUE SPACES. 00091000
000125      03 DPLACE     PIC X(8) VALUE SPACES. 00092000
000126      03 DPLACE     PIC X(8) VALUE SPACES. 00093000
000127      01 FAIL3CON.   PIC XX VALUE SPACES. 00094000
000128      03 FILLER      PIC X(20) VALUE "THE THREE READINGS O". 00095000
000129      03 FILLER      PIC X(20) VALUE "F 'TIME-OF-DAY' SHOU". 00096000
000130      03 FILLER      PIC X(20) VALUE "LD BE EQUAL OR IN AS". 00097000
000131      03 FILLER      PIC X(20) VALUE "CENDING ORDER,      ". 00098000
000132      03 FILLER      PIC X(20) VALUE "CENDING ORDER,      ". 00099000
000133      01 FAIL3CON1.   PIC XX VALUE SPACES. 00100000
000134      03 FILLER      PIC X(20) VALUE "BUT THEY ARE:      ". 00101000
000135      03 FILLER      PIC X(20) VALUE "BUT THEY ARE:      ". 00102000
000136      01 FAIL3CON2.   PIC XX VALUE SPACES. 00103000
000137      03 FILLER      PIC X(6) VALUE SPACES. 00104000
000138      03 TPLACE     PIC X(6) VALUE SPACES. 00105000
000139      03 TPLACE     PIC X(6) VALUE SPACES. 00106000
000140      01 FAILCON.    PIC XX VALUE SPACES. 00107000
000141      03 FILLER      PIC X(20) VALUE "TEST CASE SAMPLE F". 00108000
000142      03 FILLER      PIC X(20) VALUE "AILED.      ". 00109000
000143      03 FILLER      PIC X(20) VALUE "AILED.      ". 00110000
000144      01 SUCCESS.    PIC XX VALUE SPACES. 00111000
000145      03 FILLER      PIC X(20) VALUE "TEST CASE SAMPLE I". 00112000
000146      03 FILLER      PIC X(20) VALUE "S SUCCESSFUL.      ". 00113000
000147      03 FILLER      PIC X(20) VALUE "S SUCCESSFUL.      ". 00114000
000148      EJECT          00115000
000149      PROCEDURE DIVISION. 00116000
000150      THIS-IS-A SECTION. 00117000
000151      START-HERE. 00118000
000152      *OLD** MOVE TIME-OF-DAY TO TIME1 00118000
000153      ACCEPT LCP-TIME-OF-DAY-74 FROM TIME 00118000
000154      MOVE LCP-TIME-74 TO LCP-TIME-OF-DAY-68
000155      MOVE LCP-TIME-OF-DAY-68 TO TIME1
000156      ABJ6005 00 NEW CODE GENERATED FOR
000157      TIME-OF-DAY
000158      OPEN OUTPUT PRINT-FILE 00119000
000159      *OLD** MOVE CURRENT-DATE TO DATE1 00120000
000160      ACCEPT LCP-DATE-NEW-74 FROM DATE 00120000
000161      MOVE CORRESPONDING LCP-DATE-NEW-74 TO LCP-CURRENT-DATE-68
  
```

```

000159          MOVE LCP-CURRENT-DATE-68 TO DATE1                      ABJ6003 00 NEW CODE GENERATED FOR
                                                CURRENT-DATE
000160 *OLD**    MOVE CURRENT-DATE TO DATE2                          00121000
000161          ACCEPT LCP-DATE-NEW-74 FROM DATE                      00121000
000162          MOVE CORRESPONDING LCP-DATE-NEW-74 TO LCP-CURRENT-DATE-68
000163          MOVE LCP-CURRENT-DATE-68 TO DATE2                      ABJ6003 00 NEW CODE GENERATED FOR
                                                CURRENT-DATE
000164 *OLD**    MOVE CURRENT-DATE TO DATE3.                          00122000
000165          ACCEPT LCP-DATE-NEW-74 FROM DATE                      00122000
000166          MOVE CORRESPONDING LCP-DATE-NEW-74 TO LCP-CURRENT-DATE-68
000167          MOVE LCP-CURRENT-DATE-68 TO DATE3.                      ABJ6003 00 NEW CODE GENERATED FOR
                                                CURRENT-DATE
000168
000169 *OLD**    MOVE TIME-OF-DAY TO TIME2.                            00123000
000170          ACCEPT LCP-TIME-OF-DAY-74 FROM TIME                    00124000
000171          MOVE LCP-TIME-74 TO LCP-TIME-OF-DAY-68                00124000
000172          MOVE LCP-TIME-OF-DAY-68 TO TIME2.                      ABJ6005 00 NEW CODE GENERATED FOR
                                                TIME-OF-DAY
000173          IF DATE1 EQUAL DATE2 AND EQUAL DATE3 THEN            00125000
000174          NEXT SENTENCE                                          00126000
000175 *OLD**    OTHERWISE                                          00127000 ABJ6021 00 OTHERWISE REPLACED BY ELSE
000176          ELSE                                                00127000
000177          MOVE FAILSWCH TO TRIPSWCH                             00128000
000178          MOVE DATE1 TO DPLACE                                  00129000
000179          WRITE OUT-LINE FROM FAIL2CON                          00130000
000180          WRITE OUT-LINE FROM FAIL2CON2                        00131000
000181          MOVE DATE2 TO DPLACE                                  00132000
000182          WRITE OUT-LINE FROM FAIL2CON2                        00133000
000183          MOVE DATE3 TO DPLACE                                  00134000
000184          WRITE OUT-LINE FROM FAIL2CON2.                       00135000
000185 *OLD**    MOVE TIME-OF-DAY TO TIME3.                            00136000
000186          ACCEPT LCP-TIME-OF-DAY-74 FROM TIME                    00136000
000187          MOVE LCP-TIME-74 TO LCP-TIME-OF-DAY-68
000188          MOVE LCP-TIME-OF-DAY-68 TO TIME3.                      ABJ6005 00 NEW CODE GENERATED FOR
                                                TIME-OF-DAY
000189          IF (TIME1 LESS THAN TIME2 OR EQUAL TIME2) AND        00137000
000190          (TIME2 LESS THAN TIME3 OR EQUAL TIME3) THEN            00138000
000191          NEXT SENTENCE                                          00139000
000192 *OLD**    OTHERWISE                                          00140000 ABJ6021 00 OTHERWISE REPLACED BY ELSE
000193          ELSE                                                00140000
000194          MOVE FAILSWCH TO TRIPSWCH                             00141000
000195          MOVE TIME1 TO TPLACE                                  00142000
000196          WRITE OUT-LINE FROM FAIL3CON                          00143000
000197          WRITE OUT-LINE FROM FAIL3CON1                        00144000
000198          WRITE OUT-LINE FROM FAIL3CON2                        00145000
000199          MOVE TIME2 TO TPLACE                                  00146000
000200          WRITE OUT-LINE FROM FAIL3CON2                        00147000
000201          MOVE TIME3 TO TPLACE                                  00148000
000202          WRITE OUT-LINE FROM FAIL3CON2.                       00149000
000203          AFTER-THOUGHT.                                        00150000
000204 *OLD**    EXAMINE A-POEM TALLYING ALL SPACES REPLACING BY "*" 00151000 ABJ6018 00 TALLY IS INITIALIZED
000205          MOVE ZERO TO TALLY                                    00151000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000206          INSPECT A-POEM TALLYING TALLY FOR ALL SPACES REPLACING ALL
  
```

```

000207                SPACES BY "*"
000208                MOVE TALLY TO MY-COUNTER                00152000
000209                MOVE LINE1 OF A-POEM TO OUT-LINE WRITE OUT-LINE 00153000
000210                MOVE LINE2 OF A-POEM TO OUT-LINE WRITE OUT-LINE 00154000
000211 *OLD**          EXAMINE A-POEM TALLYING ALL "*".          00155000 ABJ6018 00 TALLY IS INITIALIZED
000212                MOVE ZERO TO TALLY                        00155000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000213                INSPECT A-POEM TALLYING TALLY FOR ALL "*".
000214                IF TALLY = MY-COUNTER                    00156000
000215                    MOVE "OK" TO OUT-LINE WRITE OUT-LINE 00157000
000216 *OLD**          OTHERWISE                                00158000 ABJ6021 00 OTHERWISE REPLACED BY ELSE
000217                ELSE                                    00158000
000218                    MOVE "BAH" TO OUT-LINE WRITE OUT-LINE. 00159000
000219 *OLD**          EXAMINE A-POEM TALLYING ALL "E"            00160000 ABJ6018 00 TALLY IS INITIALIZED
000220                MOVE ZERO TO TALLY                        00160000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000221                INSPECT A-POEM TALLYING TALLY FOR ALL "E"
000222                PERFORM THREE-LINES                        00161000
000223 *OLD**          EXAMINE A-POEM TALLYING UNTIL FIRST "."    00162000 ABJ6018 00 TALLY IS INITIALIZED
000224                MOVE ZERO TO TALLY                        00162000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000225                INSPECT A-POEM TALLYING TALLY FOR CHARACTERS BEFORE "."
000226                PERFORM THREE-LINES                        00163000
000227 *OLD**          EXAMINE A-POEM TALLYING LEADING "R"        00164000 ABJ6018 00 TALLY IS INITIALIZED
000228                MOVE ZERO TO TALLY                        00164000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000229                INSPECT A-POEM TALLYING TALLY FOR LEADING "R"
000230                PERFORM THREE-LINES                        00165000
000231                MOVE 2 TO I                                00166000
000232 *OLD**          EXAMINE A-NUMBER(I) TALLYING ALL 1        00167000 ABJ6018 00 TALLY IS INITIALIZED
000233                MOVE ZERO TO TALLY                        00167000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000234                INSPECT A-NUMBER(I) TALLYING TALLY FOR ALL "1"
000235                PERFORM THREE-LINES                        00168000
000236 *OLD**          EXAMINE A-NUMBER(I) TALLYING LEADING 0 REPLACING BY 2. 00169000 ABJ6018 00 TALLY IS INITIALIZED
000237                MOVE ZERO TO TALLY                        00169000 ABJ6019 00 EXAMINE REPLACED BY INSPECT
000238                INSPECT A-NUMBER(I) TALLYING TALLY FOR LEADING "0" REPLACING
000239                    LEADING "0" BY "2".                    00170000
000240                THREE-LINES.                                00170000
000241                ADD TALLY TO MY-COUNTER.                    00171000
000242                MOVE TALLY TO OUT-LINE WRITE OUT-LINE      00172000
000243                MOVE MY-COUNTER TO OUT-LINE WRITE OUT-LINE. 00173000
000244                THE-END.                                    00174000
000245                IF TRIPSWCH EQUAL FAILSWCH OR MY-COUNTER NOT EQUAL 125 00175000
000246                    WRITE OUT-LINE FROM FAILCON            00176000
000247 *OLD**          OTHERWISE                                00177000 ABJ6021 00 OTHERWISE REPLACED BY ELSE
000248                ELSE                                    00177000
000249                    WRITE OUT-LINE FROM SUCCESS.            00178000
000250                CLOSE PRINT-FILE.                            00179000
000251                STOP RUN.                                    00180000 ABJ6126 99 *-----*
    
```

OPTIONS IN EFFECT :

Check procedure names YES Source language level DOS/VS COBOL LANGLVL(1)
 Flag Report Writer statements... YES CICS NO
 Remove obsolete elements YES Lines per report page60
 Negate implicit EXIT PROGRAM ... YES VSE system date format..... MM/DD/YY
 Generate END PROGRAM header NO Resequence source lines NO
 Compile after converting YES
 Flag manual changes (new source) NO Reserved word suffix 74
 Add DATE FORMAT clauses (MLE) NO Generate new program..... YES
 Remove VALUE clauses in FS & LS YES Generate new copy members YES
 FLAG:IF FILE-STATUS (NOT) = "00" YES Replace like-named copy members. NO
 Flag BLL cell arithmetic YES Print old source lines YES
 BLL cell conversion method..... A Print copy members YES
 Search source for literal delim. YES Print diagnostics of level >=... 00
 Literal delimiter (QUOTE/APOST). QUOTE Generate tokenization listing... NO
 OPTION-15 NO SQL NO

HIGHEST SEVERITY MESSAGE FOR THIS CONVERSION: 00

0033 MESSAGES ISSUED

0033 MESSAGES PRINTED

LINEID	MSGID	RC	MESSAGE TEXT
000019	ABJ6011	00	REMARKS CHANGED TO COMMENT
000025	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000027	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000049	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000062	ABJ6119	00	RECORDING MODE CLAUSE REMOVED
000062	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000062	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000064	ABJ6004	00	LCP-TIME-OF-DAY-68 GENERATED IN WORKING-STORAGE
000064	ABJ6002	00	LCP-CURRENT-DATE-68 GENERATED IN WORKING-STORAGE
000154	ABJ6005	00	NEW CODE GENERATED FOR TIME-OF-DAY
000159	ABJ6003	00	NEW CODE GENERATED FOR CURRENT-DATE
000163	ABJ6003	00	NEW CODE GENERATED FOR CURRENT-DATE
000167	ABJ6003	00	NEW CODE GENERATED FOR CURRENT-DATE
000172	ABJ6005	00	NEW CODE GENERATED FOR TIME-OF-DAY
000176	ABJ6021	00	OTHERWISE REPLACED BY ELSE
000188	ABJ6005	00	NEW CODE GENERATED FOR TIME-OF-DAY
000193	ABJ6021	00	OTHERWISE REPLACED BY ELSE
000205	ABJ6018	00	TALLY IS INITIALIZED
000205	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000212	ABJ6018	00	TALLY IS INITIALIZED
000212	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000217	ABJ6021	00	OTHERWISE REPLACED BY ELSE
000220	ABJ6018	00	TALLY IS INITIALIZED
000220	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000224	ABJ6018	00	TALLY IS INITIALIZED
000224	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000228	ABJ6018	00	TALLY IS INITIALIZED
000228	ABJ6019	00	EXAMINE REPLACED BY INSPECT

000233	ABJ6018	00	TALLY IS INITIALIZED
000233	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000237	ABJ6018	00	TALLY IS INITIALIZED
000237	ABJ6019	00	EXAMINE REPLACED BY INSPECT
000248	ABJ6021	00	OTHERWISE REPLACED BY ELSE

COBOL conversion with COPY

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN ABJIVP02 15 APR 1998 16:13:43 PAGE 1
 LINEID SEQNBR-A 1 B.. ... 2 COBOL SOURCE STATEMENTS ... 6 7 .IDENTFCN MSGID SEV --- D I A G N O S T I C S ---

```

000001 IDENTIFICATION DIVISION. 00001000
000002 PROGRAM-ID. ABJIVP02. 00002000
000003 * PROGRAM CONVERTED BY
000004 * CCCA FOR VSE/ESA 5686-A07
000005 * CONVERSION DATE 04/20/98 17:47:56.
000006 * ----- *00003000
000007 * LICENSED MATERIALS - PROPERTY OF IBM *00004000
000008 * *00005000
000009 * 5785-CCC 5785-ABJ 5648-B05 5686-A07 *00006000
000010 * *00007000
000011 * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00008000
000012 * *00009000
000013 * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *00010000
000014 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP *00011000
000015 * SCHEDULE CONTRACT WITH IBM CORP. *00012000
000016 * *00013000
000017 * ----- *00014000
000018 *OLD** REMARKS. 00015000 ABJ6011 00 REMARKS CHANGED TO COMMENT
000019 *REMARKS. 00015000
000020 *OLD** THIS PROGRAM COMPUTES THE GROSS SALARY, TAX AND NET SALARY 00016000
000021 * THIS PROGRAM COMPUTES THE GROSS SALARY, TAX AND NET SALARY 00016000
000022 *OLD** OF A GROUP OF EMPLOYEES. 00017000
000023 * OF A GROUP OF EMPLOYEES. 00017000
000024 *OLD** AUTHOR. YOUR NAME FOLLOWED BY A PERIOD. 00018000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000025 *AUTHOR. YOUR NAME FOLLOWED BY A PERIOD. 00018000
000026 *OLD** INSTALLATION. IBM-370. 00019000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000027 *INSTALLATION. IBM-370. 00019000
000028 *OLD** DATE-WRITTEN. FEB 27,1981. 00020000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000029 *DATE-WRITTEN. FEB 27,1981. 00020000
000030 * 00021000
000031 *OLD** NOTE - THE FOLLOWING AREAS ARE ADDRESSED 00022000
000032 * NOTE - THE FOLLOWING AREAS ARE ADDRESSED 00022000
000033 *OLD** 1 REMARKS 00023000
000034 * 1 REMARKS 00023000
000035 *OLD** 2 NOTE 00024000
000036 * 2 NOTE 00024000
000037 * 3 COPY FOR LANGLVL(1). 00025000
000038 * 00026000
000039 *OLD** DATE-COMPILED. TODAYS DATE. 00027000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000040 *DATE-COMPILED. TODAYS DATE. 00027000
000041 EJECT 00028000
000042 ENVIRONMENT DIVISION. 00029000
000043 CONFIGURATION SECTION. 00030000
000044 *SOURCE-COMPUTER. IBM-370. 00031000
000045 *OBJECT-COMPUTER. IBM-370. 00032000
000046 INPUT-OUTPUT SECTION. 00033000
000047 FILE-CONTROL. 00034000
000048 SELECT PRINT-OUT ASSIGN TO UR-2540R-S-PRINT. 00035000
000049 DATA DIVISION. 00036000
000050 * 00037000
000051 FILE SECTION. 00038000
000052 FD PRINT-OUT 00039000
000053 *OLD** LABEL RECORDS ARE OMITTED 00040000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED

```

```

000054 *OLD** DATA RECORDS ARE OUTPUT-RECORD ENTRY-DET. 00041000 ABJ6181 00 OBSOLETE ELEMENT IS REMOVED
000055 . 00040000
000056 *OLD** 01 OUTPUT-RECORD COPY ABJL901. 00042000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000057 01 OUTPUT-RECORD COPY ABJL901 REPLACING ==01 STD-LINE== BY 00042000
000058 == ==. 00042000
000059+ 01 STD-LINE PIC X(132).
000060 *OLD** 01 COPY ABJL902 REPLACING STEML BY PREML STHOURS BY PRHOURS 00043000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000061 COPY ABJL902 REPLACING STEML BY PREML STHOURS BY PRHOURS 00043000
000062 STSALARY BY PRSALARY STTAX BY PRTAX STNET BY PRNET. 00044000
000063+ 01 ENTRY-DET.
000064+ 03 FILLER PIC X(8).
000065+ 03 FILLER PIC X(3).
000066+ 03 STEML PIC X.
000067+ 03 FILLER PIC X(8).
000068+ 03 STHOURS PIC 99.
000069+ 03 FILLER PIC X(4).
000070+ 03 STSALARY PIC ZZZ.99.
000071+ 03 FILLER PIC X(2).
000072+ 03 STTAX PIC ZZZ.99.
000073+ 03 FILLER PIC X(4).
000074+ 03 STNET PIC ZZZ.99.
000075+ 03 FILLER PIC X(82).
000076 * 00045000
000077 WORKING-STORAGE SECTION. 00046000
000078 * 00047000
000079 *OLD** 77 NUM-OF-ITEMS COPY ABJL903. 00048000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000080 77 NUM-OF-ITEMS COPY ABJL903 REPLACING ==77 A== BY == ==. 00048000
000081+ 77 A PIC 99 VALUE 12.
000082 * 00049000
000083 *OLD** 77 COPY ABJL903A. 00050000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000084 COPY ABJL903A. 00050000
000085+ 77 WORK-GROSS PIC 9(3)V9(4).
000086 * 00051000
000087 77 WORK-TAX PIC 9(3)V9(4). 00052000
000088 77 WORK-NET PIC 9(3)V9(4). 00053000
000089 77 SUB1 PIC 99 VALUE 1. 00054000
000090 77 ERROR-FLAG PIC 9 VALUE 0. 00055000
000091 01 INPUT-AREA. 00056000
000092 03 ENTRYA. 00057000
000093 06 FILLER PIC X VALUE "A". 00058000
000094 06 FILLER PIC 99 VALUE 40. 00059000
000095 03 ENTRYB. 00060000
000096 06 FILLER PIC X VALUE "B". 00061000
000097 06 FILLER PIC 99 VALUE 41. 00062000
000098 03 ENTRYC. 00063000
000099 06 FILLER PIC X VALUE "C". 00064000
000100 06 FILLER PIC 99 VALUE 39. 00065000
000101 03 ENTRYD. 00066000
000102 06 FILLER PIC X VALUE "D". 00067000
000103 06 FILLER PIC 99 VALUE 16. 00068000
000104 03 ENTRYE. 00069000
000105 06 FILLER PIC X VALUE "E". 00070000
000106 06 FILLER PIC 99 VALUE 21. 00071000

```



