

CA-Culprit™

User Guide
15.0



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, December 2000

© 2000 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.

Contents

How to use this Manual	xiii
-------------------------------	------

The Basics

Chapter 1. Getting Started with CA-Culprit	1-1
1.1 What is CA-Culprit and what can you do with it?	1-3
1.2 How to produce a CA-Culprit report	1-4
1.2.1 Step 1. Planning the report	1-4
1.2.2 Step 2. Writing the code	1-5
1.2.3 Step 3. Executing the code	1-5
1.2.4 Step 4. Debugging the code	1-5
1.3 What you should know before you begin	1-7
1.3.1 How information is organized in a computer	1-7
1.3.2 What a file description tells you	1-9
1.3.3 How to use information in a file description	1-9
Chapter 2. Generating a Basic Report from Standard Files	2-1
2.1 Locating the data	2-4
2.2 Coding the report	2-5
2.2.1 How to define the input	2-5
2.2.2 How to define the output	2-6
2.2.3 Demonstration	2-7
Chapter 3. Enhancing the Basic Report	3-1
3.1 Adding a title	3-4
3.1.1 What you can do	3-4
3.1.2 How to do it	3-4
3.1.3 Demonstration	3-4
3.2 Adding column headings	3-6
3.2.1 What you can do	3-6
3.2.2 How to do it	3-6
3.2.3 Demonstration (1): Formatting multiple Line column headings	3-7
3.2.4 Demonstration (2): Adding subtitles	3-8
3.3 Modifying column widths	3-11
3.3.1 What you can do	3-11
3.3.2 How to do it	3-11
3.3.3 Demonstration	3-11
3.4 Formatting numeric data	3-13
3.4.1 What you can do	3-13
3.4.2 How to do it	3-13
3.4.3 Demonstration (1): Using formatting codes	3-13
3.4.4 Demonstration (2): Using formatting patterns	3-14
3.5 Adjusting line length and page depth	3-16
3.5.1 What you can do	3-16
3.5.2 How to do it	3-16
3.5.3 Demonstration	3-16

Chapter 4. Additional Standard Options	4-1
4.1 Multiple reports	4-4
4.1.1 What you can do	4-4
4.1.2 How to do it	4-4
4.1.3 Demonstration	4-4
4.2 Sequencing report output	4-8
4.2.1 What you can do	4-8
4.2.2 How to do it	4-8
4.2.3 Demonstration	4-8
4.3 Selective processing	4-10
4.3.1 What you can do	4-10
4.3.2 How to do it	4-10
4.3.3 Demonstration (1): Selecting records with single criteria	4-10
4.3.4 Demonstration (2): Selecting records with multiple criteria	4-11
4.4 Multiple detail lines	4-13
4.4.1 What you can do	4-13
4.4.2 How to do it	4-13
4.4.3 Demonstration	4-13
Chapter 5. Adding Programming Logic to a Report	5-1
5.1 How to code programming logic	5-4
5.2 Using counters	5-5
5.2.1 What you can do	5-5
5.2.2 How to do it	5-5
5.2.3 Demonstration (1): Printing the count of each occurrence	5-6
5.2.4 Demonstration (2): Printing a total count	5-7
5.2.5 Demonstration (3): Using the value of a work field	5-8
5.3 Performing arithmetic operations	5-9
5.3.1 What you can do	5-9
5.3.2 How to do it	5-9
5.3.3 Demonstration	5-9
5.4 Directing processing flow	5-12
5.4.1 What you can do	5-12
5.4.2 How to do it	5-12
5.4.3 Demonstration	5-12
Chapter 6. Controlling Total Lines	6-1
6.1 The parameters to use with totals	6-4
6.2 Selectively printing total amounts	6-5
6.2.1 What you can do	6-5
6.2.2 How to do it	6-5
6.2.3 Demonstration	6-5
6.3 Labeling total lines	6-7
6.3.1 What you can do	6-7
6.3.2 How to do it	6-7
6.3.3 Demonstration	6-7
6.4 Generating total amounts only	6-9
6.4.1 What you can do	6-9
6.4.2 How to do it	6-9
6.4.3 Demonstration	6-9
6.5 Suppressing all totals	6-11

6.5.1	What you can do	6-11
6.5.2	How to do it	6-11
6.5.3	Demonstration	6-11
6.6	Generating subtotals	6-13
6.6.1	What you can do	6-13
6.6.2	How to do it	6-13
6.6.3	Demonstration	6-13
6.7	Modifying subtotal and total line placement	6-16
6.7.1	What you can do	6-16
6.7.2	How to do it	6-16
6.7.3	Demonstration	6-16
6.8	Selectively printing output lines	6-18
6.8.1	What you can do	6-18
6.8.2	How to do it	6-18
6.8.3	Demonstration (1): Selecting totals output	6-18
6.8.4	Demonstration (2): Using a work field to print total line labels	6-21
6.9	Obtaining sort field values	6-24
6.9.1	What you can do	6-24
6.9.2	How to do it	6-24
6.9.3	Demonstration	6-24
6.10	Obtaining sorted work field values	6-26
6.10.1	What you can do	6-26
6.10.2	How to do it	6-26
6.10.3	Demonstraton	6-26
Chapter 7. Generating Reports From Database Files 7-1		
7.1	Listing data	7-4
7.1.1	What you can do	7-4
7.1.2	How to do it	7-4
7.1.3	Demonstration	7-4
7.2	Using logical records	7-6
7.2.1	What you can do	7-6
7.2.2	How to do it	7-6
7.2.3	Demonstration	7-6
7.3	Selective processing	7-8
7.3.1	What you can do	7-8
7.3.2	How to do it	7-8
7.3.3	Demonstration	7-8
Chapter 8. Building and Using Data Tables 8-1		
8.1	Creating tables	8-4
8.1.1	What you can do	8-4
8.1.2	How to do it	8-4
8.1.3	Demonstration (1): Creating a table from a sequential file	8-6
8.1.4	Demonstration (2): Creating a table from the database	8-8
8.1.5	Demonstration (3): Creating a table from an existing table	8-10
8.1.6	Demonstration (4): Creating a totals-only table	8-12
8.2	Retrieving data tables	8-13
8.2.1	What you can do	8-13
8.2.2	How to do it	8-13

8.2.3 Demonstration	8-13
8.3 Modifying data tables	8-15
8.3.1 What you can do	8-15
8.3.2 How to do it	8-15
8.3.3 Demonstration (1): Adding rows to an existing table	8-15
8.3.4 Demonstration (2): Replacing rows of an existing table	8-17
8.3.5 Demonstration (3): Deleting a table	8-18
8.3.6 Demonstration (4): Regenerating a table	8-19
8.4 Consolidating tables	8-23
8.4.1 What you can do	8-23
8.4.2 How to do it	8-23
8.4.3 Demonstration	8-25

Advanced Topics

Chapter 9. Totals Processing Techniques	9-1
9.1 Calculations on accumulated totals	9-4
9.1.1 What you can do	9-4
9.1.2 How to do it	9-4
9.1.3 Demonstration	9-4
9.2 Multiple-level subtotals	9-6
9.2.1 What you can do	9-6
9.2.2 How to do it	9-6
9.2.3 Demonstration	9-6
9.3 Creating a sparse listing	9-10
9.3.1 What you can do	9-10
9.3.2 How to do it	9-10
9.3.3 Demonstration	9-10
9.4 Obtaining work field values	9-13
9.4.1 Demonstration	9-13
Chapter 10. Using Subscripts	10-1
10.1 Explicit subscripts	10-4
10.1.1 What you can do	10-4
10.1.2 How to do it	10-4
10.1.3 Demonstration (1): Using a literal subscript	10-4
10.1.4 Demonstration (2): Using a counter as a subscript	10-6
10.2 Zero subscripts	10-8
10.2.1 What you can do	10-8
10.2.2 How to do it	10-8
10.2.3 Demonstration	10-8
10.3 Fixed repeating fields	10-11
10.3.1 What you can do	10-11
10.3.2 How to do it	10-11
10.3.3 Demonstration	10-11
10.4 Variable repeating groups	10-13
10.4.1 What you can do	10-13
10.4.2 How to do it	10-13
10.4.3 Demonstration	10-13
10.5 Floating fields	10-15

10.5.1	What you can do	10-15
10.5.2	How to do it	10-15
10.5.3	Demonstration	10-15
10.6	Obtaining accumulated totals	10-17
10.6.1	What you can do	10-17
10.6.2	How to do it	10-17
10.6.3	Demonstration	10-17
10.7	Obtaining specific field values	10-19
10.7.1	What you can do	10-19
10.7.2	How to do it	10-19
10.7.3	Demonstration	10-19
10.8	Obtaining sort-key values	10-22
10.8.1	What you can do	10-22
10.8.2	How to do it	10-22
10.8.3	Demonstration	10-22
Chapter 11. File Matching 11-1		
11.1	What file matching is	11-3
11.1.1	How file matching works	11-3
11.1.2	How to code match-file runs	11-7
11.2	Matching single-occurrence files	11-8
11.2.1	Demonstration	11-8
11.3	Matching multiple transactions with a single-entry master	11-10
11.3.1	Demonstration (1): dropping unmatched records	11-10
11.3.2	Demonstration (2): selecting matching account numbers	11-13
11.4	Listing accounts without transactions	11-15
11.4.1	Demonstration	11-15
11.5	Listing transactions not on the master file	11-18
11.5.1	Demonstration	11-18
11.6	Using the match-file facility for table initialization	11-21
11.6.1	What you can do	11-21
11.6.2	How to do it	11-21
11.6.3	Demonstration	11-21
11.7	Using files defined to the data dictionary	11-25
11.7.1	What you can do	11-25
11.7.2	How to do it	11-25
11.7.3	Demonstration	11-25
11.8	Creating unique match-key names	11-26
11.8.1	What you can do	11-26
11.8.2	How to do it	11-26
11.8.3	Demonstration	11-26
11.9	Qualified fields	11-27
11.9.1	What you can do	11-27
11.9.2	How to do it	11-27
11.9.3	Demonstration	11-27
Chapter 12. Using and Modifying Copied Code 12-1		
12.1	Copying stored code (USE parameter)	12-4
12.1.1	What you can do	12-4
12.1.2	How to do it	12-4