```

000107      03  ENTRYF.                                00072000
000108      06  FILLER PIC X      VALUE "F".          00073000
000109      06  FILLER PIC 99     VALUE 44.          00074000
000110      03  ENTRYG.                                00075000
000111      06  FILLER PIC X      VALUE "G".          00076000
000112      06  FILLER PIC 99     VALUE 55.          00077000
000113      03  ENTRYH.                                00078000
000114      06  FILLER PIC X      VALUE "H".          00079000
000115      06  FILLER PIC 99     VALUE 60.          00080000
000116      03  ENTRYI.                                00081000
000117      06  FILLER PIC X      VALUE "I".          00082000
000118      06  FILLER PIC 99     VALUE 41.          00083000
000119      03  ENTRYJ.                                00084000
000120      06  FILLER PIC X      VALUE "J".          00085000
000121      06  FILLER PIC 99     VALUE 42.          00086000
000122      03  ENTRYK.                                00087000
000123      06  FILLER PIC X      VALUE "K".          00088000
000124      06  FILLER PIC 99     VALUE 39.          00089000
000125      03  ENTRYL.                                00090000
000126      06  FILLER PIC X      VALUE "L".          00091000
000127      06  FILLER PIC 99     VALUE 32.          00092000
000128      *
000129      *OLD** 01  COPY ABJL904 REPLACING A BY REDEF-AREA B BY INPUT-AREA. 00094000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000130      COPY ABJL904 REPLACING A BY REDEF-AREA B BY INPUT-AREA. 00094000
000131+      01  A REDEFINES B.
000132+      03  ENTRY-ITEM OCCURS 12 TIMES.
000133+      06  EMPLOYEE PIC X.
000134+      06  HOURS-WORK PIC 99.
000135      *
000136      01  HDG-1.                                00095000
000137      03  FILLER PIC X(8) VALUE SPACES.          00096000
000138      03  FILLER PIC X(21) VALUE "_____". 00097000
000139      03  FILLER PIC X(21) VALUE "_____". 00098000
000140      03  FILLER PIC X(82) VALUE SPACES.          00099000
000141      01  HDG-2.                                00100000
000142      03  FILLER PIC X(8) VALUE SPACES.          00101000
000143      03  FILLER PIC X(10) VALUE SPACES.          00102000
000144      03  FILLER PIC X(16) VALUE "HOURS GROSS ". 00103000
000145      03  FILLER PIC X(13) VALUE "TAX NET".      00104000
000146      03  FILLER PIC X(85) VALUE SPACES.          00105000
000147      01  HDG-3.                                00106000
000148      03  FILLER PIC X(8) VALUE SPACES.          00107000
000149      03  FILLER PIC X(10) VALUE "EMPLOYEE ".    00108000
000150      03  FILLER PIC X(8) VALUE "WORKED ".      00109000
000151      03  FILLER PIC X(8) VALUE "SALARY ".      00110000
000152      03  FILLER PIC X(10) VALUE "DEDUCTED ".  00111000
000153      03  FILLER PIC X(6) VALUE "SALARY".      00112000
000154      03  FILLER PIC X(82) VALUE SPACES.          00113000
000155      01  HDG-4.                                00114000
000156      03  FILLER PIC X(8) VALUE SPACES.          00115000
000157      03  FILLER PIC X(10) VALUE "_____".    00116000
000158      03  FILLER PIC X(8) VALUE "_____".      00117000
000159      03  FILLER PIC X(8) VALUE "_____".      00118000
000159      03  FILLER PIC X(8) VALUE "_____".      00119000
    
```

```

000160      03 FILLER      PIC X(10) VALUE "_____".          00120000
000161      03 FILLER      PIC X(6)  VALUE "_____".          00121000
000162      03 FILLER      PIC X(82) VALUE SPACES.            00122000
000163      PROCEDURE DIVISION.                                00123000
000164      OPEN OUTPUT PRINT-OUT.                             00124000
000165      WRITE OUTPUT-RECORD FROM HDG-1.                    00125000
000166      WRITE OUTPUT-RECORD FROM HDG-2.                    00126000
000167      WRITE OUTPUT-RECORD FROM HDG-3.                    00127000
000168      WRITE OUTPUT-RECORD FROM HDG-4.                    00128000
000169      PERFORM PROCESS THRU PROCESS2 VARYING SUB1 FROM 1 BY 1 00129000
000170      UNTIL SUB1 GREATER THAN NUM-OF-ITEMS.              00130000
000171      WRITE OUTPUT-RECORD FROM HDG-4.                    00131000
000172      GO TO EOJ-ROUTINE.                                  00132000
000173      PROCESS.                                           00133000
000174      MOVE SPACES TO ENTRY-DET.                            00134000
000175      MOVE EMPLOYEE(SUB1) TO PREMPL.                       00135000
000176      MOVE HOURS-WORK(SUB1) TO PRHOURS.                   00136000
000177      COMPUTE WORK-GROSS ROUNDED = HOURS-WORK(SUB1) * 4.00. 00137000
000178      MOVE WORK-GROSS TO PRSALARY.                          00138000
000179      IF WORK-GROSS GREATER THAN 150.00                   00139000
000180      COMPUTE WORK-TAX ROUNDED = (WORK-GROSS - 150) * .2 + 5 00140000
000181      GO TO PROCESS2.                                       00141000
000182      IF WORK-GROSS NOT LESS THAN 100.00                 00142000
000183      COMPUTE WORK-TAX = (WORK-GROSS - 100) * .1          00143000
000184      GO TO PROCESS2.                                       00144000
000185      MOVE ZEROS TO WORK-TAX.                              00145000
000186      PROCESS2.                                           00146000
000187      MOVE WORK-TAX TO PRTAX                                00147000
000188      COMPUTE WORK-NET = WORK-GROSS - WORK-TAX           00148000
000189      MOVE WORK-NET TO PRNET                                00149000
000190      WRITE ENTRY-DET.                                     00150000
000191      EOJ-ROUTINE.                                        00151000
000192      IF ERROR-FLAG = ZERO                                00152000
000193      MOVE "TEST CASE LCPTST09 IS SUCCESSFUL." TO OUTPUT-RECORD 00153000
000194      WRITE OUTPUT-RECORD                                  00154000
000195      *OLD** OTHERWISE                                     00155000 ABJ6021 00 OTHERWISE REPLACED BY ELSE
000196      ELSE                                                 00155000
000197      MOVE "TEST CASE LCPTST09 FAILED." TO OUTPUT-RECORD 00156000
000198      WRITE OUTPUT-RECORD.                                  00157000
000199      CLOSE PRINT-OUT.                                     00158000
000200      STOP RUN.                                           00159000 ABJ6126 99 *-----*
                                                                * END OF COBOL CONVERSION *
                                                                * 5648-B05 COBOL CONVERSION *
                                                                *-----*

```

OPTIONS IN EFFECT :

Check procedure names YES Source language level DOS/VS COBOL LANGLVL(1)
 Flag Report Writer statements... YES CICS NO
 Remove obsolete elements YES Lines per report page60
 Negate implicit EXIT PROGRAM ... YES VSE system date format..... MM/DD/YY
 Generate END PROGRAM header ... NO Resequence source lines NO
 Compile after converting YES
 Flag manual changes (new source) NO Reserved word suffix 74
 Add DATE FORMAT clauses (MLE) NO Generate new program..... YES
 Remove VALUE clauses in FS & LS YES Generate new copy members YES
 FLAG:IF FILE-STATUS (NOT) = "00" YES Replace like-named copy members. NO
 Flag BLL cell arithmetic YES Print old source lines YES
 BLL cell conversion method..... A Print copy members YES
 Search source for literal delim. YES Print diagnostics of level >=... 00
 Literal delimiter (QUOTE/APOST). QUOTE Generate tokenization listing... NO
 OPTION-15 NO SQL NO

HIGHEST SEVERITY MESSAGE FOR THIS CONVERSION: 00

0013 MESSAGES ISSUED

0013 MESSAGES PRINTED

LINEID	MSGID	RC	MESSAGE TEXT
000019	ABJ6011	00	REMARKS CHANGED TO COMMENT
000025	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000027	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000029	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000040	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000055	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000055	ABJ6181	00	OBSOLETE ELEMENT IS REMOVED
000057	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000061	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000080	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000084	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000130	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000196	ABJ6021	00	OTHERWISE REPLACED BY ELSE

COBOL conversion with CICS commands

5648-B05 V2R1 - IBM COBOL CONVERSION AID - SAMPLE RUN ABJIVP03 27 APR 1998 18:43:36 PAGE 1
 LINEID SEQNBR-A 1 B.. ... 2 COBOL SOURCE STATEMENTS ... 6 7 .IDENTFCN MSGID SEV --- D I A G N O S T I C S ---

```

000001 CBL QUOTE                                00001000
000002 ID DIVISION.                            00002000
000003 PROGRAM-ID. ABJIVP03.                    00003000
000004 * PROGRAM CONVERTED BY
000005 * CCCA FOR VSE/ESA 5686-A07
000006 * CONVERSION DATE 04/20/98 18:00:49.
000007 * -----*00004000
000008 * LICENSED MATERIALS - PROPERTY OF IBM      *00005000
000009 *                                           *00006000
000010 * 5785-CCC 5785-ABJ 5648-B05 5686-A07      *00007000
000011 *                                           *00008000
000012 * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00009000
000013 *                                           *00010000
000014 * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *00011000
000015 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP *00012000
000016 * SCHEDULE CONTRACT WITH IBM CORP.          *00013000
000017 *                                           *00014000
000018 * -----*00015000
000019 ENVIRONMENT DIVISION.                     00016000
000020 DATA DIVISION.                           00017000
000021 WORKING-STORAGE SECTION.                  00018000 ABJ6212 00 WORKING POINTER FOR CICS
000022 77 LCP-WS-ADDR-COMP PIC S9(8) COMP.      REDEFINES LCP-WS-ADDR-COMP      ADDED TO WORKING STORAGE
000023 77 LCP-WS-ADDR-PNTR REDEFINES LCP-WS-ADDR-COMP
000024 USAGE POINTER.
000025 77 PCB PIC X(4) VALUE "PCB ".              00019000
000026 77 GN PIC X(4) VALUE "GN ".               00020000
000027 77 GU PIC X(4) VALUE "GU ".              00021000
000028 77 GNP PIC X(4) VALUE "GNP ".            00022000
000029 77 TERM PIC X(4) VALUE "TERM".          00023000
000030 77 SAVE-TCAFCRC PIC X VALUE SPACE.        00024000
000031 77 SAVE-TCADLTR PIC X VALUE SPACE.      00025000
000032 77 SAVE-STATUS-CODE PIC XX VALUE SPACES. 00026000
000033 01 SAVE-TCACCCA PIC X(32) VALUE SPACES. 00027000
000034 01 PAGE-OVERFLOW-CTR PIC S9(4) COMP.    00028000
000035 01 DFHBMSCA.                            00029000
000036 02 DFHBMPM PICTURE X VALUE IS " ".     00030000
000037 02 DFHBMPNL PICTURE X VALUE IS " ".     00031000
000038 02 DFHBMASK PICTURE X VALUE IS "0".     00032000
000039 02 DFHBMUNP PICTURE X VALUE IS " ".     00033000
000040 02 DFHBMUNN PICTURE X VALUE IS "&".     00034000
000041 02 DFHBMPRO PICTURE X VALUE IS "-".     00035000
000042 02 DFHBMBRY PICTURE X VALUE IS "H".     00036000
000043 02 DFHBMDAR PICTURE X VALUE IS "<".     00037000
000044 02 DFHBMFSE PICTURE X VALUE IS "A".     00038000
000045 02 DFHBMPRF PICTURE X VALUE IS "/".     00039000
000046 02 DFHBMASF PICTURE X VALUE IS "1".     00040000
000047 02 DFHBMASB PICTURE X VALUE IS "8".     00041000
000048 02 DFHBMEOF PICTURE X VALUE IS "I".     00042000
000049 02 DFHBMDDET PICTURE X VALUE IS "J".     00043000
000050 02 DFHSA PICTURE X VALUE IS " ".        00044000
000051 02 DFHCOLOR PICTURE X VALUE IS "0".     00045000
000052 02 DFHPS PICTURE X VALUE IS "6".       00046000
000053 02 DFHHLT PICTURE X VALUE IS "a".       00047000

```

000054	02	DFH3270	PICTURE X	VALUE IS	"{".	00048000
000055	02	DFHVAL	PICTURE X	VALUE IS	"A".	00049000
000056	02	DFHALL	PICTURE X	VALUE IS	" "	00050000
000057	02	DFHERROR	PICTURE X	VALUE IS	" "	00051000
000058	02	DFHDFT	PICTURE X	VALUE IS	"y".	00052000
000059	02	DFHDFCOL	PICTURE X	VALUE IS	"y".	00053000
000060	02	DFHBLUE	PICTURE X	VALUE IS	"1".	00054000
000061	02	DFHRED	PICTURE X	VALUE IS	"2".	00055000
000062	02	DFHPINK	PICTURE X	VALUE IS	"3".	00056000
000063	02	DFHGREEN	PICTURE X	VALUE IS	"4".	00057000
000064	02	DFHTURQ	PICTURE X	VALUE IS	"5".	00058000
000065	02	DFHYELLO	PICTURE X	VALUE IS	"6".	00059000
000066	02	DFHNEUTR	PICTURE X	VALUE IS	"7".	00060000
000067	02	DFHBASE	PICTURE X	VALUE IS	"y".	00061000
000068	02	DFHDFHI	PICTURE X	VALUE IS	"y".	00062000
000069	02	DFHBLINK	PICTURE X	VALUE IS	"1".	00063000
000070	02	DFHREVR	PICTURE X	VALUE IS	"2".	00064000
000071	02	DFHUNDLN	PICTURE X	VALUE IS	"4".	00065000
000072	02	DFHMFIL	PICTURE X	VALUE IS	" "	00066000
000073	02	DFHMENT	PICTURE X	VALUE IS	" "	00067000
000074	02	DFHMF	PICTURE X	VALUE IS	" "	00068000
000075	02	DFHUNNOD	PICTURE X	VALUE IS	"(".	00069000
000076	02	DFHUNIMD	PICTURE X	VALUE IS	"I".	00070000
000077	02	DFHUNNUM	PICTURE X	VALUE IS	"J".	00071000
000078	02	DFHUNINT	PICTURE X	VALUE IS	"R".	00072000
000079	02	DFHUNNON	PICTURE X	VALUE IS	"J".	00073000
000080	02	DFHPROTI	PICTURE X	VALUE IS	"Y".	00074000
000081	02	DFHPROTN	PICTURE X	VALUE IS	"%".	00075000
000082	02	DFHMT	PICTURE X	VALUE IS	" "	00076000
000083	02	DFHMF	PICTURE X	VALUE IS	" "	00077000
000084	02	DFHMET	PICTURE X	VALUE IS	" "	00078000
000085	02	DFHMF	PICTURE X	VALUE IS	" "	00079000
000086						00080000
000087	01	DFHAID.				00081000
000088	02	DFHNULL	PIC X	VALUE IS	" "	00082000
000089	02	DFHENTER	PIC X	VALUE IS	QUOTE.	00083000
000090	02	DFHCLEAR	PIC X	VALUE IS	" "	00084000
000091	02	DFHCLRP	PIC X	VALUE IS	"J".	00085000
000092	02	DFHPEN	PIC X	VALUE IS	"=".	00086000
000093	02	DFHOPID	PIC X	VALUE IS	"W".	00087000
000094	02	DFHMSRE	PIC X	VALUE IS	"X".	00088000
000095	02	DFHSTRF	PIC X	VALUE IS	"h".	00089000
000096	02	DFHTRIG	PIC X	VALUE IS	" "" "	00090000
000097	02	DFHPA1	PIC X	VALUE IS	"%".	00091000
000098	02	DFHPA2	PIC X	VALUE IS	">".	00092000
000099	02	DFHPA3	PIC X	VALUE IS	" , "	00093000
000100	02	DFHPP1	PIC X	VALUE IS	"1".	00094000
000101	02	DFHPP2	PIC X	VALUE IS	"2".	00095000
000102	02	DFHPP3	PIC X	VALUE IS	"3".	00096000
000103	02	DFHPP4	PIC X	VALUE IS	"4".	00097000
000104	02	DFHPP5	PIC X	VALUE IS	"5".	00098000
000105	02	DFHPP6	PIC X	VALUE IS	"6".	00099000
000106	02	DFHPP7	PIC X	VALUE IS	"7".	00100000

```

000107      02 DFHPF8   PIC X VALUE IS "8".           00101000
000108      02 DFHPF9   PIC X VALUE IS "9".           00102000
000109      02 DFHPF10  PIC X VALUE IS ":".           00103000
000110      02 DFHPF11  PIC X VALUE IS "#".           00104000
000111      02 DFHPF12  PIC X VALUE IS "@".           00105000
000112      02 DFHPF13  PIC X VALUE IS "A".           00106000
000113      02 DFHPF14  PIC X VALUE IS "B".           00107000
000114      02 DFHPF15  PIC X VALUE IS "C".           00108000
000115      02 DFHPF16  PIC X VALUE IS "D".           00109000
000116      02 DFHPF17  PIC X VALUE IS "E".           00110000
000117      02 DFHPF18  PIC X VALUE IS "F".           00111000
000118      02 DFHPF19  PIC X VALUE IS "G".           00112000
000119      02 DFHPF20  PIC X VALUE IS "H".           00113000
000120      02 DFHPF21  PIC X VALUE IS "I".           00114000
000121      02 DFHPF22  PIC X VALUE IS ">".           00115000
000122      02 DFHPF23  PIC X VALUE IS ">.".          00116000
000123      02 DFHPF24  PIC X VALUE IS "<.".          00117000
000124      02 DFHPF24  PIC X VALUE IS "<.".          00118000
000125      02 DFHPF24  PIC X VALUE IS "<.".          00119000
000126      01 PSBNAME PIC X(8).                       00120000
000127      01 DLIO PIC X(70).                          00121000
000128      01 SSA1.                                    00122000
000129          02 FILLER PIC X(19) VALUE "ID (NUM =".  00123000
000130          02 SSAIKEY PIC X(5).                    00124000
000131          02 FILLER PIC X VALUE ")".              00125000
000132      01 SSA2.                                    00126000
000133          02 FILLER PIC X(19) VALUE "CHEQUE (COMPTE =". 00127000
000134          02 SSA2KEY PIC X(5).                    00128000
000135          02 FILLER PIC X VALUE ")".              00129000
000136      01 SSA3.                                    00130000
000137          02 FILLER PIC X(19) VALUE "PRET (PRENUM =". 00131000
000138          02 SSA3KEY PIC X(6).                    00132000
000139          02 FILLER PIC X VALUE ")".              00133000
000140 *OLD** 01 MAP11 COPY ABJCQIN.                       00134000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000141      01 MAP11. COPY ABJCQIN REPLACING ==01 MAP11.== BY ==. 00134000
000142+ * ----- *00001000
000143+ * LICENSED MATERIALS - PROPERTY OF IBM *00002000
000144+ * *00003000
000145+ * 5785-CCC 5785-ABJ 5648-B05 5686-A07 *00004000
000146+ * *00005000
000147+ * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00006000
000148+ * *00007000
000149+ * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *00008000
000150+ * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP *00009000
000151+ * SCHEDULE CONTRACT WITH IBM CORP. *00010000
000152+ * *00011000
000153+ * ----- *00012000
000154+ 01 MAP11.                                          00013000
000155+     02 FILLER PIC X(12).                          00014000
000156+     02 TITEL COMP PIC S9(4).                       00015000
000157+     02 TITELF PICTURE X.                            00016000
000158+     02 FILLER REDEFINES TITELF.                    00017000
000159+     03 TITELA PICTURE X.                            00018000

```