12.2	Assigning values to symbolic fields (USE parameter)	12-5
12.2.1	What you can do	12-5
12.2.2	How to do it	12-5
12.2.3	Demonstration	12-5
12.3	Assigning default values to symbolic parameters (USE parameter)	12-8
12.3.1	What you can do	12-8
12.3.2	How to do it	12-8
12.3.3	Demonstration	12-8
12.4	Nesting the USE parameter	12-10
12.4.1	What you can do	12-10
12.4.2	How to do it	12-10
12.4.3	Demonstration	12-10
12.5	Modifying code (USE parameter)	12-11
12.5.1	What you can do	12-11
12.5.2	How to do it	12-11
12.5.3	Demonstration (1): Changing report numbers and character strings .	12-11
12.5.4	Demonstration (2): Using a KEEP clause	12-13
12.5.5	Demonstration (3): Using a DROP clause	12-14
12.5.6	Demonstration (4): Using the RENUMBER clause	12-15
12.6	Copying and modifying code (=COPY parameter)	12-18
12.6.1	What you can do	12-18
12.6.2	How to do it	12-18
12.6.3	Demonstration	12-18
12.7	Copying and modifying code (=MACRO parameter)	12-20
12.7.1	What you can do	12-20
12.7.2	How to do it	12-20
12.7.3	Demonstration (1): Providing symbolic parameter values	12-20
12.7.4	Demonstration (2): Modifying parameters and the report number .	12-22
12.8	Listing the contents of a data file	12-24
12.8.1	What you can do	12-24
12.8.2	How to do it	12-24
12.8.3	Demonstration (1): Printing fields with AMLIST3	12-24
12.8.4	Procedure	12-24
12.8.5	Demonstration (2): Modifying code using WITH VALUES and CHANGE	12-26
Chapter 13. Additional Standard File Facilities		13-1
13.1	Creating nonprint report output	13-4
13.1.1	What you can do	13-4
13.1.2	How to do it	13-4
13.1.3	Demonstration (1): Writing to a sequential file	13-5
13.1.4	Demonstration (2): Converting to packed decimal format	13-6
13.1.5	Demonstration (3): Writing complete records	13-8
13.1.6	Demonstration (4): Writing totals-only	13-8
13.1.7	Demonstration (5): Writing variable-length records	13-9
13.1.8	Demonstration (6): Writing from the database	13-10
13.2	Creating variable headings	13-11
13.2.1	What you can do	13-11
13.2.2	How to do it	13-11
13.2.3	Demonstration (1): Using the SORT parameter	13-11
13.2.4	Demonstration (2): Using the SORT/NOSORT parameters	13-12

13.3 Accessing non-database files defined in the data dictionary	13-14
13.3.1 What you can do	13-14
13.3.2 How to do it	13-14
Chapter 14. Additional CA-IDMS/DB Facilities	14-1
14.1 How to prepare for record retrieval	14-4
14.2 Retrieving partial paths	14-5
14.2.1 What you can do	14-5
14.2.2 How to do it	14-5
14.2.3 Demonstration	14-5
14.3 Retrieving record types by name	14-7
14.3.1 What you can do	14-7
14.3.2 How to do it	14-7
14.3.3 Demonstration	14-7
14.4 Retrieving record types by key	14-10
14.4.1 What you can do	14-10
14.4.2 How to do it	14-10
14.4.3 Demonstration (1): Accessing records by key	14-11
14.4.4 Demonstration (2): Accessing records with a key file	14-12
14.5 Retrieving stand-alone records	14-14
14.5.1 What you can do	14-14
14.5.2 How to do it	14-14
14.5.3 Demonstration	14-14
14.6 Retrieving all record occurrences	14-16
14.6.1 What you can do	14-16
14.6.2 How to do it	14-16
14.6.3 Demonstration	14-16
14.7 Testing for record occurrences	14-19
14.7.1 What you can do	14-19
14.7.2 How to do it	14-19
14.7.3 Demonstration	14-19
14.8 Accessing bill-of-materials structures	14-21
14.8.1 What you can do	14-21
14.8.2 How to do it	14-21
14.8.3 Demonstration	14-21
14.9 Accessing multiple-member sets	14-24
14.9.1 What you can do	14-24
14.9.2 How to do it	14-24
14.9.3 Demonstration	14-24

Using SQL Tables

Chapter 15. Retrieving Data With SQL	15-1
15.1 Getting the exact column names for an SQL table	15-4
15.2 Using the AS clause to rename SQL columns	15-5
15.3 Retrieving floating point data from SQL columns	15-6
15.4 Displaying an SQL column with data type BINARY	15-7
15.5 How to test and display null values	15-9
15.6 Handling null values at total time	15-10

15.7 Understanding CA-Culprit decimal point handling	15-12
15.8 Interpreting common error messages in SQL retrieval	15-13
Chapter 16. Creating New SQL Data Tables	
16.1 Create table syntax generated by CA-Culprit	16-4
16.2 Converting ASF tables to SQL tables	16-5
16.3 Assigning Null values to an SQL column	16-6
16.4 How CA-Culprit handles data insertion errors	16-7
Chapter 17. Adding, Replacing, and Dropping Data on SQL Tables	
17.1 Updating SQL columns of type DATE	17-3
17.2 Getting the exact syntax for updating an SQL table	17-4
17.3 Drop table example	17-5

Appendices

Appendix A. The Personnel File Description	A-1
Appendix B. Debugging a CA-Culprit Report	
B.1 Reviewing CA-Culprit listings and messages	B-4
B.1.1 Sequential parameter listing	B-4
B.1.2 Input parameter listing	B-4
B.1.3 Runtime messages	B-5
B.1.4 The report	B-6
B.2 Using CULLUS48	B-8
B.3 Printing additional report information	B-9
B.3.1 Demonstration	B-9
Appendix C. Coding for Efficiency	
C.1 Eliminating unnecessary records	C-4
C.2 SELECT vs. type 7 logic	C-5
C.3 Avoiding unnecessary logic	C-6
C.4 Combining tasks for multiple reports	C-7
C.5 Avoiding unwanted buffer dumps	C-8
C.6 Creating subtotal work fields for presorted input files	C-9
C.7 JCL considerations	C-11
C.8 Miscellaneous hints	C-12
Appendix D. Precoded CA-Culprit User Modules	
D.1 Input modules	D-4
D.2 Procedure modules	D-5
D.3 Output modules	D-7
Appendix E. Employee Database Subschema	
E.1 Data structure diagram	E-4
E.2 Subschema listing	E-5
Appendix F. Summary of CA-Culprit parameters	
F.1 The basic CA-Culprit functions	F-4
F.2 Processing operations	F-5

F.3	Summary of advanced capabilities for standard files	F-6
F.4	Summary of parameters for using database data	F-8
F.5	Summary of advanced capabilities for accessing the database	F-9
F.6	Minimum coding requirements for using data tables	F-10
F.7	Minimum coding requirements for using SQL tables	F-12
Appendix G. How Totals Processing Works		G-1
G.1	The extract processing phase (CULL)	G-6
G.2	The output phase (CULE)	G-7
G.3	The processing steps when an extract file is read	G-8
Index	X-1

How to use this Manual

What this manual is about

- Generating CA-Culprit reports from conventional files and the CA-IDMS/DB database
- Creating and modifying tables in the CA-ICMS (Information Center Management System) environment
- Debugging CA-Culprit report code
- Coding for efficiency

Who should use this manual

- New CA-Culprit users with little or no data processing background
- Experienced CA-Culprit users with programming backgrounds

How this manual is organized

The manual is divided into three parts:

- Part I is for new CA-Culprit users who need to produce CA-Culprit reports immediately.
- Part II is for experienced CA-Culprit users who need to perform complex CA-Culprit operations.
- Part III is for experienced CA-Culprit users who need to retrieve, create, modify, and drop SQL tables.

Each part is a series of tasks. Each task includes:

- How-to instructions
- An example that includes:
 - A report objective
 - The code to produce the report
 - Explanation of the code
 - The printed report

How to use this manual

- New users:
 - Begin with Chapters 1 through 4.
 - Refer to additional topics, as needed, in Chapters 5 through 8.
 - Examples can be used as models for reports or as learning exercises.
- Experienced users:
 - Refer to topics, as needed.
 - Refer to the *CA-Culprit Reference Guide*, for additional information about syntax and internal-processing sequences.

Related documentation

For more information about CA-Culprit, see the following:

CA-Culprit Reference Guide

CA-Culprit Messages and Codes

CA-Culprit User Modules

The Basics

Chapter 1. Getting Started with CA-Culprit

1.1	What is CA-Culprit and what can you do with it?	1-3
1.2	How to produce a CA-Culprit report	1-4
1.2.1	Step 1. Planning the report	1-4
1.2.2	Step 2. Writing the code	1-5
1.2.3	Step 3. Executing the code	1-5
1.2.4	Step 4. Debugging the code	1-5
1.3	What you should know before you begin	1-7
1.3.1	How information is organized in a computer	1-7
1.3.2	What a file description tells you	1-9
1.3.3	How to use information in a file description	1-9

1.1 What is CA-Culprit and what can you do with it?

CA-Culprit is a batch utility with which you can generate files, printed reports, and data tables from conventional and database files. You can print these reports or write them to conventional files. Tables are stored in the database as part of CA-IDMS/DB and CA-ICMS.

CA-Culprit provides a means for writing detailed or summary reports with a minimum of coding requirements. By using CA-Culprit parameters, you can:

- Retrieve data from conventional file and database structures
- Read and process up to 32 conventional files in one CA-Culprit run
- Produce automatically formatted or customized printed reports
- Produce detailed or summary reports
- Produce up to 100 reports in a single CA-Culprit run
- Produce reports on standard or special forms
- Write reports to conventional files on cards, tape, or disk
- Produce nonprint output in any format (for example, packed decimal)
- Read, create, update, and store data tables in a CA-ICMS environment
- Perform logical operations (computations, comparisons, sorting, file matching, branching, calls to external modules)
- Control processing order

1.2 How to produce a CA-Culprit report

Producing a CA-Culprit report involves four basic steps:

1. Planning the report
2. Writing the code
3. Executing the code
4. Debugging the code

1.2.1 Step 1. Planning the report

1. Define the report objectives:
 - a. What information do you want in the report?
 - b. Do you need each data occurrence?
 - c. Do you need totals? Subtotals?
2. Identify the data source:
 - a. What is the name of the file?
 - b. What data is in the file?
 - c. How is the file organized?
3. Define the report output:
 - a. Do you need a title? Subtitles?
 - b. What columns do you need?
 - c. What are the column headings?
 - d. Do you need total line labels?
4. Define the processing functions required to obtain the information needed in the report:
 - a. Are certain data items to be selected?
 - b. Should the data be sorted?
 - c. What calculations (if any) are needed?
 - d. Is testing (comparisons) needed between data items?
 - e. Will branching be used?
 - f. What logical sequence is required?
 - g. Other procedures?

1.2.2 Step 2. Writing the code

You tell CA-Culprit what to do and when to do it by coding instructions on CA-Culprit parameters. You identify each CA-Culprit parameter by a word or number you code in a specific column position.

A few general rules apply to all CA-Culprit coding:

- Enter code in uppercase letters.
- Enter coding in the first ten columns in exactly the correct position
 - Mnemonic code that describes input files starts in column 2:
IN 200 F 400
REC EMP-NAME
 - Numeric code starts in column 2:
0151*010 ' '
 - Mnemonic code that indicates copying and modifying stored code starts in column 1:
USE
=COPY
=MACRO
=MEND
 - Continuation lines start in column 1:
* ...
- Entries following column 10 need not begin in a specific column, but must be separated by a comma or at least one space.

1.2.3 Step 3. Executing the code

You execute CA-Culprit code with Job Control Language (JCL). Consult your data-processing department for the JCL used to run CA-Culprit reports at your site.

1.2.4 Step 4. Debugging the code

No matter how careful you are, errors in your code will occur from time to time. At the end of each run, CA-Culprit provides listings that flag errors:

- The Sequential Parameter Listing
- The Input Parameter Listing
- Run-Time Messages

►► Appendix B, “Debugging a CA-Culprit Report” on page B-1 describes the CA-Culprit listings and how to use them to debug CA-Culprit code.

1.2 How to produce a CA-Culprit report

1.3 What you should know before you begin

You do not need to be an expert to write CA-Culprit reports. You do not need to be a computer programmer. A few basic data processing concepts are enough to get you started. From there, you can learn as you go.

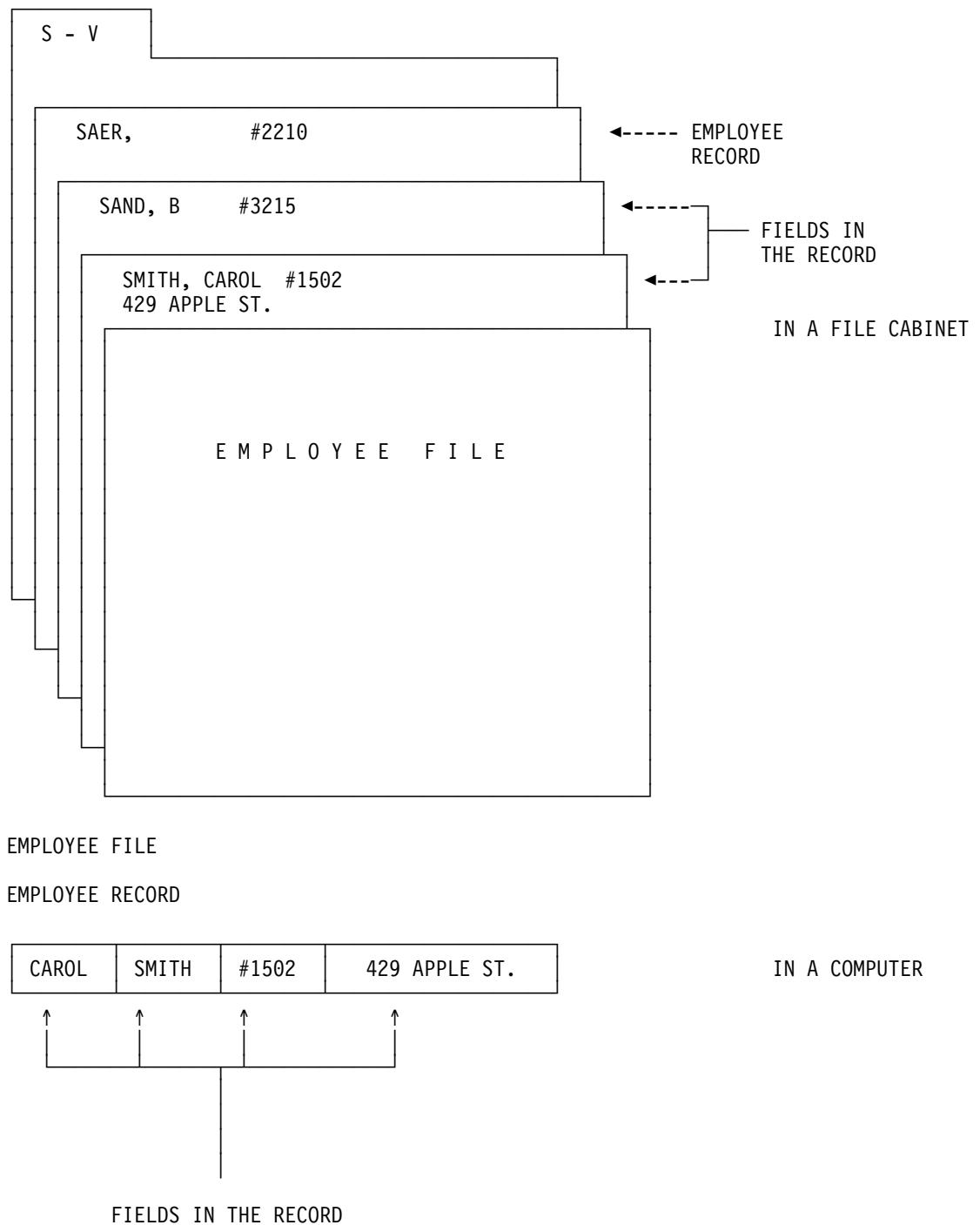
1.3.1 How information is organized in a computer

Information is organized in a computer in much the same way written information is organized in a file cabinet:

- Collections of related information, called **files**, are analogous to file folders. For example, most companies have personnel files.
- Each file has subdivisions called **records**, which are analogous to the papers in a folder. For example, the personnel file contains employee records.
- Each record contains basic units of data called **fields**, which are analogous to related fragments of information contained on the papers in a folder. For example, an employee record contains a name field.

The figure below illustrates the relationships between files, records, and fields:

1.3 What you should know before you begin



1.3.2 What a file description tells you

When you obtain a description of the data file you are going to use for your report, you will notice that the file is described in these terms:

- **Record size**, which indicates the length of a single record measured in characters or bytes.
- **Block size**, which indicates the number of records stored next to each other before there is a spacing interval. The block size is also measured in bytes.
- **Record format**, which indicates whether the records in the file equal (fixed) in length, vary in length, or have an undefined length.
- **File type**, which indicates how the records are organized in the file. The possibilities are a sequential file, an ISAM file, a card file, a VSAM file, or a database.