```

000160+      02 TITLEI PIC X(35).                00019000
000161+      02 CUSTNOL COMP PIC S9(4).        00020000
000162+      02 CUSTNOF PICTURE X.            00021000
000163+      02 FILLER REDEFINES CUSTNOF.    00022000
000164+      03 CUSTNOA PICTURE X.            00023000
000165+      02 CUSTNOI PIC X(5).            00024000
000166+      02 CHECKNOL COMP PIC S9(4).    00025000
000167+      02 CHECKNOF PICTURE X.          00026000
000168+      02 FILLER REDEFINES CHECKNOF.   00027000
000169+      03 CHECKNOA PICTURE X.          00028000
000170+      02 CHECKNOI PIC X(5).            00029000
000171+      02 LOANNOL COMP PIC S9(4).    00030000
000172+      02 LOANNOF PICTURE X.          00031000
000173+      02 FILLER REDEFINES LOANNOF.    00032000
000174+      03 LOANNOA PICTURE X.          00033000
000175+      02 LOANNOI PIC X(6).            00034000
000176+      01 MAP10 REDEFINES MAP1I.        00035000
000177+      02 FILLER PIC X(12).            00036000
000178+      02 FILLER PICTURE X(3).        00037000
000179+      02 TITLEO PIC X(35).            00038000
000180+      02 FILLER PICTURE X(3).        00039000
000181+      02 CUSTNOO PIC X(5).            00040000
000182+      02 FILLER PICTURE X(3).        00041000
000183+      02 CHECKNOO PIC X(5).          00042000
000184+      02 FILLER PICTURE X(3).        00043000
000185+      02 LOANNOO PIC X(6).            00044000
000186+      01 MAP2I.                        00045000
000187+      02 FILLER PIC X(12).            00046000
000188+      02 ERRNAMEL COMP PIC S9(4).    00047000
000189+      02 ERRNAMEF PICTURE X.          00048000
000190+      02 FILLER REDEFINES ERRNAMEF.   00049000
000191+      03 ERRNAMEA PICTURE X.          00050000
000192+      02 ERRNAMEI PIC X(8).            00051000
000193+      02 ERRNOL COMP PIC S9(4).    00052000
000194+      02 ERRNOF PICTURE X.            00053000
000195+      02 FILLER REDEFINES ERRNOF.    00054000
000196+      03 ERRNOA PICTURE X.            00055000
000197+      02 ERRNOI PIC X(6).            00056000
000198+      01 MAP20 REDEFINES MAP2I.        00057000
000199+      02 FILLER PIC X(12).            00058000
000200+      02 FILLER PICTURE X(3).        00059000
000201+      02 ERRNAMEO PIC X(8).            00060000
000202+      02 FILLER PICTURE X(3).        00061000
000203+      02 ERRNOO PIC X(6).            00062000
000204      LINKAGE SECTION.                  00135000
000205 *OLD** 01 DFHBLDLS SYNCHRONIZED.        00136000 ABJ6203 00 BLL'S ARE REMOVED
000206 *01 DFHBLDLS SYNCHRONIZED.            00136000
000207 *OLD** 02 BLLCBAR PICTURE XXXX.        00137000
000208 * 02 BLLCBAR PICTURE XXXX.            00137000
000209 *OLD** 02 CSACBAR PICTURE XXXX.        00138000
000210 * 02 CSACBAR PICTURE XXXX.            00138000
000211 *OLD** 02 CSAOPBAR PICTURE S9(8) USAGE IS COMPUTATIONAL. 00139000
000212 * 02 CSAOPBAR PICTURE S9(8) USAGE IS COMPUTATIONAL. 00139000

```

000213	*OLD**	02	TCACBAR PICTURE S9(8) USAGE IS COMPUTATIONAL.	00140000
000214	*	02	TCACBAR PICTURE S9(8) USAGE IS COMPUTATIONAL.	00140000
000215	*OLD**	02	PCB-LIST-PTR PIC S9(8) COMP.	00141000
000216	*	02	PCB-LIST-PTR PIC S9(8) COMP.	00141000
000217	*OLD**	02	PCB1-PTR PIC S9(8) COMP.	00142000
000218	*	02	PCB1-PTR PIC S9(8) COMP.	00142000
000219	*OLD**	02	CINQOUT-PTR PIC S9(8) COMP.	00143000
000220	*	02	CINQOUT-PTR PIC S9(8) COMP.	00143000
000221	*OLD**	02	ERRORMP-PTR PIC S9(8) COMP.	00144000
000222	*	02	ERRORMP-PTR PIC S9(8) COMP.	00144000
000223	*OLD**	02	CIDLOUT-PTR PIC S9(8) COMP.	00145000
000224	*	02	CIDLOUT-PTR PIC S9(8) COMP.	00145000
000225		01	DFHCSADS SYNCHRONIZED.	00146000
000226		02	CSAFILLER PICTURE X(512).	00147000
000227		02	FILLER1 REDEFINES CSAFILLER.	00148000
000228		03	FILLER.	00149000
000229		04	FILLER PICTURE X(76).	00150000
000230		04	CSACDTA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00151000
000231		04	CSATODP PICTURE S9(7) USAGE IS COMPUTATIONAL-3.	00152000
000232		04	FILLER PICTURE X(12).	00153000
000233		04	CSACTODB PICTURE S9(8) USAGE IS COMPUTATIONAL.	00154000
000234		04	FILLER PICTURE X(24).	00155000
000235		04	CSAJYDP PICTURE 9(7) USAGE IS COMPUTATIONAL-3.	00156000
000236		04	FILLER PICTURE X(64).	00157000
000237		03	FILLER.	00158000
000238		04	FILLER PICTURE X(8).	00159000
000239		04	CSAOPFLA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00160000
000240		04	FILLER PICTURE X(20).	00161000
000241		04	FILLER.	00162000
000242		05	CSAKCNAC PICTURE XXXX.	00163000
000243		05	CSASCNAC PICTURE XXXX.	00164000
000244		05	CSAPCNAC PICTURE XXXX.	00165000
000245		05	CSAICNAC PICTURE XXXX.	00166000
000246		05	CSADCNAC PICTURE XXXX.	00167000
000247		05	CSATCNAC PICTURE XXXX.	00168000
000248		05	CSAFCNAC PICTURE XXXX.	00169000
000249		05	CSATDNAC PICTURE XXXX.	00170000
000250		05	CSATSNAC PICTURE XXXX.	00171000
000251		05	CSASANAC PICTURE XXXX.	00172000
000252		05	CSATRNAC PICTURE XXXX.	00173000
000253		05	CSAPINAC PICTURE XXXX.	00174000
000254		05	FILLER PICTURE X(4).	00175000
000255		05	CSASPNAC PICTURE XXXX.	00176000
000256		05	CSATCRWE PICTURE XXXX.	00177000
000257		03	FILLER PICTURE X(215).	00178000
000258		03	CSAUTA1 PICTURE S9(5) USAGE IS COMPUTATIONAL-3.	00179000
000259		03	CSAUTA2 PICTURE S9(5) USAGE IS COMPUTATIONAL-3.	00180000
000260		03	CSAUTA3 PICTURE S9(5) USAGE IS COMPUTATIONAL-3.	00181000
000261		03	CSAUTA4 PICTURE S9(5) USAGE IS COMPUTATIONAL-3.	00182000
000262		03	FILLER PICTURE X(1).	00183000
000263	*		ABOVE FILLER ADDED BY APAR PN26174	00184000
000264				00185000
000265				00186000

000266	01 DFHTCADS PICTURE X(64) SYNCHRONIZED.	00187000
000267	01 CSAOPFL REDEFINES DFHTCADS SYNCHRONIZED.	00188000
000268	02 CSAATP PICTURE XXXX.	00189000
000269	02 CSAATTCH PICTURE XXXX.	00190000
000270	02 CSADLI PICTURE XXXX.	00191000
000271	02 CSABFNAC PICTURE XXXX.	00192000
000272	02 CSABMS PICTURE XXXX.	00193000
000273	02 CSATMSVT PICTURE XXXX.	00194000
000274	02 CSAJCN1 PICTURE XXXX.	00195000
000275	02 CSAJCN2 PICTURE XXXX.	00196000
000276	02 CSASRNAC PICTURE XXXX.	00197000
000277	02 CSASRTBA PICTURE XXXX.	00198000
000278	02 CSAKPNAC PICTURE XXXX.	00199000
000279	02 CSAATMSP PICTURE XXXX.	00200000
000280	02 CSAHLTBA PICTURE XXXX.	00201000
000281	02 CSAJCTBA PICTURE XXXX.	00202000
000282	01 DFHTCA SYNCHRONIZED.	00203000
000283	02 FILLER.	00204000
000284	03 FILLER PICTURE X(8).	00205000
000285	03 TCAFCAAA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00206000
000286	03 FILLER REDEFINES TCAFCAAA.	00207000
000287	04 TCAFCAA1 PICTURE X.	00208000
000288	04 FILLER PICTURE X(3).	00209000
000289	03 FILLER.	00210000
000290	04 FILLER PICTURE X(8).	00211000
000291	04 TCATCEA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00212000
000292	04 FILLER REDEFINES TCATCEA.	00213000
000293	05 TCATCQA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00214000
000294	04 TCATCTR1 PICTURE 9(4) USAGE IS COMPUTATIONAL.	00215000
000295	04 FILLER REDEFINES TCATCTR1.	00216000
000296	05 TCATCEI PICTURE X.	00217000
000297	05 TCATCTR PICTURE X.	00218000
000298	04 FILLER REDEFINES TCATCTR1.	00219000
000299	05 TCATCDC PICTURE X.	00220000
000300	05 FILLER PICTURE X.	00221000
000301	04 TCATCDP PICTURE X.	00222000
000302	04 FILLER PICTURE X(5).	00223000
000303	04 TCATCRS PICTURE X(60).	00224000
000304	04 FILLER REDEFINES TCATCRS.	00225000
000305	05 TCATCDP1 PICTURE 9(4) USAGE IS COMPUTATIONAL.	00226000
000306	05 FILLER PICTURE X(58).	00227000
000307	03 FILLER.	00228000
000308	04 TCASCCA.	00229000
000309	05 TCASCSA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00230000
000310	04 FILLER REDEFINES TCASCCA.	00231000
000311	05 TCAFCTL PICTURE S9(8) USAGE IS COMPUTATIONAL.	00232000
000312	04 FILLER REDEFINES TCASCCA.	00233000
000313	05 TCASCTR PICTURE X.	00234000
000314	05 TCASCIB PICTURE X.	00235000
000315	05 TCASCNB PICTURE 9(4) USAGE IS COMPUTATIONAL.	00236000
000316	04 FILLER REDEFINES TCASCCA.	00237000
000317	05 TCASCRI PICTURE 9(4) USAGE IS COMPUTATIONAL.	00238000
000318	05 FILLER PICTURE X(2).	00239000

000319	04 TCAFCTL1 PICTURE S9(8) USAGE IS COMPUTATIONAL.	00240000
000320	04 FILLER PICTURE X(28).	00241000
000321	03 FILLER.	00242000
000322	04 TCACCCA.	00243000
000323	05 TCACCCA1 PICTURE X(32).	00244000
000324	05 TCACCRS1 PICTURE X(56).	00245000
000325	05 TCACCSV1 PICTURE S9(4) USAGE IS COMPUTATIONAL.	00246000
000326	05 TCACCRSV PICTURE XX.	00247000
000327	05 TCACCSV2 PICTURE XXXX.	00248000
000328	04 FILLER REDEFINES TCACCCA.	00249000
000329	05 TCATPAPR PICTURE X.	00250000
000330	88 TCATPVAL VALUE "6".	00251000
000331	88 TCATPNVL VALUE "7".	00252000
000332	88 TCATPLNR VALUE " ".	00253000
000333	05 FILLER PICTURE X.	00254000
000334	05 TCATPOS PICTURE S9(4) USAGE IS COMPUTATIONAL.	00255000
000335	05 TCATPCS PICTURE S9(4) USAGE IS COMPUTATIONAL.	00256000
000336	05 TCATPOC PICTURE S9(4) USAGE IS COMPUTATIONAL.	00257000
000337	05 TCATPLDM PICTURE XX.	00258000
000338	05 TCATPCON PIC S9(4) USAGE IS COMPUTATIONAL.	00259000
000339	05 TCATPPNM PICTURE X(8).	00260000
000340	05 FILLER PICTURE X(76).	00261000
000341	04 FILLER REDEFINES TCACCCA.	00262000
000342	05 FILLER PICTURE X(24).	00263000
000343	05 TCAKCTI PICTURE X(4).	00264000
000344	05 TCAKCF A PICTURE S9(8) USAGE IS COMPUTATIONAL.	00265000
000345	05 FILLER PICTURE X(64).	00266000
000346	04 FILLER REDEFINES TCACCCA.	00267000
000347	05 TCAICDA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00268000
000348	05 FILLER REDEFINES TCAICDA.	00269000
000349	06 TCAICTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00270000
000350	06 FILLER PICTURE X(2).	00271000
000351	05 FILLER REDEFINES TCAICDA.	00272000
000352	06 TCAICRC PICTURE X.	00273000
000353	06 FILLER PICTURE X(3).	00274000
000354	05 TCAICQID.	00275000
000355	07 TCAICQPX PICTURE XX.	00276000
000356	07 FILLER PICTURE X(6).	00277000
000357	05 TCAICRT PICTURE S9(7) USAGE IS COMPUTATIONAL-3.	00278000
000358	05 TCAICTI PICTURE X(4).	00279000
000359	05 TCAICTID PICTURE X(4).	00280000
000360	05 FILLER PICTURE X(4).	00281000
000361	05 TCAFCTR1 PICTURE 9(4) USAGE IS COMPUTATIONAL.	00282000
000362	05 FILLER PICTURE X(66).	00283000
000363	04 FILLER REDEFINES TCACCCA.	00284000
000364	05 TCAPCLA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00285000
000365	05 FILLER REDEFINES TCAPCLA.	00286000
000366	06 TCAPCTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00287000
000367	06 FILLER PICTURE X(2).	00288000
000368	05 FILLER REDEFINES TCAPCLA.	00289000
000369	06 TCAPCRC PICTURE X.	00290000
000370	88 PCPGMIDER VALUE " ".	00291000
000371	88 PCNORESP VALUE " ".	00292000

000372	88	ICNORESP	VALUE " ".	00293000
000373	88	ICENDDATA	VALUE " ".	00294000
000374	88	ICIOERROR	VALUE " ".	00295000
000375	88	ICTRNIDER	VALUE " ".	00296000
000376	88	ICTRMIDER	VALUE " ".	00297000
000377	88	ICTSINVLD	VALUE " ".	00298000
000378	88	ICEXPIRD	VALUE " ".	00299000
000379	88	ICNOTFND	VALUE "¶".	00300000
000380	88	ICINVREQ	VALUE "ÿ".	00301000
000381	88	TSNORESP	VALUE " ".	00302000
000382	88	TSENERORR	VALUE " ".	00303000
000383	88	TSIDERROR	VALUE " ".	00304000
000384	88	TSIOERROR	VALUE " ".	00305000
000385	88	TSINVREQ	VALUE " ".	00306000
000386	88	TDNORESP	VALUE " ".	00307000
000387	88	TDQUEZERO	VALUE " ".	00308000
000388	88	TDIDERROR	VALUE " ".	00309000
000389	88	TDIOERROR	VALUE " ".	00310000
000390	88	TDNOTOPEN	VALUE " ".	00311000
000391	88	TDNOSPACE	VALUE " ".	00312000
000392	88	FCNORESP	VALUE " ".	00313000
000393	88	FCDSIDER	VALUE " ".	00314000
000394	88	FCSEGIDER	VALUE " ".	00315000
000395	88	FCINVREQ	VALUE " ".	00316000
000396	88	FCDUPDS	VALUE " ".	00317000
000397	88	FCNOTOPEN	VALUE " ".	00318000
000398	88	FCENDFILE	VALUE " ".	00319000
000399	88	FCIOERROR	VALUE "ø".	00320000
000400	88	FCNOTFND	VALUE "a".	00321000
000401	88	FCDUPREC	VALUE "b".	00322000
000402	88	FCNOSPACE	VALUE "c".	00323000
000403	88	FCDUPKEY	VALUE "d".	00324000
000404	88	FCILLOGIC	VALUE " ".	00325000
000405	06	TCAPCFLA	PICTURE X.	00326000
000406	06	TCAPCARO	PICTURE X.	00327000
000407	06	FILLER	PICTURE X.	00328000
000408	05	TCAPCPI	PICTURE X(8).	00329000
000409	05	FILLER	REDEFINES TCAPCPI.	00330000
000410	06	TCAPCERA	PICTURE S9(8) USAGE IS COMPUTATIONAL.	00331000
000411	06	FILLER	PICTURE X(4).	00332000
000412	05	TCAPCAC	PICTURE XXXX.	00333000
000413	05	TCAPCPSW	PICTURE X(8).	00334000
000414	05	TCAPCINT	PICTURE X(8).	00335000
000415	05	FILLER	PICTURE X(64).	00336000
000416	04	FILLER	REDEFINES TCACCCA.	00337000
000417	05	TCADCTR	PICTURE 9(4) USAGE IS COMPUTATIONAL.	00338000
000418	05	TCADCNB	PICTURE 9(4) USAGE IS COMPUTATIONAL.	00339000
000419	05	TCADCSA	PICTURE S9(8) USAGE IS COMPUTATIONAL.	00340000
000420	05	FILLER	PICTURE XXXX.	00341000
000421	05	TCADCDC	PICTURE XXXX.	00342000
000422	05	FILLER	PICTURE X(80).	00343000
000423	04	FILLER	REDEFINES TCACCCA.	00344000
000424	05	TCAFCAA	PICTURE S9(8) USAGE IS COMPUTATIONAL.	00345000

000425	05	FILLER REDEFINES TCAFCAA.	00346000
000426	06	TCAFCTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00347000
000427	06	FILLER PICTURE X(2).	00348000
000428	05	FILLER REDEFINES TCAFCAA.	00349000
000429	06	TCAFRCR PICTURE X.	00350000
000430	06	FILLER PICTURE X(3).	00351000
000431	05	TCAFCDI PICTURE X(8).	00352000
000432	05	TCAFURL PICTURE 9(4) USAGE IS COMPUTATIONAL.	00353000
000433	05	FILLER REDEFINES TCAFURL.	00354000
000434	06	TCAFNRD PICTURE 9(4) USAGE IS COMPUTATIONAL.	00355000
000435	05	FILLER PICTURE X(6).	00356000
000436	05	FILLER PICTURE X(8).	00357000
000437	05	TCAFRI PICTURE S9(8) USAGE IS COMPUTATIONAL.	00358000
000438	05	FILLER PICTURE X(64).	00359000
000439	04	FILLER REDEFINES TCACCCA.	00360000
000440	05	TCATDAA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00361000
000441	05	FILLER REDEFINES TCATDAA.	00362000
000442	06	TCATDRC PICTURE X.	00363000
000443	06	FILLER PICTURE X(3).	00364000
000444	05	FILLER REDEFINES TCATDAA.	00365000
000445	06	TCATDTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00366000
000446	06	FILLER PICTURE X(2).	00367000
000447	05	TCATDDI PICTURE XXXX.	00368000
000448	05	FILLER PICTURE X(88).	00369000
000449	04	FILLER REDEFINES TCACCCA.	00370000
000450	05	TCATSDA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00371000
000451	05	FILLER REDEFINES TCATSDA.	00372000
000452	06	TCATSRC PICTURE X.	00373000
000453	06	FILLER PICTURE X(3).	00374000
000454	05	FILLER REDEFINES TCATSDA.	00375000
000455	06	TCATSTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00376000
000456	06	FILLER PICTURE X(2).	00377000
000457	05	TCATSDI PICTURE X(8).	00378000
000458	05	TCATSRN PICTURE 9(4) USAGE IS COMPUTATIONAL.	00379000
000459	05	FILLER PICTURE X(2).	00380000
000460	05	TCATSTR2 PICTURE 9(4) USAGE IS COMPUTATIONAL.	00381000
000461	05	FILLER PICTURE X(78).	00382000
000462	04	FILLER REDEFINES TCACCCA.	00383000
000463	05	TCAMSTR1 PICTURE X(8).	00384000
000464	05	FILLER REDEFINES TCAMSTR1.	00385000
000465	06	FILLER PICTURE X.	00386000
000466	06	TCAMSTR2 PICTURE X.	00387000
000467	06	TCAMSTR3 PICTURE X.	00388000
000468	06	TCAMSTR4 PICTURE X.	00389000
000469	06	TCAMSTR5 PICTURE X.	00390000
000470	06	TCAMSTR6 PICTURE X.	00391000
000471	06	TCAMSTR7 PICTURE X.	00392000
000472	06	TCAMSTR8 PICTURE X.	00393000
000473	05	FILLER REDEFINES TCAMSTR1.	00394000
000474	06	TCAMSRC1 PICTURE X.	00395000
000475	06	TCAMSRC2 PICTURE X.	00396000
000476	06	TCAMSRC3 PICTURE X.	00397000
000477	06	TCAMSRI1 PICTURE X.	00398000

000478	06	TCAMSPGN PICTURE 9(4) USAGE IS COMPUTATIONAL.	00399000
000479	06	TCAMSOCN PICTURE 9(4) USAGE IS COMPUTATIONAL.	00400000
000480	05	FILLER REDEFINES TCAMSTR1.	00401000
000481	06	FILLER PICTURE XX.	00402000
000482	06	TCAMSRC3H PICTURE 9(4) USAGE IS COMPUTATIONAL.	00403000
000483	06	FILLER PICTURE X(4).	00404000
000484	05	FILLER REDEFINES TCAMSTR1.	00405000
000485	06	TCAMSRC PICTURE XXX.	00406000
000486	06	FILLER PICTURE X(5).	00407000
000487	05	TCAMSIOA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00408000
000488	05	FILLER REDEFINES TCAMSIOA.	00409000
000489	06	TCAMSTA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00410000
000490	05	TCAMSFSC PICTURE XXXX.	00411000
000491	05	FILLER REDEFINES TCAMSFSC.	00412000
000492	06	TCABMSFB PICTURE 9(4) USAGE IS COMPUTATIONAL.	00413000
000493	06	FILLER REDEFINES TCABMSFB.	00414000
000494	07	TCABMSWC PICTURE X.	00415000
000495	07	FILLER PICTURE X.	00416000
000496	06	FILLER REDEFINES TCABMSFB.	00417000
000497	07	TCAMSWCC PICTURE X.	00418000
000498	07	TCAMSJ PICTURE X.	00419000
000499	06	TCABMSCP PICTURE S9(4) USAGE IS COMPUTATIONAL.	00420000
000500	05	TCABMSMN PICTURE X(8).	00421000
000501	05	FILLER REDEFINES TCABMSMN.	00422000
000502	06	TCABMSMA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00423000
000503	06	FILLER PICTURE X(4).	00424000
000504	05	FILLER REDEFINES TCABMSMN.	00425000
000505	06	TCAMSHDR PICTURE S9(8) USAGE IS COMPUTATIONAL.	00426000
000506	06	FILLER PICTURE X(4).	00427000
000507	05	FILLER REDEFINES TCABMSMN.	00428000
000508	06	TCAMSRLA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00429000
000509	06	TCAMSRTI PICTURE S9(7) USAGE IS COMPUTATIONAL-3.	00430000
000510	06	FILLER REDEFINES TCAMSRTI.	00431000
000511	07	TCAMSTRL PICTURE S9(8) USAGE IS COMPUTATIONAL.	00432000
000512	05	TCAMSMN PICTURE X(8).	00433000
000513	05	FILLER REDEFINES TCAMSMN.	00434000
000514	06	TCAMMSA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00435000
000515	06	FILLER PICTURE X(4).	00436000
000516	05	FILLER REDEFINES TCAMSMN.	00437000
000517	06	TCAMSTI PICTURE X(4).	00438000
000518	06	FILLER PICTURE X.	00439000
000519	06	TCAMSOC PICTURE XXX.	00440000
000520	05	TCAMSLDM PICTURE XX.	00441000
000521	05	TCAMSLDC PICTURE X.	00442000
000522	05	TCAMSRID PICTURE XX.	00443000
000523	05	FILLER PICTURE XXX.	00444000
000524	05	TCAMSFMP PICTURE X(8).	00445000
000525	05	FILLER PICTURE X(48).	00446000
000526	04	FILLER REDEFINES TCACCCA.	00447000
000527	05	TCASPTR PICTURE 9(4) USAGE IS COMPUTATIONAL.	00448000
000528	05	FILLER PICTURE X(94).	00449000
000529	04	FILLER REDEFINES TCACCCA.	00450000
000530	05	TCADLIO PICTURE S9(8) USAGE IS COMPUTATIONAL.	00451000

000531	05	FILLER REDEFINES TCADLIO.	00452000
000532	06	FILLER PICTURE X.	00453000
000533	06	TCADLTR PICTURE X.	00454000
000534	88	FCDLINA VALUE "y".	00455000
000535	88	FCPSBSCH VALUE " ".	00456000
000536	88	FCPSBNF VALUE " ".	00457000
000537	88	FCTASKNA VALUE " ".	00458000
000538	88	FCPSBNA VALUE " ".	00459000
000539	88	FCLANGCON VALUE " ".	00460000
000540	88	FCPSBFAIL VALUE " ".	00461000
000541	88	FCFUNCNS VALUE " ".	00462000
000542	88	FCTERMNS VALUE " ".	00463000
000543	06	FILLER PICTURE X(2).	00464000
000544	05	TCADLPCB PICTURE S9(8) USAGE IS COMPUTATIONAL.	00465000
000545	05	TCADLPSB PICTURE X(8).	00466000
000546	05	TCADLSSA PICTURE S9(8) USAGE IS COMPUTATIONAL.	00467000
000547	05	TCADLPAR PICTURE S9(8) USAGE IS COMPUTATIONAL.	00468000
000548	05	TCADLECB PICTURE S9(8) USAGE IS COMPUTATIONAL.	00469000
000549	05	FILLER REDEFINES TCADLECB.	00470000
000550	06	TCADLLAN PICTURE X(4).	00471000
000551	05	TCADLFUN PICTURE X(4).	00472000
000552	05	FILLER PICTURE X(64).	00473000
000553	04	FILLER.	00474000
000554	05	TCATRF1 PICTURE S9(8) USAGE IS COMPUTATIONAL.	00475000
000555	05	FILLER REDEFINES TCATRF1.	00476000
000556	06	TCATRF1H PICTURE 9(4) USAGE IS COMPUTATIONAL.	00477000
000557	06	FILLER PICTURE X(2).	00478000
000558	05	FILLER REDEFINES TCATRF1.	00479000
000559	06	TCATRF1F PICTURE S9(8) USAGE IS COMPUTATIONAL.	00480000
000560	05	FILLER REDEFINES TCATRF1.	00481000
000561	06	TCATRF1C PICTURE X(4).	00482000
000562	05	FILLER REDEFINES TCATRF1.	00483000
000563	06	TCATRF1P PICTURE 9(7) USAGE IS COMPUTATIONAL-3.	00484000
000564	05	FILLER REDEFINES TCATRF1.	00485000
000565	06	TCATRF1A PICTURE X.	00486000
000566	06	FILLER PICTURE X(3).	00487000
000567	05	TCATRF2 PICTURE S9(8) USAGE IS COMPUTATIONAL.	00488000
000568	05	FILLER REDEFINES TCATRF2.	00489000
000569	06	TCATRF2H PICTURE 9(4) USAGE IS COMPUTATIONAL.	00490000
000570	06	FILLER PICTURE X(2).	00491000
000571	05	FILLER REDEFINES TCATRF2.	00492000
000572	06	TCATRF2F PICTURE S9(8) USAGE IS COMPUTATIONAL.	00493000
000573	05	FILLER REDEFINES TCATRF2.	00494000
000574	06	TCATRF2C PICTURE X(4).	00495000
000575	05	FILLER REDEFINES TCATRF2.	00496000
000576	06	TCATRF2P PICTURE 9(7) USAGE IS COMPUTATIONAL-3.	00497000
000577	05	FILLER REDEFINES TCATRF2.	00498000
000578	06	TCATRF2A PICTURE X.	00499000
000579	06	FILLER PICTURE X(3).	00500000
000580	05	TCATRRI PICTURE 9(4) USAGE IS COMPUTATIONAL.	00501000
000581	05	TCATRRI1 PICTURE 9(4) USAGE IS COMPUTATIONAL.	00502000
000582	05	FILLER PICTURE X(4).	00503000
000583	05	TCAJCAAD PICTURE S9(8) USAGE IS COMPUTATIONAL.	00504000