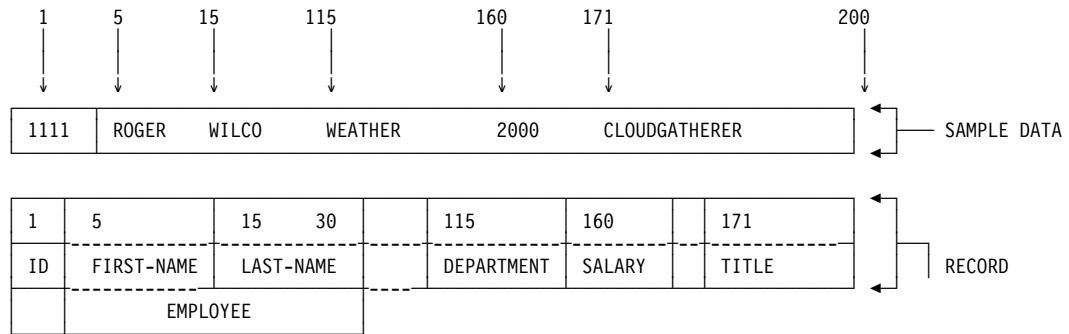
The file description also gives size and location of each data field:

- **Start position**, which is the beginning location of the field.
- **Length**, which is the amount of space allocated for the particular field. The field length is expressed in bytes.
- **Data type**, which identifies the data as: alphanumeric (a combination of alphabetic characters and numbers) or numeric (zoned decimal, packed decimal, unsigned packed decimal, binary, or bit).

1.3.3 How to use information in a file description

You don't have to understand the differences between the data types to use CA-Culprit, but you must identify them correctly. You just have to read the information off of the file description and enter it correctly into your code.

The figure below shows the layout of a record within an employee data file. Sample data illustrates how an occurrence of the employee record might appear:



The sample record above has six data fields that can be used for a report. Note that the record is always 200 bytes long, even if all the space is not needed by a particular word or number. The data field lengths also remain constant, with blanks used to fill out the space.

For example, the record allocates a 10-byte space for FIRST-NAME; therefore, the five characters in ROGER are followed by five blanks so that WILCO begins in position 15, the starting position of LAST-NAME.

Chapter 2. Generating a Basic Report from Standard Files

2.1 Locating the data	2-4
2.2 Coding the report	2-5
2.2.1 How to define the input	2-5
2.2.2 How to define the output	2-6
2.2.3 Demonstration	2-7

This chapter takes you step by step through the process of writing a simple CA-Culprit report. This report, like all the sample reports in this manual, can be used as a report model for your documents or as a hands-on exercise.

Any terms used in Volume I of this manual that are not discussed in Chapter 1, “Getting Started with CA-Culprit” on page 1-1 are defined within the context of the subject matter under discussion.

For our first report we will create the listing of employee names and salaries shown in the report below, created with just five lines of code:

The report:

JUNE	BLOOMER	15,000.00
EDWARD	HUTTON	44,000.00
RUPERT	JENSON	82,000.00
MARIANNE	KIMBALL	45,000.00
DORIS	KING	14,500.00
BRIAN	NICEMAN	14,000.00
HERBERT	CRANE	75,000.00
JANE	FERNDALE	22,500.00
KATHERINE	O'HEARN	42,500.00
RALPH	TYRO	20,000.00
HENRIETTA	HENDON	240,000.00
THEMIS	PAPAZEUS	100,000.00
JOHN	RUPEE	80,000.00
ROBBY	WILDER	90,000.00
		2,522,500.00

2.1 Locating the data

The data for our report is stored in the personnel file of the Commonwealth Corporation. The figure below describes part of the file. The complete description is in Appendix A, “The Personnel File Description” on page A-1.

Information about the data you will use to create your CA-Culprit report is contained in a file description similar to the one above. The description is usually provided by your data-processing department.

RECORD SIZE	200 bytes
BLOCK SIZE	400 bytes
RECORD FORMAT	FIXED (F)
FILE TYPE	SEQUENTIAL (PS)

Data Field	Start Position	Length (bytes)	Data Type
EMPLOYEE-NAME	5	25	Alphanumeric
FIRST-NAME	5	10	Alphanumeric
LAST-NAME	15	15	Alphanumeric
DEPT-NAME	115	45	Alphanumeric
SALARY-AMOUNT	160	7	Packed Decimal
TITLE	171	20	Alphanumeric

Information about the data you will use to create your CA-Culprit report is contained in a file description similar to the one above. The description is usually provided by your data-processing department.

2.2 Coding the report

A simple CA-Culprit report is produced with three instructions (**parameters**):

- The **INPUT (IN)** parameter, which tells CA-Culprit how the file containing the data is organized
- **REC** parameters, which tell CA-Culprit what data to use
- **Type 5** parameters, which tell CA-Culprit what and where to print the data

It is important to remember that one INPUT parameter with at least one REC parameter and one type 5 parameter are minimal requirements for any CA-Culprit report produced from standard files.

Our first report lists the names of employees and their salaries from the data stored in the personnel file. We will create the report from the three required parameters.

2.2.1 How to define the input

To define the input data, code:

1. An INPUT (IN) parameter to describe the file
2. A REC parameter for each data field used in the report

Sample Parameters

Parameters	Description
IN 200 F 400 PS(TAPE)	200 tells CA-Culprit that each record in the personnel file contains 200 characters (bytes). F indicates fixed-length records. 400 indicates that the file is blocked in 400 byte segments. PS(TAPE) specifies a physical sequential file on a tape device.
REC EMPLOYEE-NAME 5 25	EMPLOYEE-NAME 5 25 directs CA-Culprit to use the data stored in positions 5 through 25 of each record.
REC SALARY-AMOUNT 160 7 3 DP=2	SALARY-AMOUNT 160 7 directs CA-Culprit to use the data from positions 160 through 166 for salary information. 3 tells CA-Culprit that the salary data is stored as packed decimal. DP=2 tells CA-Culprit that SALARY-AMOUNT has two decimal places.

2.2.2 How to define the output

To define the report output, code a **type 5** parameter for each data item. The information on a type 5 parameter is coded in the following order, from left to right:

1. A 2-digit report number
2. The parameter type (**5**)
3. A 1-digit print-row identifier
4. The column placement, consisting of either an asterisk (*) followed by a 3-digit number for relative placement or a 4-digit number for specific placement.
5. A spacing indicator consisting of a blank for a single space, a zero (0) for a double space, or a dash (-) for a triple space
6. The name of the data item to appear in the column

Sample Parameters

Parameters	Description
0151*010 EMPLOYEE	<p>01 is the assigned report number.</p> <p>5 specifies a type 5 parameter.</p> <p>1 specifies that the employee name appears on the first print row.</p> <p>*010 specifies that CA-Culprit determines the placement of the employee name on the page.</p> <p>Low-number columns print on the left-hand side of the page; high-number columns print progressively to the right of the first column. The blank in column 10 specifies single spacing.</p> <p>EMPLOYEE directs CA-Culprit to print the employee's name in this column.</p>
0151*020 SALARY	<p>0151 specifies the report number, parameter type, and first print row.</p> <p>*020 directs CA-Culprit to print SALARY to the right of the employee name.</p> <p>SALARY directs CA-Culprit to print the salary amount in this column.</p>

2.2.3 Demonstration

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC SALARY        160   5   3   DP=2
0151*010 EMPLOYEE
0151*020 SALARY

```

Result:

2.2 Coding the report

JUNE	BLOOMER	15,000.00
EDWARD	HUTTON	44,000.00
RUPERT	JENSON	82,000.00
MARIANNE	KIMBALL	45,000.00
DORIS	KING	14,500.00
BRIAN	NICEMAN	14,000.00
HERBERT	CRANE	75,000.00
JANE	FERNDALE	22,500.00
GEORGE	FONRAD	14,750.00
ROBIN	GARDNER	14,000.00
DOUGLAS	KAHALLY	20,000.00
TERENCE	KLWELLEN	43,000.00
SANDY	KRAAMER	14,000.00
HERBERT	LIPSICH	18,500.00
NANCY	TERNER	13,000.00
BETH	CLOUD	52,750.00
ALAN	DONOVAN	33,500.00
DANIEL	MOON	72,000.00
ROY	ANDALE	33,500.00
HARRY	ARM	46,000.00
C.	BREEZE	38,000.00
CAROLYN	CROW	37,500.00
BURT	LANCHESTER	54,500.00
RENE	MAKER	85,000.00
RICHARD	MUNYON	36,000.00
RICHARD	WAGNER	47,000.00
TOM	FITZHUGH	13,000.00
CYNTHIA	JOHNSON	13,500.00
MADELINE	ORGRATZI	39,000.00
ELEANOR	PEOPLES	80,000.00
MICHAEL	ANGELO	18,000.00
MONTE	BANK	80,000.00
CHARLES	BOWER	38,500.00
JOCK	JACKSON	34,000.00
CAROL	MCDougall	18,000.00
LAURA	PENMAN	39,000.00
BETSY	ZEDI	37,000.00
TERRY	CLOTH	38,000.00
PHINEAS	FINN	45,000.00
JOE	KASPAR	31,000.00
MARK	TIME	33,000.00
ROGER	WILCO	80,000.00
JANE	DOUGH	33,000.00
JAMES	GALLWAY	33,000.00
JENNIFER	GARFIELD	65,000.00
PERCY	GRANGER	34,500.00
VLADIMIR	HEAROWITZ	33,000.00
JAMES	JACOBI	55,000.00
JULIE	JENSEN	37,000.00
LARRY	LITERATA	37,500.00
KATHERINE	O'HEARN	42,500.00
RALPH	TYRO	20,000.00
HENRIETTA	HENDON	240,000.00
THEMIS	PAPAZEUS	100,000.00
JOHN	RUPEE	80,000.00
ROBBY	WILDER	90,000.00
		2,522,500.00

Chapter 3. Enhancing the Basic Report

3.1 Adding a title	3-4
3.1.1 What you can do	3-4
3.1.2 How to do it	3-4
3.1.3 Demonstration	3-4
3.2 Adding column headings	3-6
3.2.1 What you can do	3-6
3.2.2 How to do it	3-6
3.2.3 Demonstration (1): Formatting multiple Line column headings	3-7
3.2.4 Demonstration (2): Adding subtitles	3-8
3.3 Modifying column widths	3-11
3.3.1 What you can do	3-11
3.3.2 How to do it	3-11
3.3.3 Demonstration	3-11
3.4 Formatting numeric data	3-13
3.4.1 What you can do	3-13
3.4.2 How to do it	3-13
3.4.3 Demonstration (1): Using formatting codes	3-13
3.4.4 Demonstration (2): Using formatting patterns	3-14
3.5 Adjusting line length and page depth	3-16
3.5.1 What you can do	3-16
3.5.2 How to do it	3-16
3.5.3 Demonstration	3-16

CA-Culprit provides optional parameters and codes you can use to make a report easier to read. In this chapter, some the CA-Culprit most commonly used enhancement and numeric formatting options are introduced.

3.1 Adding a title

3.1.1 What you can do

You can supply a title for your report. CA-Culprit automatically prints this title at the top of each page, along with the report number, current date, and page number.

3.1.2 How to do it

To add a title, code a **type 3** parameter in the following order, from left to right:

1. A 2-digit report number
2. The parameter type (**3**)
3. A title (not to exceed 50 characters, including spaces)

3.1.3 Demonstration

Objective: This example adds a title EMPLOYEE SALARY LISTING to the report.

Parameters

Parameters	Description
013 EMPLOYEE SALARY LISTING	01 is the report number. 3 specifies a type 3 parameter. EMPLOYEE SALARY LISTING is the title for the printed report.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC SALARY        160   5 3 DP=2
013 EMPLOYEE SALARY LISTING.
0151*010 EMPLOYEE
0151*020 SALARY
```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING	04/30/99 PAGE	1
ROY	ANDALE	\$33,500.00	
HARRY	ARM	\$46,000.00	
MONTE	BANK	\$80,000.00	
CHARLES	BOWER	\$38,000.00	
C.	BREEZE	\$38,000.00	
TERRY	CLOTH	\$38,000.00	
BETH	CLOUD	\$52,750.00	
HERBERT	CRANE	\$75,000.00	
CAROLYN	CROW	\$37,500.00	
ALAN	DONOVAN	\$33,500.00	
JANE	DOUGH	\$33,000.00	
JANE	FERNDALE	\$22,500.00	
PHINEAS	FINN	\$45,000.00	
JAMES	GALLWAY	\$33,000.00	
JENNIFER	GARFIELD	\$65,000.00	
PERCY	GRANGER	\$34,500.00	