```

000584          05 TCAATAC PICTURE S9(8) USAGE IS COMPUTATIONAL.      00505000
000585          03 FILLER.                                             00506000
000586          04 TCACSPE PICTURE XXXX.                               00507000
000587          04 TCANXTID PICTURE X(4).                               00508000
000588          01 PCB-ADDR.                                           00509000
000589          02 PCB1-ADDR PIC S9(8) COMP.                            00510000
000590          01 PCB1.                                               00511000
000591          02 DBD-NAME PIC X(8).                                   00512000
000592          02 SEG-LEVEL PIC XX.                                    00513000
000593          02 STATUS-CODE PIC XX.                                  00514000
000594          02 PROC-OPTIONS PIC X(4).                              00515000
000595          02 RESERVE-DLI PIC S9(5) COMP.                         00516000
000596          02 SEG-NAME-FB PIC X(8).                               00517000
000597          02 LENGTH-FB-KEY PIC S9(5) COMP.                       00518000
000598          02 NUMB-SENS-SEGS PIC S9(5) COMP.                     00519000
000599          02 KEY-FB-AREA PIC X(30).                             00520000
000600 *OLD** 01 MAP11I COPY ABJCQOUT.                                  00521000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000601          01 MAP11I. COPY ABJCQOUT REPLACING ==01 MAP11I.== BY ==. 00521000
000602+ * ----- *00001000
000603+ * LICENSED MATERIALS - PROPERTY OF IBM *00002000
000604+ * *00003000
000605+ * 5785-CCC 5785-ABJ 5648-B05 5686-A07 *00004000
000606+ * *00005000
000607+ * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00006000
000608+ * *00007000
000609+ * US GOVERNMENT USERS RESTRICTED RIGHTS - USE, *00008000
000610+ * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP *00009000
000611+ * SCHEDULE CONTRACT WITH IBM CORP. *00010000
000612+ * *00011000
000613+ * ----- *00012000
000614+ 01 MAP11I. 00013000
000615+          02 FILLER PIC X(96). 00014000
000616+ 01 MAP110 REDEFINES MAP11I. 00015000
000617+          02 FILLER PIC X(96). 00016000
000618+ 01 MAP21I REDEFINES MAP11I. 00017000
000619+          02 FILLER PIC X(12). 00018000
000620+          02 SEGNAME1 COMP PIC S9(4). 00019000
000621+          02 SEGNAMEF PICTURE X. 00020000
000622+          02 FILLER REDEFINES SEGNAMEF. 00021000
000623+          03 SEGNAMEA PICTURE X. 00022000
000624+          02 SEGNAMEI PIC X(8). 00023000
000625+          02 SEGCONTL COMP PIC S9(4). 00024000
000626+          02 SEGCONTF PICTURE X. 00025000
000627+          02 FILLER REDEFINES SEGCONTF. 00026000
000628+          03 SEGCONTA PICTURE X. 00027000
000629+          02 SEGCONTI PIC X(70). 00028000
000630+ 01 MAP210 REDEFINES MAP21I. 00029000
000631+          02 FILLER PIC X(12). 00030000
000632+          02 FILLER PICTURE X(3). 00031000
000633+          02 SEGNAMEO PIC X(8). 00032000
000634+          02 FILLER PICTURE X(3). 00033000
000635+          02 SEGCONTO PIC X(70). 00034000
000636+ 01 MAP31I REDEFINES MAP11I. 00035000
  
```

```

000637+      02 FILLER PIC X(12).                00036000
000638+      01 MAP310 REDEFINES MAP31I.          00037000
000639+      02 FILLER PIC X(12).                00038000
000640+      01 MAP41I REDEFINES MAP11I.          00039000
000641+      02 FILLER PIC X(12).                00040000
000642+      01 MAP410 REDEFINES MAP41I.          00041000
000643+      02 FILLER PIC X(12).                00042000
000644+      01 MAP51I REDEFINES MAP11I.          00043000
000645+      02 FILLER PIC X(12).                00044000
000646+      01 MAP510 REDEFINES MAP51I.          00045000
000647+      02 FILLER PIC X(12).                00046000
000648+ *OLD** 01 MAP12I COPY ABJERRMP.            00522000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000649+      01 MAP12I. COPY ABJERRMP REPLACING ==01 MAP12I.== BY ==. 00522000
000650+      * -----*00001000
000651+      * LICENSED MATERIALS - PROPERTY OF IBM          *00002000
000652+      * * *00003000
000653+      * 5785-CCC 5785-ABJ 5648-B05 5686-A07          *00004000
000654+      * * *00005000
000655+      * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00006000
000656+      * * *00007000
000657+      * US GOVERNMENT USERS RESTRICTED RIGHTS - USE,          *00008000
000658+      * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP          *00009000
000659+      * SCHEDULE CONTRACT WITH IBM CORP.                *00010000
000660+      * * *00011000
000661+      * -----*00012000
000662+      01 MAP12I.                00013000
000663+      02 FILLER PIC X(12).          00014000
000664+      02 ERRMSGI COMP PIC S9(4).    00015000
000665+      02 ERRMSGF PICTURE X.         00016000
000666+      02 FILLER REDEFINES ERRMSGF.  00017000
000667+      03 ERRMSGI PICTURE X.         00018000
000668+      02 ERRMSGI PIC X(70).         00019000
000669+      01 MAP120 REDEFINES MAP12I.    00020000
000670+      02 FILLER PIC X(12).          00021000
000671+      02 FILLER PICTURE X(3).        00022000
000672+      02 ERRMSGO PIC X(70).         00023000
000673+ *OLD** 01 MAP13I COPY ABJCIOUT.        00523000 ABJ6088 00 LANGLEVEL 1 COPY IS CHANGED
000674+      01 MAP13I. COPY ABJCIOUT REPLACING ==01 MAP13I.== BY ==. 00523000
000675+      * -----*00001000
000676+      * LICENSED MATERIALS - PROPERTY OF IBM          *00002000
000677+      * * *00003000
000678+      * 5785-CCC 5785-ABJ 5648-B05 5686-A07          *00004000
000679+      * * *00005000
000680+      * (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED. *00006000
000681+      * * *00007000
000682+      * US GOVERNMENT USERS RESTRICTED RIGHTS - USE,          *00008000
000683+      * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP          *00009000
000684+      * SCHEDULE CONTRACT WITH IBM CORP.                *00010000
000685+      * * *00011000
000686+      * -----*00012000
000687+      01 MAP13I.                00013000
000688+      02 FILLER PIC X(12).          00014000
000689+      01 MAP130 REDEFINES MAP13I.    00015000

```



```

000690+      02 FILLER PIC X(12).                00016000
000691      PROCEDURE DIVISION.                  00524000
000692      *OLD** MOVE CSACDTA TO TCACBAR.      00525000 ABJ6207 00 BLL CONVERTED TO SET POINTER
000693      MOVE CSACDTA TO LCP-WS-ADDR-COMP      00525000 SET ADDRESS OF ...
000694      SET ADDRESS OF DFHTCA TO LCP-WS-ADDR-PNTR. ABJ6301 04 31 BIT ESA ADDRESSES WILL BE
                                           TREATED AS NEGATIVE NUMBERS:
                                           RESULTS MAY BE UNPREDICTABLE
                                           *** MANUAL UPDATE RECOMMENDED

000695      EXEC CICS HANDLE CONDITION ERROR(ERRORS) MAPFAIL(CIDL) 00526000
000696      OVERFLOW(PAGE-OVERFLOW) END-EXEC.      00527000
000697      EXEC CICS RECEIVE MAP("MAP1") MAPSET("CINQIN") END-EXEC. 00528000
000698      IF CUSTNOI = SPACES OR CUSTNOL = +0000 00529000
000699      MOVE "CUSTOMER" TO ERRNAMEO            00530000
000700      MOVE SPACES TO ERRNOO GO TO ERR-MSG.    00531000
000701      MOVE "PSBCLIG" TO PSBNAME.             00532000
000702      CALL "CBLTDLI" USING PCB PSBNAME.      00533000
000703      IF TCAFCRC NOT EQUAL TO " " GO TO INTERFACE-ERROR. 00534000
000704      *OLD** MOVE TCADLPCB TO PCB-LIST-PTR.  00535000 ABJ6207 00 BLL CONVERTED TO SET POINTER
000705      MOVE TCADLPCB TO LCP-WS-ADDR-COMP      00535000 SET ADDRESS OF ...
000706      SET ADDRESS OF PCB-ADDR TO LCP-WS-ADDR-PNTR. ABJ6301 04 31 BIT ESA ADDRESSES WILL BE
                                           TREATED AS NEGATIVE NUMBERS:
                                           RESULTS MAY BE UNPREDICTABLE
                                           *** MANUAL UPDATE RECOMMENDED

000707      *OLD** MOVE PCB1-ADDR TO PCB1-PTR.    00536000 ABJ6207 00 BLL CONVERTED TO SET POINTER
000708      MOVE PCB1-ADDR TO LCP-WS-ADDR-COMP      00536000 SET ADDRESS OF ...
000709      SET ADDRESS OF PCB1 TO LCP-WS-ADDR-PNTR. ABJ6301 04 31 BIT ESA ADDRESSES WILL BE
                                           TREATED AS NEGATIVE NUMBERS:
                                           RESULTS MAY BE UNPREDICTABLE
                                           *** MANUAL UPDATE RECOMMENDED

000710      MOVE CUSTNOI TO SSA1KEY.              00537000
000711      MOVE CHECKNOI TO SSA2KEY.             00538000
000712      MOVE LOANNOI TO SSA3KEY.            00539000
000713      IF SSA2KEY NOT = LOW-VALUE GO TO CHECK-PROC. 00540000
000714      IF SSA3KEY NOT = LOW-VALUE GO TO LOAN-PROC. 00541000
000715      CALL "CBLTDLI" USING GU PCB1 DLIO SSA1. 00542000
000716      IF TCAFCRC NOT EQUAL TO " " GO TO INTERFACE-ERROR. 00543000
000717      IF STATUS-CODE = " " GO TO GU-OK.     00544000
000718      IF STATUS-CODE = "GE" MOVE "CUSTOMER" TO ERRNAMEO 00545000
000719      MOVE CUSTNOI TO ERRNOO              00546000
000720      GO TO ERR-MSG.                      00547000
000721      GO TO ERROR1.                      00548000
000722      CHECK-PROC.                        00549000
000723      MOVE CHECKNOI TO SSA2KEY.           00550000
000724      CALL "CBLTDLI" USING GU PCB1 DLIO SSA1 SSA2. 00551000
000725      IF TCAFCRC NOT = " " GO TO INTERFACE-ERROR. 00552000
000726      IF STATUS-CODE = " " GO TO GU-OK.     00553000
000727      IF STATUS-CODE = "GE" MOVE "CHECK" TO ERRNAMEO 00554000
000728      MOVE CHECKNOI TO ERRNOO             00555000
000729      GO TO ERR-MSG.                      00556000
000730      LOAN-PROC.                         00557000
000731      MOVE LOANNOI TO SSA3KEY.           00558000
000732      CALL "CBLTDLI" USING GU PCB1 DLIO SSA1 SSA3. 00559000
000733      IF TCAFCRC NOT = " " GO TO INTERFACE-ERROR. 00560000

```

```

000734      IF STATUS-CODE = " " GO TO GU-OK.                00561000
000735      IF STATUS-CODE = "GE" MOVE "LOAN" TO ERRNAMEO    00562000
000736          MOVE LOANNOI TO ERRNOO                      00563000
000737          GO TO ERR-MSG.                                00564000
000738      GU-OK.                                            00565000
000739          MOVE 1 TO PAGE-OVERFLOW-CTR.                00566000
000740      *OLD** EXEC CICS GETMAIN SET(CINQOUT-PTR) LENGTH(96) END-EXEC. 00567000 ABJ6201 00 POINTER OPTION IN EXEC CICS
000741          EXEC CICS GETMAIN SET(ADDRESS OF MAP11I) LENGTH(96) END-EXEC. 00567000          CHANGED TO ADDRESS OF ...
000742          EXEC CICS SEND MAP("MAP11") MAPSET("CINQOUT") ACCUM 00568000
000743          ERASE PAGING FRSET FREEKB END-EXEC.          00569000
000744      PAGE-BUILD.                                       00570000
000745          MOVE SEG-NAME-FB TO SEGNAMEO.                00571000
000746          MOVE DLIO TO SEGCONTO.                       00572000
000747      SEND-MAP2.                                       00573000
000748          MOVE 2 TO PAGE-OVERFLOW-CTR.                00574000
000749          EXEC CICS SEND MAP("MAP21") MAPSET("CINQOUT") ACCUM 00575000
000750          PAGING FRSET FREEKB END-EXEC.                00576000
000751      GNP-LOOP.                                        00577000
000752          CALL "CBLTDLI" USING GNP PCB1 DLIO.          00578000
000753          IF TCAFCRC NOT = " " GO TO INTERFACE-ERROR. 00579000
000754          IF STATUS-CODE = " " OR STATUS-CODE = "GA"    00580000
000755              OR STATUS-CODE = "GK" GO TO PAGE-BUILD. 00581000
000756          IF STATUS-CODE = "GE" OR STATUS-CODE = "GB" 00582000
000757              GO TO END-GNP-LOOP.                     00583000
000758          GO TO ERROR1.                                 00584000
000759      END-GNP-LOOP.                                    00585000
000760          EXEC CICS SEND MAP("MAP31") MAPSET("CINQOUT") ACCUM 00586000
000761          PAGING FRSET FREEKB END-EXEC.                00587000
000762          EXEC CICS SEND MAP("MAP41") MAPSET("CINQOUT") ACCUM 00588000
000763          PAGING FRSET FREEKB END-EXEC.                00589000
000764      PAGE-OUT.                                        00590000
000765          EXEC CICS SEND PAGE NOAUTOPAGE END-EXEC.    00591000
000766      END-PROG.                                       00592000
000767      PROG-RETURN.                                     00593000
000768          CALL "CBLTDLI" USING TERM.                  00594000
000769          EXEC CICS RETURN TRANSID("CINQ") END-EXEC. 00595000
000770      ERRORS.                                         00596000
000771          PERFORM SAVE-INFO.                           00597000
000772          EXEC CICS DUMPCODE("ERRS") END-EXEC.         00598000
000773          GO TO PROG-RETURN.                            00599000
000774      CIDL.                                          00600000
000775      *OLD** EXEC CICS GETMAIN SET(CIDLOUT-PTR) LENGTH(12) END-EXEC. 00601000 ABJ6201 00 POINTER OPTION IN EXEC CICS
000776          EXEC CICS GETMAIN SET(ADDRESS OF MAP13I) LENGTH(12) END-EXEC. 00601000          CHANGED TO ADDRESS OF ...
000777          MOVE LOW-VALUE TO MAP13O.                   00602000
000778          EXEC CICS SEND MAP("MAP13") MAPSET("CIDLOUT") ERASE END-EXEC. 00603000
000779          EXEC CICS RETURN END-EXEC.                   00604000
000780      PAGE-OVERFLOW.                                  00605000
000781          EXEC CICS SEND MAP("MAP41") MAPSET("CINQOUT") ACCUM 00606000
000782          PAGING FREEKB END-EXEC.                      00607000
000783          EXEC CICS SEND MAP("MAP11") MAPSET("CINQOUT") ACCUM 00608000
000784          ERASE PAGING FRSET FREEKB END-EXEC.        00609000
000785          GO TO GU-OK SEND-MAP2 DEPENDING ON PAGE-OVERFLOW-CTR. 00610000
000786      ERR-MSG.                                        00611000
  
```