3.2 Adding column headings

3.2.1 What you can do

You can direct CA-Culprit to obtain column headings from:

- Field names supplied on REC parameters
- Text added to REC parameter lines
- Text added to type 5 parameter lines
- Text added to type 4 parameter lines

3.2.2 How to do it

You can use type 5 or type 4 parameters to create heading lines.

Method 1: Use any of the following codes on **type 5** parameters:

- **HF**, which directs CA-Culprit to the associated field name:

REC **EMP-NAME** 1 25
0151*020 EMP-NAME **HF**

or

010 **EMP-NAME**
0151*020 EMP-NAME **HF**

Column heading is EMP-NAME.

- **HR**, which directs CA-Culprit to heading text that is enclosed in single quotation marks on the record description (REC) line:

REC EMP-NAME 1 25 '**EMPLOYEE**'

Column heading is EMPLOYEE.

- **HH**, which directs CA-Culprit to the text enclosed in single quotation marks following the HH option on the type 5 line:

0151*020 EMP-NAME **HH 'EMPLOYEE NAME'**

Column heading is EMPLOYEE NAME.

A multiple-line heading is generated by enclosing the text for each line within a separate set of quotation marks:

0151*020 EMP-NAME HH 'EMPLOYEE' 'NAME'

or

REC EMP-NAME 'EMPLOYEE' 'NAME'
0151*020 EMP-NAME HR

Column heading is: EMPLOYEE
NAME

Method 2: Use type 4 parameters to create the headings. The type 4 parameter, typically used for special-purpose heading lines, is coded in the following order, from left to right:

1. A 2-digit report number
2. The parameter type (**4**)
3. A numeral to identify the level of the line
4. A column placement indicator consisting of either an asterisk (*) followed by a 3-digit number for relative placement or a 4-digit number for specific placement
5. A spacing indicator, such as a blank for single spacing, a zero (0) for double spacing, or a dash (-) for triple spacing
6. The text of the heading enclosed in single quotation marks or a field name

3.2.3 Demonstration (1): Formatting multiple Line column headings

Objective: This report prints a 2 line heading for the employee column and a 3 line heading for the salary column.

Parameters

Parameters	Description
REC EMPLOYEE 5 25 'EMPLOYEE' 'NAME'	'EMPLOYEE' is the first line of the EMPLOYEE column heading. 'NAME' is the second line.
0151*020 SALARY HH 'ANNUAL' 'SALARY' 'AMOUNT'	'ANNUAL' is the first line of the salary column heading. 'SALARY' is the second line. 'AMOUNT' is the third line.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5 25      'EMPLOYEE'  'NAME'
REC SALARY        160 5 3 DP=2
013 EMPLOYEE SALARY LISTING
0151*010 EMPLOYEE          HR
0151*020 SALARY           HH 'ANNUAL' 'SALARY' 'AMOUNT'
```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING	04/30/99 PAGE	1
	EMPLOYEE NAME	ANNUAL SALARY REPORT	
ROY	ANDALE	\$33,500.00	
HARRY	ARM	\$46,000.00	
MONTE	BANK	\$80,000.00	
CHARLES	BOWER	\$38,000.00	
C.	BREEZE	\$38,000.00	
TERRY	CLOTH	\$38,000.00	
BETH	CLOUD	\$52,750.00	
HERBERT	CRANE	\$75,000.00	
CAROLYN	CROW	\$37,500.00	
ALAN	DONOVAN	\$33,500.00	
JANE	DOUGH	\$33,000.00	
JANE	FERNDALE	\$22,500.00	
PHINEAS	FINN	\$45,000.00	
JAMES	GALLWAY	\$33,000.00	
JENNIFER	GARFIELD	\$65,000.00	
PERCY	GRANGER	\$34,500.00	

3.2.4 Demonstration (2): Adding subtitles

Objective: This example adds heading lines that function as subtitles, not column headings.

Parameters

Parameters	Description
01410054 'FROM 01/01/91 TO 06/01/91'	01 is the report number. 4 specifies a type 4 parameter. 1 specifies that 'FROM ...' is printed on the first row of the subtitle. 0054 indicates that this row of text begins exactly 54 spaces from the left margin.
0142*001 'ALL DEPARTMENTS'	014 specifies the report number and the parameter type, respectively. 2 specifies that 'ALL DEPARTMENTS' is printed on the second row of the subtitle. *001 specifies that this row of text is left justified.
0143*010- ' '	014 specifies the report number and the parameter type, respectively. 3 specifies the third line of the subtitle. *010 specifies a left-hand column position. - tells CA-Culprit to skip two lines before printing the next line. ' ' prints a blank space, which in this case creates a blank line.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC SALARY        160   5   3 DP=2
013 EMPLOYEE SALARY LISTING
01410054 'FROM 01/01/91 TO 06/01/91'
0142*001 'ALL DEPARTMENTS'
0143*010- ' '
0151*010 EMPLOYEE
0151*020 SALARY

```

Result:

3.2 Adding column headings

REPORT NO. 01	EMPLOYEE SALARY LISTING	09/24/99 PAGE 1 FROM 01/01/91 TO 06/01/91
ALL DEPARTMENTS		
JUNE	BLOOMER	15,000.00
EDWARD	HUTTON	44,000.00
RUPERT	JENSON	82,000.00
MARIANNE	KIMBALL	45,000.00
DORIS	KING	14,500.00
BRIAN	NICEMAN	14,000.00
HERBERT	CRANE	75,000.00
JANE	FERNDALE	22,500.00
GEORGE	FONRAD	14,750.00
ROBIN	GARDNER	14,000.00
DOUGLAS	KAHALLY	20,000.00
TERENCE	KLWELLEN	43,000.00
SANDY	KRAAMER	14,000.00
HERBERT	LIPSICH	18,500.00
NANCY	TERNER	13,000.00
BETH	CLOUD	52,750.00
ALAN	DONOVAN	33,500.00
DANIEL	MOON	72,000.00
ROY	ANDALE	33,500.00
HARRY	ARM	46,000.00
C.	BREEZE	38,000.00
CAROLYN	CROW	37,500.00
BURT	LANCHESTER	54,500.00
RENE	MAKER	91,000.00
RICHARD	MUNYON	36,000.00
RICHARD	WAGNER	47,000.00
TOM	FITZHUGH	13,000.00
CYNTHIA	JOHNSON	13,500.00
MADELINE	ORGRATZI	39,000.00
ELEANOR	PEOPLES	80,000.00
MICHAEL	ANGELO	18,000.00
MONTE	BANK	80,000.00
CHARLES	BOWER	38,500.00
JOCK	JACKSON	34,000.00
CAROL	MCDougall	18,000.00
LAURA	PENMAN	39,000.00
BETSY	ZEDI	37,000.00
TERRY	CLOTH	38,000.00
PHINEAS	FINN	45,000.00
JOE	KASPAR	31,000.00
MARK	TIME	33,000.00
ROGER	WILCO	80,000.00
JANE	DOUGH	33,000.00
JAMES	GALLWAY	33,000.00
JENNIFER	GARFIELD	65,000.00
PERCY	GRANGER	34,500.00
VLADIMIR	HEAROWITZ	33,000.00
JAMES	JACOBI	55,000.00
JULIE	JENSEN	37,000.00
LARRY	LITERATA	37,500.00
KATHERINE	O'HEARN	42,500.00
RALPH	TYRO	20,000.00
HENRIETTA	HENDON	240,000.00
THEMIS	PAPAZEUS	100,000.00
JOHN	RUPEE	80,000.00
ROBBY	WILDER	90,000.00
		2,522,500.00

3.3 Modifying column widths

3.3.1 What you can do

You can control the number of characters or digits printed in a column.

3.3.2 How to do it

To place a specified number of characters or digits in a column, code:

1. A type 5 line for the field that is to be printed.
2. A **SZ=** expression indicating the number of characters or digits you want to print on the report. This expression follows the field name and precedes any column headings.

3.3.3 Demonstration

Objective: We are going to modify our report by:

- Printing the employee last name
- Printing the first initial of the employee's first name
- Decreasing the size of the SALARY column

Parameters

Parameters	Description
REC FIRST-NAME 5 10	FIRST-NAME 5 10 defines the first-name field separately from the last-name field to allow separate processing.
REC LAST-NAME 15 15	LAST-NAME 15 15 defines the last-name field separately from the first-name field to allow separate processing.
0251*010 LAST-NAME SZ=10 HH 'EMPLOYEE'	SZ=10 sets the column width to ten characters.
0251*020 FIRST-NAME SZ=1	SZ=1 sets the columnwidth to accommodate one character, which will truncate all but the first letter of the employee's first name.
0251*030 SALARY SZ=9 HH 'SALARY'	SZ=9 sets the column width for 9 numbers; without sizing, the column saves space for 13 digits plus commas and decimal points. When reducing the size, be sure to allow enough room for totals to print without truncation.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC FIRST-NAME    5 10
REC LAST-NAME     15 15
REC SALARY        160   5 3 DP=2
023 EMPLOYEE SALARY LISTING
0251*010 LAST-NAME SZ=10 HH 'EMPLOYEE'
0251*020 FIRST-NAME SZ=1
0251*030 SALARY  SZ=9   HH 'SALARY'

```

Result:

REPORT NO. 02	EMPLOYEE SALARY LISTING EMPLOYEE	09/24/99 PAGE	1	SALARY
BLOOMER	J			15,000.00
HUTTON	E			44,000.00
JENSON	R			82,000.00
KIMBALL	M			45,000.00
KING	D			14,500.00
NICEMAN	B			14,000.00
CRANE	H			75,000.00
FERNDALE	J			22,500.00
FONRAD	G			14,750.00
GARDNER	R			14,000.00
KAHALLY	D			20,000.00
KLWELLEN	T			43,000.00
KRAAMER	S			14,000.00
WILDER	R			90,000.00
				2,522,500.00

3.4 Formatting numeric data

3.4.1 What you can do

You can format numeric data automatically by using format codes supplied by CA-Culprit or by using your own patterns.

3.4.2 How to do it

Use a format code on the type 5 line. Useful format codes for printed output are:

- **FN** — Prints leading zeros and omits commas in numbers such as those used for employee identification.
- **FS** — Inserts dashes in social security numbers.
- **FD** — Inserts slashes in dates.
- **F\$** — Inserts dollar signs in monetary information.
- **FM** — Formats fields according to a specified pattern. The numeric field is symbolically represented as follows:
 - **9**s and **Z**s represent digits. The Z suppresses leading zeroes.
 - Periods (.) represent decimal points.
 - Special characters (for example, slashes or asterisks) are preceded by a printable digit.

3.4.3 Demonstration (1): Using formatting codes

Objective: This example uses codes for automatically formatting the employee identification, social security numbers, salaries, and employment dates.

Parameters

Parameters	Description
0251*020 ID FN HH 'ID'	FN prints leading zeros and omits commas.
0251*030 SS-NUMBER FS HH 'SOCIAL SECURITY' 'NUMBER'	FS inserts dashes into the social security number and suppresses automatic totaling.
0251*040 START-DATE FD HH 0251*050 END-DATE FD HH	FD inserts slashes into the start and end date and suppresses automatic totaling.
0251*060 SALARY F\$ HH	F\$ suppresses leading zeros and inserts a floating dollar sign.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC ID      1 4 2
REC EMPLOYEE 5 25
REC SS-NUMBER 84 9 2
REC START-DATE 93 6 2
REC END-DATE 99 6 2
REC SALARY   160 5 3 DP=2
023 PERSONNEL LISTING
0251*010 EMPLOYEE SZ=20 HH 'EMPLOYEE'
0251*020 ID      FN HH 'ID'
0251*030 SS-NUMBER FS HH 'SOCIAL SECURITY' 'NUMBER'
0251*040 START-DATE FD HH 'START' 'DATE'
0251*050 END-DATE FD HH 'END' 'DATE'
0251*060 SALARY   SZ=11 F$ HH 'SALARY'

```

Result:

REPORT NO. 02		PERSONNEL LISTING		09/24/99	PAGE 1	
EMPLOYEE	ID	SOCIAL SECURITY NUMBER		START DATE	END DATE	SALARY
JUNE BLOOMER	0069	039-55-7818		80/05/05	00/00/00	\$15,000.00
EDWARD HUTTON	0100	011-22-3333		77/09/07	00/00/00	\$44,000.00
RUPERT JENSON	0011	022-34-7891		80/09/29	00/00/00	\$82,000.00
MARIANNE KIMBALL	0067	022-77-8878		78/09/19	00/00/00	\$45,000.00
DORIS KING	0106	067-84-5516		80/08/16	00/00/00	\$14,500.00
BRIAN NICEMAN	0101	033-45-6110		80/05/06	00/00/00	\$14,000.00
HERBERT CRANE	0004	016-77-7451		77/05/14	00/00/00	\$75,000.00
JANE FERNDALE	0032	034-56-7891		79/09/09	00/00/00	\$22,500.00
GEORGE FONRAD	0045	092-34-8763		80/04/14	00/00/00	\$14,750.00
ROBIN GARDNER	0053	022-33-4444		81/06/15	00/00/00	\$14,000.00
DOUGLAS KAHALLY	0049	029-66-1234		79/09/29	00/00/00	\$20,000.00
TERENCE KLWELLEN	0016	010-20-1239		78/01/06	00/00/00	\$43,000.00

3.4.4 Demonstration (2): Using formatting patterns

Objective: The next sample report uses user-supplied patterns to format employee identification numbers with asterisks (*) and consecutive data fields with dashes.

Parameters

Parameters	Description
0251*020 ID FM '9*999' HH 'ID'	FM specifies that a user-supplied pattern formats the data. '9*999' is the format pattern.
0251*040 DATES FM '99/99/99-99/99/99' HH 'EMPLOY- MENT DATES'	FM specifies that a user-supplied pattern formats the data. '99/99/99-99/99/99' is the format pattern.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC ID      1 4 2
REC EMPLOYEE 5 25
REC SOC-SEC 84 9 2
REC DATES   93 12 2
REC SALARY   160 7 3 DP=2
013 PERSONNEL LISTING REPORT
0151*010 EMPLOYEE SZ=20          HH 'EMPLOYEE'
0151*020 ID      FM '9*999'     HH 'ID'
0151*030 SOC-SEC FS             HH 'SOCIAL SECURITY' 'NUMBER'
0151*040 DATES   FM '99/99/99-99/99/99' HH 'EMPLOYMENT DATES'
0151*050 SALARY   F$            HH 'SALARY'

```

Result:

REPORT NO. 01	PERSONNEL LISTING	REPORT	09/24/99 PAGE 1	
EMPLOYEE	ID	SOCIAL SECURITY NUMBER	EMPLOYMENT DATES	SALARY
JUNE BLOOMER	0*069	039-55-7818	80/05/05-00/00/00	\$.00
EDWARD HUTTON	0*100	011-22-3333	77/09/07-00/00/00	\$.00
RUPERT JENSON	0*011	022-34-7891	80/09/29-00/00/00	\$.00
MARIANNE KIMBALL	0*067	022-77-8878	78/09/19-00/00/00	\$.00
DORIS KING	0*106	067-84-5516	80/08/16-00/00/00	\$.00
BRIAN NICEMAN	0*101	033-45-6110	80/05/06-00/00/00	\$.00
HERBERT CRANE	0*004	016-77-7451	77/05/14-00/00/00	\$.00
JANE FERNDALE	0*032	034-56-7891	79/09/09-00/00/00	\$.00
GEORGE FONRAD	0*045	092-34-8763	80/04/14-00/00/00	\$.00
ROBIN GARDNER	0*053	022-33-4444	81/06/15-00/00/00	\$.00
DOUGLAS KAHALLY	0*049	029-66-1234	79/09/29-00/00/00	\$.00
TERENCE KLWELLEN	0*016	010-20-1239	78/01/06-00/00/00	\$.00

3.5 Adjusting line length and page depth

3.5.1 What you can do

You can specify line length and page depth by using **OUTPUT (OUT)** parameter options. When the options are not used, CA-Culprit automatically prints 132 characters to a line and 55 lines to a page, not including title lines and headings.

3.5.2 How to do it

OUTPUT parameter options that adjust report size are coded by:

- A numeral to specify the number of characters on a line
- **LP=** immediately followed by a numeral to specify the number of lines on a page

3.5.3 Demonstration

Objective: We are going to print a report with a 60-character line length and a 19-line page size by using OUTPUT parameter options.

The report is shown printed without the OUTPUT parameter specifications and again with the specifications.

Parameters

Parameters	Description
01OUT 60 LP=19	01 is the report number. OUT specifies the OUTPUT parameter. 60 specifies the number of characters to be printed on a report line. LP=19 specifies a page length of 19 lines.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE    5   25      'EMPLOYEE NAME'
REC SALARY     160   5  3  DP=2
013 EMPLOYEE SALARY LIST
01OUT 60 LP=19
0151*010 EMPLOYEE  HR
0151*020 SALARY    HF
```