```

000787 EXEC CICS SEND MAP("MAP1") MAPSET("CINQIN") ACCUM 00612000
000788 PAGING FREEKB END-EXEC. 00613000
000789 EXEC CICS SEND MAP("MAP2") MAPSET("CINQIN") ACCUM 00614000
000790 PAGING FREEKB END-EXEC. 00615000
000791 EXEC CICS SEND PAGE END-EXEC. 00616000
000792 GO TO END-PROG. 00617000
000793 INTERFACE-ERROR. 00618000
000794 MOVE TCAFCRC TO SAVE-TCAFCRC. 00619000
000795 MOVE TCADLTR TO SAVE-TCADLTR. 00620000
000796 PERFORM SAVE-INFO. 00621000
000797 EXEC CICS DUMP DUMPCODE("INTE") END-EXEC. 00622000
000798 *OLD** EXEC CICS GETMAIN SET(ERRORMP-PTR) LENGTH(85) END-EXEC. 00623000 ABJ6201 00 POINTER OPTION IN EXEC CICS
000799 EXEC CICS GETMAIN SET(ADDRESS OF MAP12I) LENGTH(85) END-EXEC. 00623000 CHANGED TO ADDRESS OF ...
000800 MOVE "*** INTERFACE ERROR. DUMP IN PROGRESS.***" TO ERRMSGO. 00624000
000801 EXEC CICS SEND MAP("MAP12") MAPSET("ERRORMP") ACCUM 00625000
000802 PAGING FREEKB END-EXEC. 00626000
000803 GO TO CIDL. 00627000
000804 ERROR1. 00628000
000805 PERFORM SAVE-INFO. 00629000
000806 EXEC CICS DUMP DUMPCODE("ERRO") END-EXEC. 00630000
000807 *OLD** EXEC CICS GETMAIN SET(ERRORMP-PTR) LENGTH(85) END-EXEC. 00631000 ABJ6201 00 POINTER OPTION IN EXEC CICS
000808 EXEC CICS GETMAIN SET(ADDRESS OF MAP12I) LENGTH(85) END-EXEC. 00631000 CHANGED TO ADDRESS OF ...
000809 MOVE "*** DL/1 CALL ERROR. DUMP IN PROGRESS.***" TO ERRMSGO. 00632000
000810 EXEC CICS SEND MAP("MAP12") MAPSET("ERRORMP") ACCUM 00633000
000811 PAGING FREEKB END-EXEC. 00634000
000812 GO TO CIDL. 00635000
000813 SAVE-INFO. 00636000
000814 MOVE STATUS-CODE TO SAVE-STATUS-CODE. 00637000
000815 MOVE TCACCCA TO SAVE-TCACCCA. 00638000
000816 END-PGM. 00639000
000817 STOP RUN. 00640000 ABJ6126 99 *-----*
* END OF COBOL CONVERSION *
* 5648-B05 COBOL CONVERSION *
*-----*

```

OPTIONS IN EFFECT :

```

Check procedure names ..... YES Source language level ..... DOS/VS COBOL LANGLVL(1)
Flag Report Writer statements... YES CICS ..... YES
Remove obsolete elements ..... YES Lines per report page .....60
Negate implicit EXIT PROGRAM ... YES VSE system date format..... MM/DD/YY
Generate END PROGRAM header .... NO Resequence source lines ..... NO
Compile after converting ..... YES
Flag manual changes (new source) NO Reserved word suffix ..... 74
Add DATE FORMAT clauses (MLE) NO Generate new program..... YES
Remove VALUE clauses in FS & LS YES Generate new copy members ..... YES
FLAG:IF FILE-STATUS (NOT) = "00" YES Replace like-named copy members. NO
Flag BLL cell arithmetic ..... YES Print old source lines ..... YES
BLL cell conversion method..... A Print copy members ..... YES
Search source for literal delim. YES Print diagnostics of level >=... 00
Literal delimiter (QUOTE/APOST). QUOTE Generate tokenization listing... NO
OPTION-15 ..... NO SQL ..... NO

```

HIGHEST SEVERITY MESSAGE FOR THIS CONVERSION: 04

0016 MESSAGES ISSUED

0016 MESSAGES PRINTED

LINEID	MSGID	RC	MESSAGE TEXT
000021	ABJ6212	00	WORKING POINTER FOR CICS ADDED TO WORKING STORAGE
000141	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000206	ABJ6203	00	BLL'S ARE REMOVED
000601	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000649	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000674	ABJ6088	00	LANGLEVEL 1 COPY IS CHANGED
000693	ABJ6207	00	BLL CONVERTED TO SET POINTER SET ADDRESS OF ...
000693	ABJ6301	04	31 BIT ESA ADDRESSES WILL BE TREATED AS NEGATIVE NUMBERS: RESULTS MAY BE UNPREDICTABLE *** MANUAL UPDATE RECOMMENDED
000705	ABJ6207	00	BLL CONVERTED TO SET POINTER SET ADDRESS OF ...
000705	ABJ6301	04	31 BIT ESA ADDRESSES WILL BE TREATED AS NEGATIVE NUMBERS: RESULTS MAY BE UNPREDICTABLE *** MANUAL UPDATE RECOMMENDED
000708	ABJ6207	00	BLL CONVERTED TO SET POINTER SET ADDRESS OF ...
000708	ABJ6301	04	31 BIT ESA ADDRESSES WILL BE TREATED AS NEGATIVE NUMBERS: RESULTS MAY BE UNPREDICTABLE *** MANUAL UPDATE RECOMMENDED
000741	ABJ6201	00	POINTER OPTION IN EXEC CICS CHANGED TO ADDRESS OF ...
000776	ABJ6201	00	POINTER OPTION IN EXEC CICS CHANGED TO ADDRESS OF ...
000799	ABJ6201	00	POINTER OPTION IN EXEC CICS CHANGED TO ADDRESS OF ...
000808	ABJ6201	00	POINTER OPTION IN EXEC CICS CHANGED TO ADDRESS OF ...

Tokenization

In conversion phase 1, the input program is tokenized and written to the TOKEN data set. To get a listing of the input program in its tokenized form, set the **Generate tokenization listing** on Conversion Options panel 1 to Y (for details, see “Setting conversion options” on page 19).

The generated output lists each line of the COBOL program and the tokenization for the line.

The columns on the right hand side of this listing are described below.

SEQ-NO

TOKEN-SEQUENCE

Line number in the COBOL source program.

POS TOKEN-POSITION

Starting position in the COBOL statement.

LNGTH

TOKEN-LENGTH

Length of the token.

TYPE TOKEN-TYPE-CODE

Type of the token.

CODE TOKEN-CHANGE-CODE

Indicates type of processing.

FLAG TOKEN-FLAG

Indicates paragraph, statement, or clause.

These identifiers and their values are described in Appendix E, “Predefined data items,” on page 175.

The following is a partial tokenization listing of the program ABJIVP01.

	SEQ-NO/POS/LNGTH/TYPE/CODE/FLAG
IDENTIFICATION DIVISION.	00001000
IDENTIFICATION	000010 01 014 W 990 01
DIVISION	000010 16 008 W 990
.	000010 24 001 000
PROGRAM-ID. ABJIVP01.	00002000
PROGRAM-ID	000020 01 010 W 990 01
.	000020 11 001 000
ABJIVP01	000020 13 008 W 000
.	000020 21 001 000
* -----	* 00003000
* LICENSED MATERIALS - PROPERTY OF IBM	* 00004000
*	* 00005000
* 5785-CCC 5785-ABJ 5648-B05 5686-A07	* 00006000
*	* 00007000
* (C) COPYRIGHT IBM CORP. 1982, 1998. ALL RIGHTS RESERVED.	* 00008000
*	* 00009000
* US GOVERNMENT USERS RESTRICTED RIGHTS - USE,	* 00010000
* DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP	* 00011000
* SCHEDULE CONTRACT WITH IBM CORP.	* 00012000
*	* 00013000
* -----	* 00014000
REMARKS.	00015000
REMARKS	000150 01 007 W 990 01
THIS PROGRAM IS BEING WRITTEN TO TEST THE PROPER CONVERSION	00016000
FROM OS/VS COBOL SOURCE LANGUAGE TO IBM SOURCE LANGUAGE.	00017000
AUTHOR. XXXXXX.	00018000
AUTHOR	000180 01 006 W 856 01
DATE-WRITTEN. JANUARY 24, 1983.	00019000
DATE-WRITTEN	000190 01 012 W 856 01
NOTE - THE FOLLOWING AREAS ARE ADDRESSED	00020000
1 REMARKS	00021000
2 THEN	00022000
3 OTHERWISE	00023000
4 CURRENT-DATE	00024000
5 TIME-OF-DAY	00025000
6 NOTE	00026000
7 EXAMINE...TALLYING...REPLACING	00027000
8 JUSTIFIED.	00028000
.	00029000
.	00030000
DATE-COMPILED. TODAYS DATE.	00031000
DATE-COMPILED	000310 01 013 W 990 01
EJECT	00032000
ENVIRONMENT DIVISION.	00033000
ENVIRONMENT	000330 01 011 W 990 01
DIVISION	000330 13 008 W 990
.	000330 21 001 000
INPUT-OUTPUT SECTION.	00034000
INPUT-OUTPUT	000340 01 012 W 990 01
SECTION	000340 14 007 W 990
.	000340 21 001 000
FILE-CONTROL.	00035000
FILE-CONTROL	000350 01 012 W 999 01
.	000350 13 001 000
SELECT PRINT-FILE	00036000
SELECT	000360 05 006 W 990 02
ASSIGN TO UT-3330-S-DDPRINT.	00037000
PRINT-FILE	000360 12 010 W 000
ASSIGN	000370 05 006 W 990 02
TO	000370 12 002 W 999
UT-3330-S-DDPRINT	000370 15 017 W 000
.	000370 32 001 000
DATA DIVISION.	00038000
DATA	000380 01 004 W 999 21
DIVISION	000380 06 008 W 990
.	000380 14 001 000
FILE SECTION.	00039000
FILE	000390 01 004 W 999 01
SECTION	000390 06 007 W 990
.	000390 13 001 000
FD PRINT-FILE	00040000
FD	000400 01 002 W 990 02

```

RECORDING MODE IS F                                00041000
PRINT-FILE ::::::::::::::::::::::::::::::::::::: 000400 05 010 W 000
RECORDING ::::::::::::::::::::::::::::::::::::: 000410 05 009 W 999 02
      MODE ::::::::::::::::::::::::::::::::::::: 000410 15 004 W 999
      IS ::::::::::::::::::::::::::::::::::::: 000410 20 002 W 999
LABEL RECORDS ARE STANDARD                         00042000
      F ::::::::::::::::::::::::::::::::::::: 000410 23 001 W 000
LABEL ::::::::::::::::::::::::::::::::::::: 000420 05 005 W 999 02
      RECORDS ::::::::::::::::::::::::::::: 000420 11 007 W 999
      ARE ::::::::::::::::::::::::::::: 000420 19 003 W 999
DATA RECORD IS OUT-LINE.                          00043000
      STANDARD ::::::::::::::::::::: 000420 23 008 W 999
DATA ::::::::::::::::::::: 000430 05 004 W 999 21
      RECORD ::::::::::::::::::::: 000430 10 006 W 999 02
      IS ::::::::::::::::::::: 000430 17 002 W 999
      OUT-LINE ::::::::::::::::::::: 000430 20 008 W 000
      . ::::::::::::::::::::: 000430 28 001 000
01 OUT-LINE PIC X(80).                             00044000
01 ::::::::::::::::::::: 000440 01 002 N 990
      OUT-LINE ::::::::::::::::::::: 000440 05 008 W 000
      PIC ::::::::::::::::::::: 000440 24 003 P 999 02
      X(80) ::::::::::::::::::::: 000440 28 005 P 000
      . ::::::::::::::::::::: 000440 33 001 000
WORKING-STORAGE SECTION.                         00045000
WORKING-STORAGE ::::::::::::::::::::: 000450 01 015 W 999 01
      SECTION ::::::::::::::::::::: 000450 17 007 W 999
      . ::::::::::::::::::::: 000450 24 001 000
77 MY-COUNTER PIC 9(5) VALUE 0.                   00046000
77 ::::::::::::::::::::: 000460 01 002 N 990
      MY-COUNTER ::::::::::::::::::::: 000460 05 010 W 000
      PIC ::::::::::::::::::::: 000460 24 003 P 999 02
      9(5) ::::::::::::::::::::: 000460 28 004 P 000
      VALUE ::::::::::::::::::::: 000460 34 005 W 999 02
      0 ::::::::::::::::::::: 000460 40 001 N 999
      . ::::::::::::::::::::: 000460 41 001 000
77 TRIPSWCH PIC 9 VALUE 0.                       00047000
77 ::::::::::::::::::::: 000470 01 002 N 990
      TRIPSWCH ::::::::::::::::::::: 000470 05 008 W 000
      PIC ::::::::::::::::::::: 000470 24 003 P 999 02
      9 ::::::::::::::::::::: 000470 28 001 P 000
      VALUE ::::::::::::::::::::: 000470 34 005 W 999 02
      0 ::::::::::::::::::::: 000470 40 001 N 999
      . ::::::::::::::::::::: 000470 41 001 000
77 PASSWCH PIC 9 VALUE 0.                       00048000
77 ::::::::::::::::::::: 000480 01 002 N 990
      PASSWCH ::::::::::::::::::::: 000480 05 007 W 000
      PIC ::::::::::::::::::::: 000480 24 003 P 999 02
      9 ::::::::::::::::::::: 000480 28 001 P 000
      VALUE ::::::::::::::::::::: 000480 34 005 W 999 02
      0 ::::::::::::::::::::: 000480 40 001 N 999
      . ::::::::::::::::::::: 000480 41 001 000
77 FAILSWCH PIC 9 VALUE 1.                      00049000
77 ::::::::::::::::::::: 000490 01 002 N 990
      FAILSWCH ::::::::::::::::::::: 000490 05 008 W 000
      PIC ::::::::::::::::::::: 000490 24 003 P 999 02
      9 ::::::::::::::::::::: 000490 28 001 P 000
      VALUE ::::::::::::::::::::: 000490 34 005 W 999 02
      1 ::::::::::::::::::::: 000490 40 001 N 999
      . ::::::::::::::::::::: 000490 41 001 000
77 CURRFLAG PIC 9 VALUE 0.                      00050000
77 ::::::::::::::::::::: 000500 01 002 N 990
      CURRFLAG ::::::::::::::::::::: 000500 05 008 W 000
      PIC ::::::::::::::::::::: 000500 24 003 P 999 02
      9 ::::::::::::::::::::: 000500 28 001 P 000
      VALUE ::::::::::::::::::::: 000500 34 005 W 999 02
      0 ::::::::::::::::::::: 000500 40 001 N 999
      . ::::::::::::::::::::: 000500 41 001 000
77 TOFDFLAG PIC 9 VALUE 0.                     00051000
77 ::::::::::::::::::::: 000510 01 002 N 990
      TOFDFLAG ::::::::::::::::::::: 000510 05 008 W 000
      PIC ::::::::::::::::::::: 000510 24 003 P 999 02

```

```

          9 ..... 000510 28 001 P 000
          VALUE ..... 000510 34 005 W 990 02
          0 ..... 000510 40 001 N 999
          . ..... 000510 41 001 000
77 I          PIC 9      VALUE 0.          00052000
77 ..... 000520 01 002 N 990
  I ..... 000520 05 001 W 000
          PIC ..... 000520 24 003 P 990 02
          9 ..... 000520 28 001 P 000
          VALUE ..... 000520 34 005 W 990 02
          0 ..... 000520 40 001 N 999
          . ..... 000520 41 001 000
77 DATE1      PIC X(8)  VALUE SPACES.     00053000
77 ..... 000530 01 002 N 990
  DATE1 ..... 000530 05 005 W 000
          PIC ..... 000530 24 003 P 990 02
          X(8) ..... 000530 28 004 P 000
          VALUE ..... 000530 34 005 W 990 02
          SPACES ..... 000530 40 006 W 999
          . ..... 000530 46 001 000
77 DATE2      PIC X(8)  VALUE SPACES.     00054000
77 ..... 000540 01 002 N 990
  DATE2 ..... 000540 05 005 W 000
          PIC ..... 000540 24 003 P 990 02
          X(8) ..... 000540 28 004 P 000
          VALUE ..... 000540 34 005 W 990 02
          SPACES ..... 000540 40 006 W 999
          . ..... 000540 46 001 000
77 DATE3      PIC X(8)  VALUE SPACES.     00055000
77 ..... 000550 01 002 N 990
  DATE3 ..... 000550 05 005 W 000
          PIC ..... 000550 24 003 P 990 02
          X(8) ..... 000550 28 004 P 000
          VALUE ..... 000550 34 005 W 990 02
          SPACES ..... 000550 40 006 W 999
          . ..... 000550 46 001 000
77 TIME1      PIC X(6)  VALUE SPACES.     00056000
77 ..... 000560 01 002 N 990
  TIME1 ..... 000560 05 005 W 000
          PIC ..... 000560 24 003 P 990 02
          X(6) ..... 000560 28 004 P 000
          VALUE ..... 000560 34 005 W 990 02
          SPACES ..... 000560 40 006 W 999
          . ..... 000560 46 001 000
77 TIME2      PIC X(6)  VALUE SPACES.     00057000
77 ..... 000570 01 002 N 990
  TIME2 ..... 000570 05 005 W 000
          PIC ..... 000570 24 003 P 990 02
          X(6) ..... 000570 28 004 P 000
          VALUE ..... 000570 34 005 W 990 02
          SPACES ..... 000570 40 006 W 999
          . ..... 000570 46 001 000
77 TIME3      PIC X(6)  VALUE SPACES.     00058000
77 ..... 000580 01 002 N 990
  TIME3 ..... 000580 05 005 W 000
          PIC ..... 000580 24 003 P 990 02
          X(6) ..... 000580 28 004 P 000
          VALUE ..... 000580 34 005 W 990 02
          SPACES ..... 000580 40 006 W 999
          . ..... 000580 46 001 000
          00059000
01 ORIGINAL-NUMBER. 00060000
01 ..... 000600 01 002 N 990
  ORIGINAL-NUMBER ..... 000600 05 015 W 000
          . ..... 000600 20 001 000
          05 FILLER PIC 9(18) VALUE 0. 00061000
          05 ..... 000610 05 002 N 000
          FILLER ..... 000610 09 006 W 999
          PIC ..... 000610 20 003 P 990 02
          9(18) ..... 000610 24 005 P 000
          VALUE ..... 000610 30 005 W 990 02

```

```

0 .....: 000610 36 001 N 999
. ....: 000610 37 001 000
05 FILLER PIC 9(18) VALUE 0. 00062000
05 .....: 000620 05 002 N 000
FILLER .....: 000620 09 006 W 999
PIC .....: 000620 20 003 P 990 02
9(18) .....: 000620 24 005 P 000
VALUE .....: 000620 30 005 W 990 02
0 .....: 000620 36 001 N 999
. ....: 000620 37 001 000
05 FILLER PIC 9(18) VALUE 00000009099843576. 00063000
05 .....: 000630 05 002 N 000
FILLER .....: 000630 09 006 W 999
PIC .....: 000630 20 003 P 990 02
9(18) .....: 000630 24 005 P 000
VALUE .....: 000630 30 005 W 990 02
00000009099843576 .....: 000630 36 018 N 000
. ....: 000630 54 001 000
05 FILLER PIC 9(18) VALUE 1212121212121290. 00064000
05 .....: 000640 05 002 N 000
FILLER .....: 000640 09 006 W 999
PIC .....: 000640 20 003 P 990 02
9(18) .....: 000640 24 005 P 000
VALUE .....: 000640 30 005 W 990 02
1212121212121290 .....: 000640 36 018 N 000
. ....: 000640 54 001 000
00065000
01 THIS-DEF REDEFINES ORIGINAL-NUMBER. 00066000
01 .....: 000660 01 002 N 990
THIS-DEF .....: 000660 05 008 W 000
REDEFINES .....: 000660 14 009 W 990 02
ORIGINAL-NUMBER .....: 000660 24 015 W 000
. ....: 000660 39 001 000
03 A-NUMBER OCCURS 2 TIMES. 00067000
03 .....: 000670 05 002 N 000
A-NUMBER .....: 000670 09 008 W 000
OCCURS .....: 000670 18 006 W 990 02
2 .....: 000670 25 001 N 000
TIMES .....: 000670 27 005 W 999
. ....: 000670 32 001 000
05 LINE1 PIC 9(18). 00068000
05 .....: 000680 09 002 N 000
LINE1 .....: 000680 13 005 W 000
PIC .....: 000680 24 003 P 990 02
9(18) .....: 000680 28 005 P 000
. ....: 000680 33 001 000
05 LINE2 PIC 9(18). 00069000
05 .....: 000690 09 002 N 000
LINE2 .....: 000690 13 005 W 000
PIC .....: 000690 24 003 P 990 02
9(18) .....: 000690 28 005 P 000
. ....: 000690 33 001 000
00070000
01 A-POEM. 00071000
01 .....: 000710 01 002 N 990
A-POEM .....: 000710 05 006 W 000
. ....: 000710 11 001 000
03 LINE1. 00072000
03 .....: 000720 05 002 N 000
LINE1 .....: 000720 09 005 W 000
. ....: 000720 14 001 000
05 FILLER PIC X(20) VALUE "ROSES ARE RED VIOLET". 00073000
05 .....: 000730 09 002 N 000
FILLER .....: 000730 13 006 W 999
PIC .....: 000730 24 003 P 990 02
X(20) .....: 000730 28 005 P 000
VALUE .....: 000730 34 005 W 990 02
"ROSES ARE RED VIOLET" .....: 000730 40 022 L 864 00
. ....: 000730 62 001 000
05 FILLER PIC X(20) VALUE "S ARE BLUE, ". 00074000
05 .....: 000740 09 002 N 000

```



```

FILLER .....: 000740 13 006 W 999
PIC .....: 000740 24 003 P 990 02
X(20) .....: 000740 28 005 P 000
VALUE .....: 000740 34 005 W 990 02
"S ARE BLUE, " .....: 000740 40 022 L 864 00
. ....: 000740 62 001 000