Result BEFORE:

REPORT NO. 01	EMPLOYEE	EMPLOYEE SALARY LIST	01/28/99 PAGE	1
			SALARY	
	MICHAEL	ANGELO	18,000.00	
	DONTE	BANK	80,000.00	
	ALBERT	BREEZE	38,000.00	
	BETH M.	CLOUD	52,750.00	
	ALAN	DONOVAN	33,500.00	
	PERCY	EINSTEIN	34,500.00	
	JANE	FERNDALE	22,500.00	
	TOM	FITZHUGH	13,000.00	
	ROBIN	GARDNER	14,000.00	
	JENNIFER	GARFIELD	65,000.00	
	VLADIMIR	HEAROWITZ	33,000.00	
	EDWARD	HUTTON	44,000.00	
	JOCK	JACKSON	34,000.00	
	JAMES	JACOBI	55,000.00	
	JULIE	JANSSEN	37,000.00	
	RUPERT	JENSON	82,000.00	
	MARYLOU	JOHNSON	12.00	
	CYNTHIA	JOHNSON	13,500.00	
	DOUGLAS	KAHALLY	20,000.00	

Result AFTER:

REPORT NO. 01	EMPLOYEE	EMPLOYEE SALARY LIST	01/28/99 PAGE	1
			SALARY	
	MICHAEL	ANGELO	18,000.00	
	DONTE	BANK	80,000.00	
	ALBERT	BREEZE	38,000.00	
	BETH M.	CLOUD	52,750.00	
	ALAN	DONOVAN	33,500.00	
	PERCY	EINSTEIN	34,500.00	
	JANE	FERNDALE	22,500.00	
	TOM	FITZHUGH	13,000.00	
	ROBIN	GARDNER	14,000.00	
	JENNIFER	GARFIELD	65,000.00	
	VLADIMIR	HEAROWITZ	33,000.00	
	EDWARD	HUTTON	44,000.00	
	JOCK	JACKSON	34,000.00	
	JAMES	JACOBI	55,000.00	
	JULIE	JANSSEN	37,000.00	
	RUPERT	JENSON	82,000.00	
	MARYLOU	JOHNSON	12.00	
	CYNTHIA	JOHNSON	13,500.00	
	DOUGLAS	KAHALLY	20,000.00	

Chapter 4. Additional Standard Options

4.1	Multiple reports	4-4
4.1.1	What you can do	4-4
4.1.2	How to do it	4-4
4.1.3	Demonstration	4-4
4.2	Sequencing report output	4-8
4.2.1	What you can do	4-8
4.2.2	How to do it	4-8
4.2.3	Demonstration	4-8
4.3	Selective processing	4-10
4.3.1	What you can do	4-10
4.3.2	How to do it	4-10
4.3.3	Demonstration (1): Selecting records with single criteria	4-10
4.3.4	Demonstration (2): Selecting records with multiple criteria	4-11
4.4	Multiple detail lines	4-13
4.4.1	What you can do	4-13
4.4.2	How to do it	4-13
4.4.3	Demonstration	4-13

After you have created your basic CA-Culprit report, you can use some of the CA-Culprit optional capabilities to produce multiple reports in a single run, sequence report output, select data for processing, and specify multiple detail lines.

4.1 Multiple reports

4.1.1 What you can do

In a single CA-Culprit run, you can produce up to 100 reports that use the same data set.

4.1.2 How to do it

1. Assign a 2-digit report number to each report you want to create.
2. Code the report number on report-specific parameters.

4.1.3 Demonstration

Objective: We are going to produce three different reports in one run from the Commonwealth personnel file.

Parameters

Parameters	Description
01OUT 60 013 EMPLOYEE SALARIES 0151*010 EMPLOYEE SZ=20 HR 0151*020 SALARY SZ=11 F\$ HR	All parameters having 01 coded in columns 2 and 3 belong to report 01. OUT 60 specifies 60-character lines. EMPLOYEE SALARIES is the title. EMPLOYEE and SALARY are the two data fields printed for this report. Both columns have a specified width. SALARY is formatted with dollar signs.
02OUT 80 023 DEPARTMENT PERSONNEL LIST 0251*010 DEPARTMENT SZ=25 HR 0251*020 TITLE HR 0251*030 EMPLOYEE SZ=20 HR	All parameters having 02 coded in columns 2 and 3 belong to report 02. OUT 80 specifies 80-character lines. DEPARTMENT PERSONNEL LISTING is the title for report 02. Report 02 uses information from the DEPARTMENT, TITLE, and EMPLOYEE input fields.
03OUT 80 033 EMPLOYEE DEPARTMENTS 0351*010 EMPLOYEE SZ=20 HR 0351*020 DEPARTMENT SZ=25 HR	All parameters having 03 coded in columns 2 and 3 belong to report 03. OUT 80 specifies 80-character lines. EMPLOYEE DEPARTMENTS is the title for report 03. Report 03 uses information from the EMPLOYEE and DEPARTMENT input fields.

Complete code

col. 2
 ↓
 IN 200 F 400 PS(TAPE)
 REC EMPLOYEE 5 25 'EMPLOYEE NAME'
 REC DEPARTMENT 115 45 'DEPARTMENT'
 REC SALARY 160 5 3 DP=2 'ANNUAL SALARY'
 REC TITLE 171 20 'TITLE'
01OUT 60
013 EMPLOYEE SALARIES
0151*010 EMPLOYEE SZ=20 HR
0151*020 SALARY SZ=11 F\$ HR
02OUT 80
023 DEPARTMENT PERSONNEL LISTING
0251*010 DEPARTMENT SZ=25 HR
0251*020 TITLE HR
0251*030 EMPLOYEE SZ=20 HR
03OUT 80
033 EMPLOYEE DEPARTMENTS
0351*010 EMPLOYEE SZ=20 HR
0351*020 DEPARTMENT SZ=25 HR

Results

REPORT NO.	EMPLOYEE SALARIES	09/24/99 PAGE	1
	EMPLOYEE NAME	ANNUAL SALARY	
JUNE	BLOOMER	\$15,000.00	
EDWARD	HUTTON	\$44,000.00	
RUPERT	JENSON	\$82,000.00	
MARIANNE	KIMBALL	\$45,000.00	
DORIS	KING	\$14,500.00	
BRIAN	NICEMAN	\$14,000.00	
HERBERT	CRANE	\$75,000.00	
JANE	FERNDALE	\$22,500.00	
GEORGE	FONRAD	\$14,750.00	
ROBIN	GARDNER	\$14,000.00	
DOUGLAS	KAHALLY	\$20,000.00	
TERENCE	KLWELLEN	\$43,000.00	
ROBBY	WILDER	\$90,000.00	
			\$2,522,500.00

REPORT NO.	DEPARTMENT PERSONNEL LISTING	09/24/99 PAGE	1
	DEPARTMENT	TITLE	EMPLOYEE NAME
ACCOUNTING AND PAYROLL	PAYROLL CLERK	JUNE	BLOOMER
ACCOUNTING AND PAYROLL	FINANCIAL ANALYST	EDWARD	HUTTON
ACCOUNTING AND PAYROLL	MGR ACCTNG/PAYROLL	RUPERT	JENSON
ACCOUNTING AND PAYROLL	ACCOUNTANT	MARIANNE	KIMBALL
ACCOUNTING AND PAYROLL	AR CLERK	DORIS	KING
ACCOUNTING AND PAYROLL	AP CLERK	BRIAN	NICEMAN
COMPUTER OPERATIONS	MGR COMPUTER OPS	HERBERT	CRANE
COMPUTER OPERATIONS	COMPUTER OPERATOR	JANE	FERNDALE
COMPUTER OPERATIONS	DATA ENTRY CLERK	GEORGE	FONRAD
COMPUTER OPERATIONS	DATA ENTRY CLERK	ROBIN	GARDNER
COMPUTER OPERATIONS	COMPUTER OPERATOR	DOUGLAS	KAHALLY
COMPUTER OPERATIONS	SYSTEMS PROGRAMMER	TERENCE	KLWELLEN
EXECUTIVE ADMINISTRATION	DIR CORP CONFUSION	ROBBY	WILDER

REPORT NO. 03	EMPLOYEE NAME	EMPLOYEE DEPARTMENTS	09/24/99 PAGE	1
			DEPARTMENT	
JUNE	BLOOMER	ACCOUNTING AND PAYROLL		
EDWARD	HUTTON	ACCOUNTING AND PAYROLL		
RUPERT	JENSON	ACCOUNTING AND PAYROLL		
MARIANNE	KIMBALL	ACCOUNTING AND PAYROLL		
DORIS	KING	ACCOUNTING AND PAYROLL		
BRIAN	NICEMAN	ACCOUNTING AND PAYROLL		
HERBERT	CRANE	COMPUTER OPERATIONS		
JANE	FERNDALE	COMPUTER OPERATIONS		
GEORGE	FONRAD	COMPUTER OPERATIONS		
ROBIN	GARDNER	COMPUTER OPERATIONS		
DOUGLAS	KAHALLY	COMPUTER OPERATIONS		
TERENCE	KLWELLEN	COMPUTER OPERATIONS		
ROBBY	WILDER	EXECUTIVE ADMINISTRATION		

4.2 Sequencing report output

4.2.1 What you can do

You can sequence alphanumeric and numeric data in your report in ascending or descending order.