03 LINE2. 00075000
03 .....: 000750 05 002 N 000
LINE2 .....: 000750 09 005 W 000
. ....: 000750 14 001 000

05 FILLER PIC X(20) VALUE "SUGAR IS SWEET AND S". 00076000
05 .....: 000760 09 002 N 000
FILLER .....: 000760 13 006 W 999
PIC .....: 000760 24 003 P 990 02
X(20) .....: 000760 28 005 P 000
VALUE .....: 000760 34 005 W 990 02
"SUGAR IS SWEET AND S" .....: 000760 40 022 L 864 00
. ....: 000760 62 001 000

05 FILLER PIC X(20) VALUE "O ARE YOU. " 00077000
05 .....: 000770 09 002 N 000
FILLER .....: 000770 13 006 W 999
PIC .....: 000770 24 003 P 990 02
X(20) .....: 000770 28 005 P 000
VALUE .....: 000770 34 005 W 990 02
"O ARE YOU. " .....: 000770 40 022 L 864 00
. ....: 000770 62 001 000

00078000
00079000
01 FAIL1CON2. 00080000
01 .....: 000800 01 002 N 990
FAIL1CON2 .....: 000800 05 009 W 000
. ....: 000800 14 001 000

03 FILLER PIC XX VALUE SPACES. 00081000
03 .....: 000810 05 002 N 000
FILLER .....: 000810 09 006 W 999
PIC .....: 000810 24 003 P 990 02
XX .....: 000810 28 002 P 000
VALUE .....: 000810 34 005 W 990 02
SPACES .....: 000810 40 006 W 999
. ....: 000810 46 001 000

03 CPLACE PIC X(20) VALUE SPACES. 00082000
03 .....: 000820 05 002 N 000
CPLACE .....: 000820 09 006 W 000
PIC .....: 000820 24 003 P 990 02
X(20) .....: 000820 28 005 P 000
VALUE .....: 000820 34 005 W 990 02
SPACES .....: 000820 40 006 W 999
. ....: 000820 46 001 000

00083000
01 FAIL2CON. 00084000
01 .....: 000840 01 002 N 990
FAIL2CON .....: 000840 05 008 W 000
. ....: 000840 13 001 000

03 FILLER PIC X(20) VALUE "ALL THREE READINGS O". 00085000
03 .....: 000850 05 002 N 000
FILLER .....: 000850 09 006 W 999
PIC .....: 000850 24 003 P 990 02
X(20) .....: 000850 28 005 P 000
VALUE .....: 000850 34 005 W 990 02
"ALL THREE READINGS O" .....: 000850 40 022 L 864 00
. ....: 000850 62 001 000

03 FILLER PIC X(20) VALUE "F 'CURRENT-DATE' SHO". 00086000
03 .....: 000860 05 002 N 000
FILLER .....: 000860 09 006 W 999
PIC .....: 000860 24 003 P 990 02
X(20) .....: 000860 28 005 P 000
VALUE .....: 000860 34 005 W 990 02
"F 'CURRENT-DATE' SHO" .....: 000860 40 022 L 864 00
. ....: 000860 62 001 000

03 FILLER PIC X(20) VALUE "ULD BE THE SAME, BUT". 00087000

```

```

03 .....: 000870 05 002 N 000
    FILLER .....: 000870 09 006 W 999
        PIC .....: 000870 24 003 P 990 02
        X(20) .....: 000870 28 005 P 000
        VALUE .....: 000870 34 005 W 990 02
            "ULD BE THE SAME, BUT" .....: 000870 40 022 L 864 00
            . .....: 000870 62 001 000

03 FILLER PIC X(20) VALUE " THEY ARE: ". 00088000
03 .....: 000880 05 002 N 000
    FILLER .....: 000880 09 006 W 999
        PIC .....: 000880 24 003 P 990 02
        X(20) .....: 000880 28 005 P 000
        VALUE .....: 000880 34 005 W 990 02
            " THEY ARE: " .....: 000880 40 022 L 864 00
            . .....: 000880 62 001 000
                                00089000
                                00090000
01 FAIL2CON2.
01 .....: 000900 01 002 N 990
    FAIL2CON2 .....: 000900 05 009 W 000
        . .....: 000900 14 001 000

03 FILLER PIC XX VALUE SPACES. 00091000
03 .....: 000910 05 002 N 000
    FILLER .....: 000910 09 006 W 999
        PIC .....: 000910 24 003 P 990 02
        XX .....: 000910 28 002 P 000
        VALUE .....: 000910 34 005 W 990 02
            SPACES .....: 000910 40 006 W 999
            . .....: 000910 46 001 000

03 DPLACE PIC X(8) VALUE SPACES. 00092000
03 .....: 000920 05 002 N 000
    DPLACE .....: 000920 09 006 W 000
        PIC .....: 000920 24 003 P 990 02
        X(8) .....: 000920 28 004 P 000
        VALUE .....: 000920 34 005 W 990 02
            SPACES .....: 000920 40 006 W 999
            . .....: 000920 46 001 000
                                00093000
                                00094000
01 FAIL3CON.
01 .....: 000940 01 002 N 990
    FAIL3CON .....: 000940 05 008 W 000
        . .....: 000940 13 001 000

03 FILLER PIC X(20) VALUE "THE THREE READINGS O". 00095000
03 .....: 000950 05 002 N 000
    FILLER .....: 000950 09 006 W 999
        PIC .....: 000950 24 003 P 990 02
        X(20) .....: 000950 28 005 P 000
        VALUE .....: 000950 34 005 W 990 02
            "THE THREE READINGS O" .....: 000950 40 022 L 864 00
            . .....: 000950 62 001 000

03 FILLER PIC X(20) VALUE "F 'TIME-OF-DAY' SHOU". 00096000
03 .....: 000960 05 002 N 000
    FILLER .....: 000960 09 006 W 999
        PIC .....: 000960 24 003 P 990 02
        X(20) .....: 000960 28 005 P 000
        VALUE .....: 000960 34 005 W 990 02
            "F 'TIME-OF-DAY' SHOU" .....: 000960 40 022 L 864 00
            . .....: 000960 62 001 000

03 FILLER PIC X(20) VALUE "LD BE EQUAL OR IN AS". 00097000
03 .....: 000970 05 002 N 000
    FILLER .....: 000970 09 006 W 999
        PIC .....: 000970 24 003 P 990 02
        X(20) .....: 000970 28 005 P 000
        VALUE .....: 000970 34 005 W 990 02
            "LD BE EQUAL OR IN AS" .....: 000970 40 022 L 864 00
            . .....: 000970 62 001 000

03 FILLER PIC X(20) VALUE "CENDING ORDER, ". 00098000
03 .....: 000980 05 002 N 000
    FILLER .....: 000980 09 006 W 999
        PIC .....: 000980 24 003 P 990 02
        X(20) .....: 000980 28 005 P 000
        VALUE .....: 000980 34 005 W 990 02

```

```

                                "CENDING ORDER,      " .....: 000980 40 022 L 864 00
                                . .....: 000980 62 001 000
                                00099000
01 FAIL3CON1.                    00100000
01 .....: 001000 01 002 N 990
  FAIL3CON1 .....: 001000 05 009 W 000
  . .....: 001000 14 001 000
03 FILLER      PIC X(20) VALUE "BUT THEY ARE:      ". 00101000
03 .....: 001010 05 002 N 000
  FILLER .....: 001010 09 006 W 999
                                PIC .....: 001010 24 003 P 990 02
                                X(20) .....: 001010 28 005 P 000
                                VALUE .....: 001010 34 005 W 990 02
                                "BUT THEY ARE:      " .....: 001010 40 022 L 864 00
                                . .....: 001010 62 001 000
                                00102000
01 FAIL3CON2.                    00103000
01 .....: 001030 01 002 N 990
  FAIL3CON2 .....: 001030 05 009 W 000
  . .....: 001030 14 001 000
03 FILLER      PIC XX  VALUE SPACES.              00104000
03 .....: 001040 05 002 N 000
  FILLER .....: 001040 09 006 W 999
                                PIC .....: 001040 24 003 P 990 02
                                XX .....: 001040 28 002 P 000
                                VALUE .....: 001040 34 005 W 990 02
                                SPACES .....: 001040 40 006 W 999
                                . .....: 001040 46 001 000
03 TPLACE      PIC X(6)  VALUE SPACES.            00105000
03 .....: 001050 05 002 N 000
  TPLACE .....: 001050 09 006 W 000
                                PIC .....: 001050 24 003 P 990 02
                                X(6) .....: 001050 28 004 P 000
                                VALUE .....: 001050 34 005 W 990 02
                                SPACES .....: 001050 40 006 W 999
                                . .....: 001050 46 001 000
                                00106000
01 FAILCON.                    00107000
01 .....: 001070 01 002 N 990
  FAILCON .....: 001070 05 007 W 000
  . .....: 001070 12 001 000
03 FILLER      PIC X(20) VALUE "TEST CASE SAMPLE  F". 00108000
03 .....: 001080 05 002 N 000
  FILLER .....: 001080 09 006 W 999
                                PIC .....: 001080 24 003 P 990 02
                                X(20) .....: 001080 28 005 P 000
                                VALUE .....: 001080 34 005 W 990 02
                                "TEST CASE SAMPLE  F" .....: 001080 40 022 L 864 00
                                . .....: 001080 62 001 000
03 FILLER      PIC X(20) VALUE "AILED.              ". 00109000
03 .....: 001090 05 002 N 000
  FILLER .....: 001090 09 006 W 999
                                PIC .....: 001090 24 003 P 990 02
                                X(20) .....: 001090 28 005 P 000
                                VALUE .....: 001090 34 005 W 990 02
                                "AILED.              " .....: 001090 40 022 L 864 00
                                . .....: 001090 62 001 000
                                00110000
01 SUCCESS.                    00111000
01 .....: 001110 01 002 N 990
  SUCCESS .....: 001110 05 007 W 000
  . .....: 001110 12 001 000
03 FILLER      PIC X(20) VALUE "TEST CASE SAMPLE  I". 00112000
03 .....: 001120 05 002 N 000
  FILLER .....: 001120 09 006 W 999
                                PIC .....: 001120 24 003 P 990 02
                                X(20) .....: 001120 28 005 P 000
                                VALUE .....: 001120 34 005 W 990 02
                                "TEST CASE SAMPLE  I" .....: 001120 40 022 L 864 00
                                . .....: 001120 62 001 000
03 FILLER      PIC X(20) VALUE "S SUCCESSFUL.      ". 00113000

```

```

03 ..... 001130 05 002 N 000
FILLER ..... 001130 09 006 W 999
      PIC ..... 001130 24 003 P 990 02
      X(20) ..... 001130 28 005 P 000
      VALUE ..... 001130 34 005 W 990 02
      "S SUCCESSFUL. " ..... 001130 40 022 L 864 00
      . ..... 001130 62 001 000

EJECT ..... 00114000
PROCEDURE DIVISION. ..... 00115000
PROCEDURE ..... 001150 01 009 W 990 01
      DIVISION ..... 001150 11 008 W 990
      . ..... 001150 19 001 000

THIS-IS-A SECTION. ..... 00116000
THIS-IS-A ..... 001160 01 009 W 860 01
      SECTION ..... 001160 11 007 W 990
      . ..... 001160 18 001 000

START-HERE. ..... 00117000
START-HERE ..... 001170 01 010 W 860 01
      . ..... 001170 11 001 000

MOVE TIME-OF-DAY TO TIME1 ..... 00118000
MOVE ..... 001180 05 004 W 851 03
      TIME-OF-DAY ..... 001180 10 011 W 990
      TO ..... 001180 22 002 W 999

OPEN OUTPUT PRINT-FILE ..... 00119000
      TIME1 ..... 001180 25 005 W 000
OPEN ..... 001190 05 004 W 990 03
      OUTPUT ..... 001190 10 006 W 999

MOVE CURRENT-DATE TO DATE1 ..... 00120000
      PRINT-FILE ..... 001190 17 010 W 000
MOVE ..... 001200 05 004 W 851 03
      CURRENT-DATE ..... 001200 10 012 W 990
      TO ..... 001200 23 002 W 999

MOVE CURRENT-DATE TO DATE2 ..... 00121000
      DATE1 ..... 001200 26 005 W 000
MOVE ..... 001210 05 004 W 851 03
      CURRENT-DATE ..... 001210 10 012 W 990
      TO ..... 001210 23 002 W 999

MOVE CURRENT-DATE TO DATE3. ..... 00122000
      DATE2 ..... 001210 26 005 W 000
MOVE ..... 001220 05 004 W 851 03
      CURRENT-DATE ..... 001220 10 012 W 990
      TO ..... 001220 23 002 W 999
      DATE3 ..... 001220 26 005 W 000
      . ..... 001220 31 001 000

MOVE TIME-OF-DAY TO TIME2. ..... 00123000
MOVE ..... 001240 05 004 W 851 03
      TIME-OF-DAY ..... 001240 10 011 W 990
      TO ..... 001240 22 002 W 999
      TIME2 ..... 001240 25 005 W 000
      . ..... 001240 30 001 000

IF DATE1 EQUAL DATE2 AND EQUAL DATE3 THEN ..... 00125000
IF ..... 001250 05 002 W 999 03
      DATE1 ..... 001250 08 005 W 000
      EQUAL ..... 001250 14 005 W 991
      DATE2 ..... 001250 20 005 W 000
      AND ..... 001250 26 003 W 999
      EQUAL ..... 001250 30 005 W 991
      DATE3 ..... 001250 36 005 W 000

      NEXT SENTENCE ..... 00126000
      THEN ..... 001250 42 004 W 990 03
      NEXT ..... 001260 09 004 W 999 03

      OTHERWISE ..... 00127000
      SENTENCE ..... 001260 14 008 W 999
      MOVE FAILSWCH TO TRIPSWCH ..... 00128000

      OTHERWISE ..... 001270 05 009 W 990
      MOVE ..... 001280 09 004 W 851 03
      FAILSWCH ..... 001280 14 008 W 000
      TO ..... 001280 23 002 W 999
      MOVE DATE1 TO DPLACE ..... 00129000
      TRIPSWCH ..... 001280 26 008 W 000

```

```

MOVE ..... 001290 09 004 W 851 03
  DATE1 ..... 001290 14 005 W 000
    TO ..... 001290 20 002 W 999
WRITE OUT-LINE FROM FAIL2CON          00130000
  DPLACE ..... 001290 23 006 W 000
WRITE ..... 001300 09 005 W 990 03
  OUT-LINE ..... 001300 15 008 W 000
    FROM ..... 001300 24 004 W 999
WRITE OUT-LINE FROM FAIL2CON2        00131000
  FAIL2CON ..... 001300 29 008 W 000
WRITE ..... 001310 09 005 W 990 03
  OUT-LINE ..... 001310 15 008 W 000
    FROM ..... 001310 24 004 W 999
MOVE DATE2 TO DPLACE                 00132000
  FAIL2CON2 ..... 001310 29 009 W 000
MOVE ..... 001320 09 004 W 851 03
  DATE2 ..... 001320 14 005 W 000
    TO ..... 001320 20 002 W 999
WRITE OUT-LINE FROM FAIL2CON2        00133000
  DPLACE ..... 001320 23 006 W 000
WRITE ..... 001330 09 005 W 990 03
  OUT-LINE ..... 001330 15 008 W 000
    FROM ..... 001330 24 004 W 999
MOVE DATE3 TO DPLACE                 00134000
  FAIL2CON2 ..... 001330 29 009 W 000
MOVE ..... 001340 09 004 W 851 03
  DATE3 ..... 001340 14 005 W 000
    TO ..... 001340 20 002 W 999
WRITE OUT-LINE FROM FAIL2CON2.       00135000
  DPLACE ..... 001340 23 006 W 000
WRITE ..... 001350 09 005 W 990 03
  OUT-LINE ..... 001350 15 008 W 000
    FROM ..... 001350 24 004 W 999
  FAIL2CON2 ..... 001350 29 009 W 000
    . ..... 001350 38 001 000
MOVE TIME-OF-DAY TO TIME3.           00136000
MOVE ..... 001360 05 004 W 851 03
  TIME-OF-DAY ..... 001360 10 011 W 990
    TO ..... 001360 22 002 W 999
  TIME3 ..... 001360 25 005 W 000
    . ..... 001360 30 001 000
IF (TIME1 LESS THAN TIME2 OR EQUAL TIME2) AND 00137000
IF ..... 001370 05 002 W 999 03
( ..... 001370 08 001 000
  TIME1 ..... 001370 09 005 W 000
  LESS ..... 001370 15 004 W 991
  THAN ..... 001370 20 004 W 990
    TIME2 ..... 001370 25 005 W 000
    OR ..... 001370 31 002 W 999
    EQUAL ..... 001370 34 005 W 991
      TIME2 ..... 001370 40 005 W 000
    ) ..... 001370 45 001 863 00
(TIME2 LESS THAN TIME3 OR EQUAL TIME3) THEN 00138000
AND ..... 001370 47 003 W 999
( ..... 001380 08 001 000
  TIME2 ..... 001380 09 005 W 000
  LESS ..... 001380 15 004 W 991
  THAN ..... 001380 20 004 W 990
    TIME3 ..... 001380 25 005 W 000
    OR ..... 001380 31 002 W 999
    EQUAL ..... 001380 34 005 W 991
      TIME3 ..... 001380 40 005 W 000
    ) ..... 001380 45 001 863 00
NEXT SENTENCE                        00139000
  THEN ..... 001380 47 004 W 990 03
NEXT ..... 001390 09 004 W 999 03
OTHERWISE ..... 00140000
  SENTENCE ..... 001390 14 008 W 999
MOVE FAILSWCH TO TRIPSWCH           00141000
OTHERWISE ..... 001400 05 009 W 990

```

```

MOVE ..... 001410 09 004 W 851 03
  FAILSWCH ..... 001410 14 008 W 000
    TO ..... 001410 23 002 W 999
MOVE TIME1 TO TPLACE 00142000
  TRIPSWCH ..... 001410 26 008 W 000
MOVE ..... 001420 09 004 W 851 03
  TIME1 ..... 001420 14 005 W 000
    TO ..... 001420 20 002 W 999
WRITE OUT-LINE FROM FAIL3CON 00143000
  TPLACE ..... 001420 23 006 W 000
WRITE ..... 001430 09 005 W 990 03
  OUT-LINE ..... 001430 15 008 W 000
    FROM ..... 001430 24 004 W 999
WRITE OUT-LINE FROM FAIL3CON1 00144000
  FAIL3CON ..... 001430 29 008 W 000
WRITE ..... 001440 09 005 W 990 03
  OUT-LINE ..... 001440 15 008 W 000
    FROM ..... 001440 24 004 W 999
WRITE OUT-LINE FROM FAIL3CON2 00145000
  FAIL3CON1 ..... 001440 29 009 W 000
WRITE ..... 001450 09 005 W 990 03
  OUT-LINE ..... 001450 15 008 W 000
    FROM ..... 001450 24 004 W 999
MOVE TIME2 TO TPLACE 00146000
  FAIL3CON2 ..... 001450 29 009 W 000
MOVE ..... 001460 09 004 W 851 03
  TIME2 ..... 001460 14 005 W 000
    TO ..... 001460 20 002 W 999
WRITE OUT-LINE FROM FAIL3CON2 00147000
  TPLACE ..... 001460 23 006 W 000
WRITE ..... 001470 09 005 W 990 03
  OUT-LINE ..... 001470 15 008 W 000
    FROM ..... 001470 24 004 W 999
MOVE TIME3 TO TPLACE 00148000
  FAIL3CON2 ..... 001470 29 009 W 000
MOVE ..... 001480 09 004 W 851 03
  TIME3 ..... 001480 14 005 W 000
    TO ..... 001480 20 002 W 999
WRITE OUT-LINE FROM FAIL3CON2. 00149000
  TPLACE ..... 001480 23 006 W 000
WRITE ..... 001490 09 005 W 990 03
  OUT-LINE ..... 001490 15 008 W 000
    FROM ..... 001490 24 004 W 999
      FAIL3CON2 ..... 001490 29 009 W 000
        . ..... 001490 38 001 000
AFTER-THOUGHT. 00150000
AFTER-THOUGHT ..... 001500 01 013 W 860 01
  . ..... 001500 14 001 000
EXAMINE A-POEM TALLYING ALL SPACES REPLACING BY "*" 00151000
EXAMINE ..... 001510 05 007 W 990 03
  A-POEM ..... 001510 13 006 W 000
    TALLYING ..... 001510 20 008 W 999
      ALL ..... 001510 29 003 W 990
        SPACES ..... 001510 33 006 W 999
          REPLACING ..... 001510 40 009 W 999 02
            BY ..... 001510 50 002 W 999
              "*" ..... 001510 53 003 L 864 00
MOVE TALLY TO MY-COUNTER 00152000
MOVE ..... 001520 05 004 W 851 03
  TALLY ..... 001520 10 005 W 999
    TO ..... 001520 16 002 W 999
MOVE LINE1 OF A-POEM TO OUT-LINE WRITE OUT-LINE 00153000
  MY-COUNTER ..... 001520 19 010 W 000
MOVE ..... 001530 05 004 W 851 03
  LINE1 ..... 001530 10 005 W 000
    OF ..... 001530 16 002 W 990
      A-POEM ..... 001530 19 006 W 000
        TO ..... 001530 26 002 W 999
          OUT-LINE ..... 001530 29 008 W 000
            WRITE ..... 001530 38 005 W 990 03
MOVE LINE2 OF A-POEM TO OUT-LINE WRITE OUT-LINE 00154000
  OUT-LINE ..... 001530 44 008 W 000
MOVE ..... 001540 05 004 W 851 03
  LINE2 ..... 001540 10 005 W 000
    OF ..... 001540 16 002 W 990
      A-POEM ..... 001540 19 006 W 000
        TO ..... 001540 26 002 W 999
          OUT-LINE ..... 001540 29 008 W 000
            WRITE ..... 001540 38 005 W 990 03
EXAMINE A-POEM TALLYING ALL ".*". 00155000
  OUT-LINE ..... 001540 44 008 W 000

```

```

EXAMINE ..... 001550 05 007 W 990 03
  A-POEM ..... 001550 13 006 W 000
    TALLYING ..... 001550 20 008 W 999
      ALL ..... 001550 29 003 W 990
        "*" ..... 001550 33 003 L 864 00
          . ..... 001550 36 001 000
IF TALLY = MY-COUNTER ..... 00156000
IF ..... 001560 05 002 W 999 03
  TALLY ..... 001560 08 005 W 999
  = ..... 001560 14 001 W 997 00
  MOVE "OK" TO OUT-LINE WRITE OUT-LINE ..... 00157000
    MY-COUNTER ..... 001560 16 010 W 000
  MOVE ..... 001570 09 004 W 851 03
    "OK" ..... 001570 14 004 L 864 00
    TO ..... 001570 19 002 W 999
      OUT-LINE ..... 001570 22 008 W 000
        WRITE ..... 001570 31 005 W 990 03
OTHERWISE ..... 00158000
  OUT-LINE ..... 001570 37 008 W 000
  MOVE "BAH" TO OUT-LINE WRITE OUT-LINE. ..... 00159000
OTHERWISE ..... 001580 05 009 W 990
  MOVE ..... 001590 09 004 W 851 03
    "BAH" ..... 001590 14 005 L 864 00
    TO ..... 001590 20 002 W 999
      OUT-LINE ..... 001590 23 008 W 000
        WRITE ..... 001590 32 005 W 990 03
          OUT-LINE ..... 001590 38 008 W 000
            . ..... 001590 46 001 000
EXAMINE A-POEM TALLYING ALL "E" ..... 00160000
EXAMINE ..... 001600 05 007 W 990 03
  A-POEM ..... 001600 13 006 W 000
    TALLYING ..... 001600 20 008 W 999
      ALL ..... 001600 29 003 W 990
        "E" ..... 001600 33 003 L 864 00
PERFORM THREE-LINES ..... 00161000
PERFORM ..... 001610 05 007 W 990 03
EXAMINE A-POEM TALLYING UNTIL FIRST ". " ..... 00162000
  THREE-LINES ..... 001610 13 011 W 000
EXAMINE ..... 001620 05 007 W 990 03
  A-POEM ..... 001620 13 006 W 000
    TALLYING ..... 001620 20 008 W 999
      UNTIL ..... 001620 29 005 W 999
        FIRST ..... 001620 35 005 W 999
          ". " ..... 001620 41 003 L 864 00
PERFORM THREE-LINES ..... 00163000
PERFORM ..... 001630 05 007 W 990 03
EXAMINE A-POEM TALLYING LEADING "R" ..... 00164000
  THREE-LINES ..... 001630 13 011 W 000
EXAMINE ..... 001640 05 007 W 990 03
  A-POEM ..... 001640 13 006 W 000
    TALLYING ..... 001640 20 008 W 999
      LEADING ..... 001640 29 007 W 999
        "R" ..... 001640 37 003 L 864 00
PERFORM THREE-LINES ..... 00165000
PERFORM ..... 001650 05 007 W 990 03
MOVE 2 TO I ..... 00166000
  THREE-LINES ..... 001650 13 011 W 000
MOVE ..... 001660 05 004 W 851 03
  2 ..... 001660 10 001 N 000
  TO ..... 001660 12 002 W 999
EXAMINE A-NUMBER(I) TALLYING ALL 1 ..... 00167000
  I ..... 001660 15 001 W 000
EXAMINE ..... 001670 05 007 W 990 03
  A-NUMBER ..... 001670 13 008 W 000
    ( ..... 001670 21 001 000
      I ..... 001670 22 001 W 000
    ) ..... 001670 23 001 863 00
    TALLYING ..... 001670 25 008 W 999
      ALL ..... 001670 34 003 W 990
PERFORM THREE-LINES ..... 00168000
  1 ..... 001670 38 001 N 990

```

```

PERFORM ..... 001680 05 007 W 990 03
EXAMINE A-NUMBER(I) TALLYING LEADING 0 REPLACING BY 2. 00169000
THREE-LINES ..... 001680 13 011 W 000
EXAMINE ..... 001690 05 007 W 990 03
A-NUMBER ..... 001690 13 008 W 000
( ..... 001690 21 001 000
I ..... 001690 22 001 W 000
) ..... 001690 23 001 863 00
TALLYING ..... 001690 25 008 W 999
LEADING ..... 001690 34 007 W 999 02
0 ..... 001690 42 001 N 999
REPLACING ..... 001690 44 009 W 999 02
BY ..... 001690 54 002 W 999
2 ..... 001690 57 001 N 000
. .... 001690 58 001 000
THREE-LINES. 00170000
THREE-LINES ..... 001700 01 011 W 860 01
. .... 001700 12 001 000
ADD TALLY TO MY-COUNTER. 00171000
ADD ..... 001710 05 003 W 990 03
TALLY ..... 001710 09 005 W 999
TO ..... 001710 15 002 W 999
MY-COUNTER ..... 001710 18 010 W 000
. .... 001710 28 001 000
MOVE TALLY TO OUT-LINE WRITE OUT-LINE 00172000
MOVE ..... 001720 05 004 W 851 03
TALLY ..... 001720 10 005 W 999
TO ..... 001720 16 002 W 999
OUT-LINE ..... 001720 19 008 W 000
WRITE ..... 001720 28 005 W 990 03
MOVE MY-COUNTER TO OUT-LINE WRITE OUT-LINE. 00173000
OUT-LINE ..... 001720 34 008 W 000
MOVE ..... 001730 05 004 W 851 03
MY-COUNTER ..... 001730 10 010 W 000
TO ..... 001730 21 002 W 999
OUT-LINE ..... 001730 24 008 W 000
WRITE ..... 001730 33 005 W 990 03
OUT-LINE ..... 001730 39 008 W 000
. .... 001730 47 001 000
THE-END. 00174000
THE-END ..... 001740 01 007 W 860 01
. .... 001740 08 001 000
IF TRIPSWCH EQUAL FAILSWCH OR MY-COUNTER NOT EQUAL 125 00175000
IF ..... 001750 05 002 W 999 03
TRIPSWCH ..... 001750 08 008 W 000
EQUAL ..... 001750 17 005 W 991
FAILSWCH ..... 001750 23 008 W 000
OR ..... 001750 32 002 W 999
MY-COUNTER ..... 001750 35 010 W 000
NOT ..... 001750 46 003 W 990
EQUAL ..... 001750 50 005 W 991
WRITE OUT-LINE FROM FAILCON 00176000
125 ..... 001750 56 003 N 000
WRITE ..... 001760 09 005 W 990 03
OUT-LINE ..... 001760 15 008 W 000
FROM ..... 001760 24 004 W 999
OTHERWISE 00177000
FAILCON ..... 001760 29 007 W 000
WRITE OUT-LINE FROM SUCCESS. 00178000
OTHERWISE ..... 001770 05 009 W 990
WRITE ..... 001780 09 005 W 990 03
OUT-LINE ..... 001780 15 008 W 000
FROM ..... 001780 24 004 W 999
SUCCESS ..... 001780 29 007 W 000
. .... 001780 36 001 000
CLOSE PRINT-FILE. 00179000
CLOSE ..... 001790 05 005 W 990 03
PRINT-FILE ..... 001790 11 010 W 000
. .... 001790 21 001 000
STOP RUN. 00180000
STOP ..... 001800 05 004 W 990 03

RUN ..... 001800 10 003 W 999
. .... 001800 13 001 000

```

LCP debugging

This section is Diagnosis, Modification, and Tuning Information.

To help you debug LCPs, CCCA can generate trace output for:

- All LCPs, using the **Generate tokenization listing** option on Conversion Options panel 1 (see “Setting conversion options” on page 19)
- Specific LCPs, using the Delete/Debug LCPs panel (see “Deleting LCPs and activating/deactivating debugging for LCPs” on page 71)

The following pages show example trace output generated by the OTHERWISE LCP and EXAMINE LCP.

This output should be used in conjunction with the LCP Compiler output.

The columns of the trace output are described below.

***CONVER**

The identifier in the LCP's *CONVER statement.

***DATE**

The date the LCP was last compiled (in the format MMDDYY).

TOKEN-TEXT

Indicates for each statement the current token or element.

LCP STMT

Number of statement given by the compiler.

LCP OPCODE

Operation code. See Appendix F, “List of LCP functions,” on page 187 for a list of LCP functions and their operation codes.

ID FILE

File used by the LCP:

TOKEN
CHANGE
WORK-*mm*
RECORD
FILE
KEY

These files are described in “LCP functions” on page 91 and “Manipulating files” on page 101.

RT Return code after reading or writing the files.

RV Internal use.

TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... 1	... 2	... 3	... 4	... 5	... 6	... 7	-CODE- RT RV
OTHERWISE	3	IFEQA									
OTHERWISE	4	IFEQA									
OTHERWISE	6	IFEQA									
OTHERWISE	8	MOVE									
OTHERWISE	9	RP									
			CHANGE	001150056	04ELSE			01	Y		
			CHANGE	001150055					Y		
OTHERWISE	10	MOVE									
OTHERWISE	11	EDMSG									
			CHANGE	001150053	00OTHERWISE REPLACED BY ELSE					YABJ602100	
OTHERWISE	12	GOTO									

TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... 1	... 2	... 3	... 4	... 5	... 6	... 7	-CODE- RT RV
EXAMINE	10	IFEQA									
EXAMINE	11	IFEQA									
EXAMINE	13	IFEQA									
EXAMINE	15	MOVE									
EXAMINE	16	MOVE									
EXAMINE	17	MOVE									
EXAMINE	18	GTPRT									
.	19	SPLN	TOKEN	00138014001	000 00.						NP
.			CHANGE	001380148							Y
.	20	MOVE									
.	21	GTNXT	TOKEN	00139005007W990	03EXAMINE						YP
EXAMINE	21	GTNXT	TOKEN	00139013006W000	00A-POEM						YP
A-POEM	22	BYID	TOKEN	00139020008W000	00TALLYING						YP
TALLYING	23	IFEQA									
TALLYING	24	MOVE									
TALLYING	25	GTPRT									
.	26	MOVE	TOKEN	00138014001	000 00.						NP
.	27	SF									
.	28	MOVE	CHANGE	001380148	18MOVE ZERO TO TALLY						0005N
.	29	EDMSG	CHANGE	001380143	00TALLY IS INITIALIZED						ABJ601800
.	30	MOVE									
.	31	MOVE									
.	32	MOVE									
.	33	MOVE									
.	34	MOVE									
.	35	MVLCP									
.	36	MOVE									
.	37	GOTO									
.	40	GTNXT	TOKEN	00139005007W990	03EXAMINE						YP
EXAMINE	41	MOVE									
EXAMINE	42	RP									
			CHANGE	001390056	07INSPECT			01	Y		
			CHANGE	001390055					Y		
EXAMINE	43	MOVE									
EXAMINE	44	EDMSG	CHANGE	001390053	00EXAMINE REPLACED BY INSPECT						YABJ601900
EXAMINE	45	GTNXT	TOKEN	00139013006W000	00A-POEM						YP
A-POEM	46	BYID	TOKEN	00139020008W000	00TALLYING						YP
TALLYING	47	IFEQA									
TALLYING	48	PRFTH									
TALLYING	53	MOVE									
TALLYING	54	INAF									

TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... .. 1 2 3 4 5 6 7	-CODE- RT RV
			CHANGE 001390208	05TALLY	01 Y
TALLYING	55	MOVE			
TALLYING	56	INAF			
			CHANGE 001390208	03FOR	01 Y
TALLYING	57	GTNXT			
			TOKEN 00139029003W990	ALL	YP
ALL	58	MOVE			
ALL	59	MOVE			
ALL	60	IFEQA			
ALL	66	GTNXT			
			TOKEN 00139033006W000	00SPACES	YP
SPACES	67	PRFTH			
SPACES	123	IFEQA			
SPACES	124	IFEQA			
SPACES	125	IFEQA			
SPACES	135	GOTO			
SPACES	154	EXIT			
SPACES	68	MOVE			
SPACES	69	MOVE			
SPACES	70	GTNXT			
			TOKEN 00139040009W999	02REPLACING	YP
REPLACING	71	IFEQA			
REPLACING	73	IFEQA			
REPLACING	83	IFEQA			
REPLACING	84	MOVE			
REPLACING	85	MOVE			
REPLACING	86	INAF			
			CHANGE 001390408	03ALL	01 Y
REPLACING	87	MOVE			
REPLACING	88	MOVE			
REPLACING	89	INAF			
			CHANGE 001390408	06SPACES	01 Y
REPLACING	90	GTNXT			
			TOKEN 00139050002W000	00BY	YP
BY	90	GTNXT			
			TOKEN 00139053003L000	00"*"	YP
"*"	91	PRFTH			
"*"	123	IFEQA			
"*"	135	GOTO			
"*"	154	EXIT			
"*"	93	EXIT			
"*"	49	GOTO			
"*"	51	GOTO			

TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... .. 1 2 3 4 5 6 7	-CODE- RT RV
EXAMINE	10	IFEQA			
EXAMINE	11	IFEQA			
EXAMINE	13	IFEQA			
EXAMINE	15	MOVE			
EXAMINE	16	MOVE			
EXAMINE	17	MOVE			
EXAMINE	18	GTPRT			
OUT-LINE	19	SPLN	TOKEN 00142044008W000	00OUT-LINE	YP
OUT-LINE	20	MOVE	CHANGE 001420448		Y
OUT-LINE	21	GTNXT	TOKEN 00143005007W990	03EXAMINE	YP
EXAMINE	21	GTNXT	TOKEN 00143013006W000	00A-POEM	YP
A-POEM	22	BYID	TOKEN 00143020008W000	00TALLYING	YP
TALLYING	23	IFEQA			
TALLYING	24	MOVE			
TALLYING	25	GTPRT			
OUT-LINE	26	MOVE	TOKEN 00142044008W000	00OUT-LINE	YP
OUT-LINE	27	SF	CHANGE 001420448	18MOVE ZERO TO TALLY	0005Y
OUT-LINE	28	MOVE	CHANGE 001420443	00TALLY IS INITIALIZED	ABJ601800
OUT-LINE	29	EDMSG			
OUT-LINE	30	MOVE			
OUT-LINE	31	MOVE			
OUT-LINE	32	MOVE			
OUT-LINE	33	MOVE			
OUT-LINE	34	MOVE			
OUT-LINE	35	MVLCF			
OUT-LINE	36	MOVE			
OUT-LINE	37	GOTO			
OUT-LINE	40	GTNXT	TOKEN 00143005007W990	03EXAMINE	YP
EXAMINE	41	MOVE			
EXAMINE	42	RP	CHANGE 001430056	07INSPECT	01 Y
			CHANGE 001430055		Y
EXAMINE	43	MOVE	CHANGE 001430053	00EXAMINE REPLACED BY INSPECT	YABJ601900
EXAMINE	44	EDMSG			
EXAMINE	45	GTNXT	TOKEN 00143013006W000	00A-POEM	YP
A-POEM	46	BYID	TOKEN 00143020008W000	00TALLYING	YP
TALLYING	47	IFEQA			
TALLYING	48	PRFTH			
TALLYING	53	MOVE			
TALLYING	54	INAF			

5648-B05 V2R1
*CONVER:EXAMINE
100540

- IBM COBOL CONVERSION AID -
*TEXT:CHANGE EXAMINE BY INSPECT

SAMPLE RUN

17 APR 1998 19:10:10
*DATE:041598

PAGE 6

TOKEN-TEXT	LCP STMT	LCP OPCODE	ID FILE	*... .. 1 2 3 4 5 6 7	-CODE- RT RV
TALLYING	55	MOVE	CHANGE 001430208	05TALLY	01 Y
TALLYING	56	INAF	CHANGE 001430208	03FOR	01 Y
TALLYING	57	GTNXT	TOKEN 00143029003W990	ALL	YP
ALL	58	MOVE			
ALL	59	MOVE			
ALL	60	IFEQA			
ALL	66	GTNXT	TOKEN 00143033003L000	00"*"	YP
"*"	67	PRFTH			
"*"	123	IFEQA			
"*"	135	GOTO			
"*"	154	EXIT			
"*"	68	MOVE			
"*"	69	MOVE			
"*"	70	GTNXT	TOKEN 00143036001	000 00.	NP
.	71	IFEQA			
.	72	GOTO			
.	93	EXIT			
.	49	GOTO			
.	51	GOTO			

Appendix I. Maintaining CCCA under MVS

This chapter describes how to re-install CCCA and how to apply service updates to CCCA. To use the maintenance procedures effectively, you should have already installed CCCA and any required products.

In addition, this chapter describes how to remove CCCA.

Re-installing CCCA

The action required here depends on the circumstance. If you want to re-install and you did not use the SMP/E ACCEPT command then use a SMP/E APPLY REDO command. However, if you did use the SMP/E ACCEPT command, then the product should be deleted before installing again. For more information refer to "Removing CCCA" on page 258.

Applying service updates

You might need to apply maintenance or service updates to CCCA periodically.

What you receive

If you report a problem with CCCA to your IBM Support Center, you will receive a tape containing one or more APARs or PTFs that have been created to solve your problem.

You might also receive a list of prerequisite APARs or PTFs, which should have been applied to your system before applying the current service. These prerequisite APARs or PTFs, might relate to CCCA or any other licensed product you have installed, including MVS.

To help you understand the service process, the following overview familiarizes you with applying service for CCCA.

Checklist for applying service

Table 8 lists the steps and associated SMP/E commands for installing corrective service on CCCA. You can use Table 8 as a checklist.

Table 8. Summary of steps for installing service on CCCA

Step	Description	SMP/E Command
__ 1	Prepare to install service.	
__ 2	Receive service.	RECEIVE
__ 3	Accept previously applied service. (optional)	ACCEPT
__ 4	Apply service.	APPLY
__ 5	Test service.	
__ 6	Accept service.	ACCEPT

Step 1. Prepare to install service

Before you start applying service:

1. Create a backup copy of the current CCCA. Save this copy of CCCA until you have completed installing the service and you are confident that the service runs correctly.
2. Research each service tape through the IBM Support Center for any errors and/or additional information. Note all errors on the tape that were reported by APARs and apply the applicable fixes.

Step 2. Receive the service

Receive the service using SMP/E RECEIVE command. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Step 3. Accept applied service (optional)

Accept any service you applied earlier but did not accept, if you are satisfied that the earlier service is not causing problems in your installation. This can be done from the SMP/E dialogs in ISPF or using a batch job. Accepting the earlier service allows you to use the SMP/E RESTORE command to return to your current level if you encounter a problem with the service you are currently applying. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Step 4. Apply the service

Apply the service using SMP/E APPLY command. You should use the SMP/E APPLY command with the CHECK operand first. Check the output; if it shows no conflict, rerun the APPLY without the CHECK option. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Step 5. Test the service

Thoroughly test your updated CCCA. Do not accept a service update until you are confident that it runs correctly.

In the event of a serious problem, you can restore the backup copy of CCCA.

Step 6. Accept the service

Accept the service using SMP/E ACCEPT command. You should use the SMP/E ACCEPT command with the CHECK operand first. Check the output; if it shows no conflict, rerun the ACCEPT without the CHECK option. This can be done from the SMP/E dialogs in ISPF or using a batch job.

Removing CCCA

To delete CCCA, you must:

- Make sure no other products depend on it.
- Use a dummy function SYSMOD to delete it.
- Receive, apply and accept the dummy function, and run the UCLIN to delete the SYSMOD entries for the deleted function and the dummy function.

Edit and submit job ABJDEL0 to delete CCCA. Consult the instructions in the sample job for more information.

Expected Return Codes and Messages: You receive message GIM39701W because the dummy function SYSMOD has no elements. The SMP/E RECEIVE command returns a return code of 4. If any USERMODs have been applied then the SMP/E

APPLY command issues a GIM44502W message indicating USERMOD changes will be lost with a return code of 4. Both these warning messages can be ignored.

The target and distribution libraries can now be deleted.

Reporting a problem with CCCA

Report any difficulties with this product to your IBM Support Center. In the United States, if an APAR is required, submit the data to the location identified in the *Field Engineering Programming System General Information* manual (PSGIM), G229-2228, as being responsible for the failing component.

Table 9 identifies the component ID (COMP ID) for CCCA.

Table 9. Component IDs

FMID	COMP ID	Component Name	REL
H09F210	5648B0500	COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM	210

Obtaining service information

Preventive Service Planning (PSP) information is continually updated as fixes are made available for problems. Check with your IBM Support Center or use either Information/Access or SoftwareXcel Extended to see whether there is additional PSP information you need. To obtain this information, specify the following UPGRADE and SUBSET values: CCCA210 and H09F210.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling (1) the exchange of information between independently created programs and other programs (including this one) and (2) the mutual use of the information that has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA
95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Notices

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Programming interface information

This book primarily documents information that is NOT intended to be used as Programming Interfaces of COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM.

This book also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of COBOL and CICS Command Level Conversion Aid for OS/390 & MVS & VM. This information is identified where it occurs by an introductory statement to a chapter or section.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Bibliography

IBM COBOL for OS/390 & VM

Compiler and Run-Time Migration Guide, GC26-4764

Programming Guide, SC26-9049

Language Reference, SC26-9046

Related Publications for OS/390 & VM

- **Language Environment[®]**

Programming Guide, SC28-1939

Programming Reference, SC28-1940

IBM COBOL for MVS & VM

Compiler and Run-Time Migration Guide, GC26-4764

Programming Guide, SC26-4767

Language Reference, SC26-4769

Related Publications for MVS & VM

- **Language Environment**

Programming Guide, SC26-4818

Programming Reference, SC26-3312

Enterprise COBOL for z/OS & OS/390

Migration Guide, GC27-1409

Language Reference, SC27-1408

Programming Guide, SC27-1412

Other publications

VM/ESA

Application Development Guide, SC24-5450

Application Development Reference, SC24-5451

Command Reference, SC24-5461

User's Guide, SC24-5460

CICS/ESA

Application Programming Guide, SC33-1169

Application Programming Reference, SC33-1170

Softcopy Publications for OS/390, MVS, and VM

The following collection kits contain IBM COBOL or related product publications.

MVS Collection, SK2T-0710

OS/390 Collection, SK2T-6700

VM Collection, SK2T-2067

Index

A

ABEND status 58
ACCEPT MESSAGE COUNT
 conversion 117
ACTUAL KEY conversion 118
ADD (LCP statement) 84
ADD conversion 142
Add DATE FORMAT clause to all date fields 22
ALPHABET clause conversion 118
ALPHABETIC class conversion 118
APAR 257
apostrophe
 enclosing character in LCP 81
APPLY clause conversion
 CORE-INDEX 118
 CYL-INDEX 118
 CYL-OVERFLOW 118
 EXTENDED-SEARCH 118
 MASTER-INDEX 118
 RECORD-OVERFLOW 118
 REORG-CRITERIA 118
 WRITE-VERIFY 118
applying maintenance 257
applying service checklist 257
applying service updates 257
arithmetic in CICS BLL cells, flagging 22
ASSIGN
 clause conversion 119
 integer conversion 119
ASSIGN...OR conversion 119
AUTHOR conversion 119

B

backup of CCCA (MVS) 258
batch mode conversion, MVS 27
BDAM conversion 119
before you convert 8
blank lines in an LCP 81
BLANK WHEN ZERO conversion 120
BLL (Base Locator for Linkage)
 cells in linkage section 29
 conversion description 4
 conversion method 22
 flagging arithmetic in 22
BLOCK CONTAINS conversion 120
bypassing
 token identifiers 93
 token processing 100

C

CALL statement
 conversion 120
 generate abend CALL statements 24
CALL statements in converted programs 55
call/program report 55
CALL...USING statements, flagging 23

change code 67
character set, LCP 78
check procedure names 22
CICS
 conversion description 4
 conversion option 29, 33
 conversion, sample COBOL program 222
 record, manipulating 106
 statements converted 141
class, CCCA job
 conversion job 9
CLOSE FOR REMOVAL conversion 121
CLOSE...WITH DISP/POSITIONING conversion 121
CMPR2 17, 117
COBOL
 DOS/VS COBOL 17, 117
 Enterprise COBOL 18
 IBM COBOL 18
 language elements converted 117
 OS/VS COBOL 17, 117
 programs, converting 17
 sample conversion
 program 211
 with CICS commands 222
 with COPY 217
 VS COBOL II 17, 18, 117
COBOL 68 Standard, definition 5
COBOL 74 Standard, definition 5
COBOL 85 Standard, definition 5
COBOL Conversion Aid message panel 74
COBOL conversion Job Statement Information panel 27
COBOL conversion Selection 28
COBOL conversion Submission panel 31
COBOL reserved word panel 66
COBOL standards 117
COBOL Standards 17, 23
COBOL/370
 reserved word table
 updating 66
COBOL/VSE
 language elements converted to 117
columns, LCP source line 78
COM-REG conversion 121
comment lines, LCP 80
comment paragraphs, list of 63
comments about the conversion 57
communication module conversion 121
compile after converting 22
 return code 24, 52
compiler, LCP
 error messages 148
 predefined data items 175
 reserved words 167
COMPLETE status 58
component id 259
component name 259
COMPUTE conversion 142

conditions, LCP 84
CONFIGURATION SECTION header
 conversion 122
confirm erase log panel 58
constructing tokens 96
control file 49, 57
controlling LCP invocation 106
CONVER (LCP statement) 81
conversion
 batch mode, MVS 27
 BLL cells in linkage section 29
 CICS commands 141, 191
 CICS description 4
 COBOL language elements 117
 converting COBOL programs 17
 date and time program was last converted 50
 debug output 19
 DLI 29
 error messages 151
 EXEC CICS commands 29
 file organization 52
 LCP debug sample output 251
 output, specifying 19
 phases, batch job 4
 reducing conversion time 23
 return code 37
 revision number 50
 sample
 COBOL program 211
 with CICS commands 222
 with COPY 217
 SQL 29
 VM 32
conversion log
 See log, conversion
Conversion Member List panel 28
Conversion Member Selection panel 32
conversion method, BLL cells 22
Conversion Selection panel 32
converted CICS commands 191
converted COBOL statements 191
converter error messages 145
converter menu 12
converting reserved words 133
copy books
 See copy members
COPY conversion 122
copy members
 generate new 19
 print in diagnostic listing 19
 replace like-named 19
 used by converted programs 54
copy/program report 54
COPY...REPLACING conversion 123
CURRENCY SIGN conversion 123
CURRENT-DATE conversion 20, 123
customizing CCCA
 how a language element is converted 64

customizing CCCA (*continued*)
to convert an additional language
element 64

D

data division, LCP 83
data item identifier, LCP 79, 83
data name, date identification file 42
DATA RECORDS conversion 124
date and time
manual conversion completed 57
program was last converted 50
DATE COMPILED conversion 124
DATE FORMAT clause
checking syntax 46
description 40
including 24
on Conversion Options panel 2 22
DATE FORMAT conversion option
description 39
how it works 46
overview 2
selecting 24, 46
date format, date identification file 42
date format, VSE system 19
date identification file
creation of 42
date identification file 42
description 42
examples 45
format 42
data name 42
date format 42
line number 42
program name 42
supplying to CCCA 42
DATE WRITTEN conversion 124
DATE-COMPILED conversion 124
DATE-WRITTEN conversion 124
debugging
activating/deactivating for a
particular LCP 71
diagnostic listing output 19
tracing executed statements
of a particular LCP 71
of all LCPs 19
declaratives conversion 124
default code, COBOL reserved 67
delete product
MVS 258
deleting LCPs 71
delimiter, COBOL
in Program/File report 50
search source for 22
specifying 22
diagnostic listing
*OLD** 21
debug output 19
heading 19
print copy members 19
print diagnostics of level >= 19
print old source lines 19
reading 35
tailoring 35
directory of the LCP library 72
DISABLE conversion 124

DIVIDE...ON SIZE ERROR 124
divisions, LCP 78
DLI option 29
DOS/VS COBOL 17, 117
Double-Byte Character Set, in date
identification file 42

E

editing tokens 96
element
definition 91
difference from token 111
processing 111
removing 94
retrieving 92
ENABLE conversion 124
enclosing character, LCP 81
END PROGRAM header, generate 22
END-EXEC conversion 143
ENTER statement conversion 124
environment options 8
erasing the conversion log 58
ERROR DECLARATIVES GIVING
conversion 124
error messages
converter 145
diagnostics from LCPs 151
LCP compiler 148
tokenization diagnostics 150
EXAMINE conversion 124
EXEC CICS command 29
EXHIBIT conversion 124
EXIT (LCP statement) 86
EXIT PROGRAM statement
conversion 125

F

file manipulation, LCP 101
FILE STATUS conversion 125
FILE-LIMIT(S) conversion 125
file/program report 53
files
CCCA messages 73
conversion required? 52
organization before/required after
conversion 52
used by a program 50
flagged COBOL statements 195
flagging
BLL cell arithmetic 22
CALL...USING statements 23
IF FILE-STATUS (NOT) = "00" 22
Report Writer statements 22
turning off to reduce conversion
time 23
USE FOR DEBUGGING
declaratives 23
fmid 259
FOR MULTIPLE REEL/UNIT clause
conversion 127
foreground conversion, VM 32
format notation, description x
function keys 10
function modification ID 259

functions, LCP
list of 187
using 91

G

generate
END PROGRAM header 22
new copy members 19
new program 19
GET- LCP functions 92
GO TO (LCP statement) 86
GREATER THEN conversion 125

H

header, generate END PROGRAM 22
heading, report 19

I

IBM Support Center
APARs and PTFs 257
UPGRADE value and SUBSET ID,
MVS 259
identifier, LCP data item 83
IF (LCP statement) 86
IF conversion 126
IF FILE-STATUS (NOT) = "00" statements,
flag 22
increment, sequence number 19
indexes (qualified) 126
industry standards 5
Information/Access 259
INITIALIZE conversion 126
inserting tokens 94
INSTALLATION conversion 126
installation verification, MVS 258
installation, MVS
installation verification program (IVP)
service installation 258
ISAM conversion 126

J

job cards, updating 27, 68
JUST RIGHT conversion 127
JUSTIFIED conversion 127

L

LABEL RECORDS clause
conversion 127
LANGLVL 17, 117
language level panel 17
LCP Development Aid
message panel 74
reserved word table updating 66
LCP development aid menu 14
LCP functions 91
LCP-*nnn* identifier 82
LCPs
activating/deactivating
debugging 71
blank lines 81

LCPs (*continued*)

- character set 78
- comment lines 80
- controlling invocation 106
- data item identifier 79, 83
- deleting 71
- description 63, 77
- developing 77
- directory (sample output) 201
- directory of the LCP library 72, 191
- divisions 78
- functions 91, 187
- language structure 78
- literals
 - nonnumeric 80
 - numeric 80
- messages 150, 151
- modifying 77
- number invoked during program
 - conversion 50
- paragraph name 79, 88
- predefined data items 175
- processing of 107
- punctuation 80
- reserved words 79, 167
- source line columns 78
- source member name 82
- statement summary 81
- statements
 - ADD 84
 - CONVER 81
 - EXIT 86
 - GO TO 86
 - IF 86
 - MOVE 88
 - PERFORM 89
 - SUBTRACT 90
- using periods 80
- LESS THEN conversion 127
- line number, date identification file 42
- LINE(S) conversion 139
- lines per report page 19
- linkage section, BLL cells in 29
- literal delimiter, COBOL
 - in Program/File report 50
 - search source for 22
 - specifying 22
- literals - nonnumeric conversion 127
- log, conversion 57
 - browsing 57
 - erasing 58
 - updating 57
- logical files, manipulating 101

M

- maintenance 257
- MAN. COMP status 58
- manual changes, flag 22
- manual conversion
 - comments 57
 - statistics, browsing and updating 57
- manuals, useful for CCCA 263
- master menu 12
- member name, LCP source 82
- MEMORY SIZE conversion 127

- menus
 - converter 12
 - LCP development aid 14
 - map of 10
 - master 12
 - navigating 10
 - options 15
- message file, viewing and updating 73
- messages
 - converter 145
 - diagnostics from LCPs 151
 - tokenization diagnostics 150
- millennium language extensions
 - See MLE
- MLE
 - description 39
- MM/DD/YY vs. DD/MM/YY (DOS/VS COBOL only) 20
- modifying tokens 94
- MOVE (LCP statement) 88
- MOVE ALL literal conversion 127
- MOVE conversion 127, 143
- MOVE CORRESPONDING
 - conversion 127
- MULTIPLE FILE/TAPE conversion 128
- MULTIPLY...ON SIZE ERROR 128
- MVS
 - batch mode conversion 27

N

- navigating menus and panels 10
- NOCHANGE status 58
- NOCMPR2 18, 117
- NOMINAL KEY conversion 128
- nonnumeric literals, LCP 80
- NOT conversion 128
- notation, description x
- NOTE conversion 128
- NSTD-REELS conversion 128
- numeric literals, LCP 80

O

- obsolete elements, remove 22
- OCCURS clause 128
- OCCURS DEPENDING ON clause
 - conversion 129
- OPEN...DISP conversion 130
- OPEN...LEAVE conversion 130
- OPEN...REREAD conversion 130
- OPEN...REVERSED conversion 131
- options, CCCA
 - BLL cell conversion method 22
 - BLL cells in linkage section 29
 - check procedure names 22
 - conversion 19
 - conversion debug output 19
 - DLI 29
 - environment 8
 - EXEC CICS commands 29
 - flag BLL cell arithmetic 22
 - flag IF FILE-STATUS (NOT) = "00" 22
 - flag Report Writer statements 22
 - generate END PROGRAM header 22

- options, CCCA (*continued*)
 - generate new copy members 19
 - generate new program 19
 - language level 17, 29, 33
 - lines per report page 19
 - literal delimiter (QUOTE or APOST) 22
 - menu 15
 - MLE conversion 22
 - Negate implicit EXIT PROGRAM 22
 - print copy members 19
 - print diagnostics of level >= 19
 - print old source lines 19
 - remove obsolete elements 22
 - remove VALUE clauses 22
 - replace like-named copy members 19
 - report heading 19
 - reporting options used by converted programs 50
 - resequence source lines 19
 - reserved word suffix 19
 - search source for literal delimiter 22
 - sequence number increment 19
 - SQL 29
 - VSE system date format 19
- ORGANIZATION clause conversion 131
- organization of files before/required after conversion 52
- OS/VS COBOL 17, 117
- OTHERWISE conversion 131
- output class, non default 27, 68
- output, sample
 - call/program report 201
 - compilation of an LCP 205
 - conversion
 - COBOL program 211
 - with CICS commands 222
 - with COPY 217
 - copy/program report 200
 - file/program report 200
 - LCP debugging 251
 - LCP directory 201
 - program/file report 199
 - tokenization 238

P

- panels
 - COBOL reserved word panel 66
 - confirm erase log 58
 - conversion options 1 19
 - conversion options 2 22
 - converter menu 12, 13
 - environment options 8
 - language level 17
 - LCP development aid menu 14
 - map of 10
 - master menu 12
 - navigating 10
 - options 15
- panels, Conversion Aid
 - batch LCP compilation (Job Statement Information) 68
 - batch LCP compilation (Selection) 69
 - batch LCP compilation (Submission) 69

- panels, Conversion Aid (*continued*)
 - COBOL conversion Job Statement Information 27
 - COBOL conversion Selection 28
 - COBOL conversion Submission 31
 - LCP Compiler Selection (VM) 70
 - LCP selection 69
- panels, Member list
 - COBOL conversion 28
- panels, Member selection, VM
 - COBOL conversion 32
- panels, VM Conversion
 - Conversion selection 32
- paragraph names, LCP 88
- PERFORM (LCP statement) 89
- PERFORM/ALTER conversion 131
- performance
 - improving by turning off flagging 23
- period insertion 131
- periods, using in LCPs 80
- PF keys 10
- phases, batch conversion job 4
- physical files
 - description 101
- PIC/PICTURE flagging 131
- picture P in RELATIVE KEY
 - conversion 133
- post-conversion compile
 - See* compile after converting
- predefined data items
 - description 91
 - list of 175
- prerequisites 257
- preventive service plan
 - UPGRADE value and SUBSET ID, MVS 259
- print copy members 19
- print diagnostics of level >= 19
- print old source lines 19
- problem
 - reporting, MVS 259
- procedure division, LCP 84
- procedure names, check 22
- PROCESS clause conversion 132
- PROCESSING MODE conversion 132
- program function keys 10
- program name conversion 132
- program name, date identification
 - file 42
- PROGRAM-ID conversion 132
- program/call report 56
- program/copy report 54
- program/file report 50
- PSP 259
- PTF
 - service tape, MVS 257
- punctuation, LCP 80

Q

- quotation mark
 - enclosing character in LCP 81

R

- railroad track format, how to read x
- re-installing, MVS 257
- READ statement conversion 132
- READY TRACE conversion 132
- RECEIVE conversion 132
- RECORD CONTAINS conversion 132
- RECORDING MODE clause
 - conversion 132
- records
 - description 101
 - fixed-use, manipulating
 - CALL 104
 - COPY 104
 - FILE 103
 - KEY 105
 - OPTION 102
 - PROGRAM 103
 - RECORD 105
 - variable-use, manipulating
 - CICS 106
 - WORK-nn 105
- REDEFINES clause in FD
 - conversion 133
- REDEFINES clause in SD
 - conversion 133
- REMARKS paragraph conversion 133
- remove
 - elements 94
 - obsolete elements 22
 - tokens 94
- remove VALUE clauses in File/Linkage
 - sections 22
- REPLACE conversion 133
- replace like-named copy members 19
- report heading 19
- REPORT WRITER statements
 - conversion 133
- Report Writer statements, flag 22
- reports
 - call/program 55
 - copy/program 54
 - file/program 53
 - generating 49
 - heading 19
 - LCP directory 72
 - lines per page 19
 - program/call 56
 - program/copy 54
 - program/file 50
 - sample output 199
- required words x
- resequence source lines 19
- RESERVE AREAS conversion 133
- reserved word file, COBOL 66
 - change code 67
 - default code 67
 - word type 67
- reserved word suffix 19
- reserved words
 - table, description 66
- reserved words, LCP
 - description 79
 - list of 167
 - used by the LCP compiler 167
- RESERVES ALTERNATES AREAS
 - conversion 133

- RESET TRACE conversion 133
- RETAIN release 259
- retrieving elements 92
- retrieving tokens 92
- return code
 - compile after conversion 24, 52
 - program conversion 37
- revision number, converted program 50
- REWRITE statement conversion 134

S

- SAME AREA clause conversion 134
- SEARCH...WHEN conversion 134
- SECURITY conversion 134
- SEEK conversion 134
- SELECT OPTIONAL conversion 134
- SEND conversion 134
- sequence number increment 19
- service
 - checklist, MVS 257
 - process for MVS 257
- SERVICE RELOAD conversion 143
- SET...TO TRUE 134
- setting conversion option for 29
- severity level 74
 - filtering from diagnostic listing 19
- signed VALUE conversion 138
- SMP/E
 - ACCEPT 258
 - APPLY 258
 - APPLY CHECK 258
 - APPLY REDO 257
 - RECEIVE 258
- SoftwareXcel Extended 259
- source language level 17, 29, 33
- SQL option 29
- START...USING KEY conversion 135
- statement summary, LCP 81
- statistics
 - CALL statements in converted
 - programs 55
 - CICS option 51
 - converted program status 50, 58
 - copy members used by converted
 - programs 54
 - date and time a program was last
 - converted 50
 - deleting 58
 - files used by converted program 50
 - language level 50
 - literal delimiter used in COBOL
 - program 50
 - number of LCPs invoked during
 - program conversion 50
 - number of user-defined words
 - appended by suffixes 50
 - options used by converted
 - program 50
 - revision number of converted
 - program 50
- status
 - converted program 50, 58
- STRING conversion 135
- structure, LCP language 78
- Subset ID 259
- SUBTRACT (LCP statement) 90

SUBTRACT conversion 143
suffix, reserved word 19
suffixes, number of user-defined words
 appended by 50
syntax notation, description x
system date format, VSE 19

T

target language level 17
THEN conversion 135
time and date a program was last
 converted 50
TIME-OF-DAY conversion 136
token
 difference from element 111
token identifiers, bypassing 93
token processing, bypassing 100
TOKEN-POINTER, moving 93
tokenization 108
 error messages 150
 sample output 238
tokenized source, manipulating using
 LCPs 91
tokens
 constructing 96
 editing 96
 inserting 94
 modifying 94
 removing 94
 retrieving 92
TOTALING/TOTALED AREA
 conversion 136
TRACE conversion 136
tracing executed LCP statements
 of all LCPs 19
 of specific LCPs 71
TRACK-AREA conversion 136
TRACK-LIMIT conversion 136
TRANSFORM statement conversion 137

U

UNSTRING 137
Upgrade
 value, MVS 259
UPSI name conversion 138
USE AFTER LABEL PROCEDURE
 conversion 138
USE AFTER STANDARD
 conversion 138
USE BEFORE STANDARD
 conversion 138
USE FOR DEBUGGING conversion 138
USE FOR DEBUGGING declaratives,
 flagging 23
USE procedures (precedence of) 138
user-defined words appended by suffixes,
 number of 50
using CCCA as supplied 12

V

VALUE clause 22
VALUE in 88 level conversion 138
VALUE OF conversion 139

VALUES conversion 138
verification
 after service 258
VM
 conversion 32
VS COBOL II 18
 converted COBOL language
 elements 117
 language elements converted to 117
 reserved word table
 updating 66
 setting source language level 17
VSE system date format 19

W

WARNING status 58
what you receive
 service 257
WHEN-COMPILED conversion 139
WHEN-COMPILED date, converting 20
word type, COBOL reserved 67
WORK-nn records 105
WRITE ... AFTER POSITIONING
 conversion 139
WRITE statement conversion 139



Printed in USA

SC26-9400-03