4.2.2 How to do it

Code the **SORT** parameter line with the following information, from left to right:

1. The 2-digit report number.
2. The word **SORT**. There is no space after the report number.
3. The name of the field to be sorted.
4. The sort order:
 - **Blank** or **A** for ascending order
 - **D** for descending order
5. Repeat steps 3 and 4 for any additional sort fields, separating entries by at least one space.

4.2.3 Demonstration

Objective: The next report lists employees in alphabetical order by performing a sort on employee last name.

Parameters

Parameters	Description
01SORT LAST-NAME	01 is the report number. The SORT parameter is coded in columns 4 to 8, immediately after the report number. LAST-NAME is the sort field. This field is also defined to the report by appearing on the REC parameter.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME     15   15
REC SALARY        160   5   3  DP=2
013 EMPLOYEE SALARY LISTING
01SORT LAST-NAME
0151*010 EMPLOYEE SZ=20  HH 'EMPLOYEE'
0151*020 SALARY       HH 'SALARY'

```

Result:

REPORT NO. 01	EMPLOYEE	EMPLOYEE SALARY LIST	01/28/99 PAGE	1
			SALARY	
MICHAEL	ANGELO		18,000.00	
DONTE	BANK		80,000.00	
ALBERT	BREEZE		38,000.00	
BETH M.	CLOUD		52,750.00	
ALAN	DONOVAN		33,500.00	
PERCY	EINSTEIN		34,500.00	
JANE	FERNDALE		22,500.00	
TOM	FITZHUGH		13,000.00	
ROBIN	GARDNER		14,000.00	
JENNIFER	GARFIELD		65,000.00	
VLADIMIR	HEAROWITZ		33,000.00	
EDWARD	HUTTON		44,000.00	
JOCK	JACKSON		34,000.00	
JAMES	JACOBI		55,000.00	
JULIE	JANSSEN		37,000.00	
RUPERT	JENSON		82,000.00	
MARYLOU	JOHNSON		12.00	
CYNTHIA	JOHNSON		13,500.00	
DOUGLAS	KAHALLY		20,000.00	

4.3 Selective processing

4.3.1 What you can do

You can refine the scope of your processing by choosing only those records that meet certain criteria.

4.3.2 How to do it

You can use either a **SELECT** or **BYPASS** parameter. The choice depends on which option allows for the clearest statement of the condition controlling the choice. You can specify more than one selection criterion on a single SELECT or BYPASS parameter. However, the two parameters cannot be applied together.

To use SELECT or BYPASS:

1. Code the SELECT or BYPASS parameter immediately after the REC parameters that describe the file.
2. Enter the data item and test condition that the data must meet; such as, **EQ** (equal to), **GT** (greater than), **LT** (less than).
3. Join multiple criteria with logical operators:
 - **AND**, to indicate that all conditions must be true
 - **OR**, to indicate that only one condition need be true

4.3.3 Demonstration (1): Selecting records with single criteria

Objective: We are going to modify our report by listing only employees earning \$50,000.00 and over.

The following code selects the appropriate records:

Parameters

Parameters	Description
SEL SALARY GE 50000.00	SEL SALARY GE 50000.00 specifies selection of salaries equal to or greater than 50,000.00.
or	
BYP SALARY LT 50000.00	BYP SALARY LT 50000.00 specifies omission of salaries less than 50,000.00.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME     15  15
REC SALARY        160  5   3  DP=2
SEL SALARY GE 50000.00
013 EMPLOYEE SALARY LISTING
01SORT LAST-NAME
0151*010 EMPLOYEE    HH 'EMPLOYEE'
0151*020 SALARY SZ=10 HH 'SALARY'

```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING EMPLOYEE	09/24/99 PAGE	1	SALARY
MONTE	BANK			80,000.00
BETH	CLOUD			52,750.00
HERBERT	CRANE			75,000.00
JENNIFER	GARFIELD			65,000.00
HENRIETTA	HENDON			240,000.00
JAMES	JACOBI			55,000.00
RUPERT	JENSON			82,000.00
BURT	LANCHESTER			54,500.00
RENE	MAKER			85,000.00
DANIEL	MOON			72,000.00
THEMIS	PAPAZEUS			100,000.00
ELEANOR	PEOPLES			80,000.00
JOHN	RUPEE			80,000.00
ROGER	WILCO			80,000.00
ROBBY	WILDER			90,000.00
				1,291,250.00

4.3.4 Demonstration (2): Selecting records with multiple criteria

Objective: Our next report selects employees earning \$15,000 or less and those earning between \$35,000 and \$50,000.

Parameters

Parameters	Description
SEL SALARY EQ (35000 TO 50000) OR SALARY LE 15000	SEL SALARY EQ (35000 TO 50000) specifies the first selection criteria. OR relates the first test to the second by allowing selection of a record that meets only one of the specified conditions. SALARY LE 15000 specifies the alternative record selection criteria.

The following BYPASS parameter achieves the same results:

Parameters

Parameters	Description
BYP SALARY EQ (15000.01 TO 34999.99) OR SALARY GT 50000	BYP SALARY EQ (15000.01 TO 34999.99) specifies the first record bypass criteria. OR relates the first test to the second by allowing bypass of a record that meets only one of the specified conditions. SALARY GT 50000 specifies the alternative record bypass criteria.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME     15   15
REC SALARY        160   5 3 DP=2
SEL SALARY EQ (35000 TO 50000) OR SALARY LE 15000
013 EMPLOYEE SALARY LISTING
01SORT LAST-NAME
0151*010 EMPLOYEE    HH 'EMPLOYEE'
0151*020 SALARY SZ=10 HH 'SALARY'
```

Result:

REPORT NO. 01	EMPLOYEE	SALARY LISTING	09/24/99 PAGE	1	SALARY
	EMPLOYEE				
	HARRY	ARM			46,000.00
	JUNE	BLOOMER			15,000.00
	CHARLES	BOWER			38,500.00
	C.	BREEZE			38,000.00
	TERRY	CLOTH			38,000.00
	CAROLYN	CROW			37,500.00
	PHINEAS	FINN			45,000.00
	TOM	FITZHUGH			13,000.00
	GEORGE	FONRAD			14,750.00
	ROBIN	GARDNER			14,000.00
	EDWARD	HUTTON			44,000.00
	JULIE	JENSEN			37,000.00
	CYNTHIA	JOHNSON			13,500.00
	MARIANNE	KIMBALL			45,000.00
	DORIS	KING			14,500.00
	TERENCE	KLWELLEN			43,000.00
	SANDY	KRAAMER			14,000.00
	LARRY	LITERATA			37,500.00
	RICHARD	MUNYON			36,000.00
	BRIAN	NICEMAN			14,000.00
	KATHERINE	O'HEARN			42,500.00
	MADELINE	ORGRATZI			39,000.00
	LAURA	PENMAN			39,000.00
	NANCY	TERNER			13,000.00
	RICHARD	WAGNER			47,000.00
	BETSY	ZEDI			37,000.00
					815,750.00

4.4 Multiple detail lines

4.4.1 What you can do

Up to this point we have used the type 5 parameter to assign several pieces of information to a single print line. In CA-Culprit this is called a **detail line**. For example,

```
MAX DELANO 12 ORCHARD LANE LIPTON NJ 07080
```

You can also assign information to multiple detail lines that print as a single block of information. For example,

```
MAX DELANO      ◀ first detail line  
12 ORCHARD LANE   ◀ second detail line  
LIPTON NJ 07080  ◀ third detail line
```

4.4.2 How to do it

To specify multiple detail lines on a type 5 parameter, change the number in *column 5* to indicate the line on which the information should appear:

0151*010	NAME	◀ <u>first detail line</u>
0152*010	STREET-ADDRESS	◀ <u>second detail line</u>
0153*010	TOWN	◀ <u>third detail line</u>
0153*015	STATE	
0153*020	ZIPCODE	

4.4.3 Demonstration

Objective: Our next report uses multiple detail lines to produce an employee address list. The employee name, street number, city, state, and zip code appear on three separate lines.

Parameters

Parameters	Description
0151*010 LAST-NAME 0151*012 FIRST-NAME	1 specifies placement of the last name and first name on the first detail line.
0152*020 STREET	2 specifies placement of the street on the second detail line.
0153*020 CITY 0153*025 STATE 0153*030 ZIP-CODE	3 specifies placement of the city, state, and zip code on the third detail line.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC FIRST-NAME      5   10
REC LAST-NAME       15   15
REC STREET          30   20
REC CITY            50   15
REC STATE           65    2
REC ZIP-CODE        67    5
01OUT 80
013 EMPLOYEE ADDRESS LIST
01SORT LAST-NAME
0151*010 LAST-NAME
0151*012 FIRST-NAME
0152*020 STREET
0153*020 CITY   SZ=10
0153*025 STATE
0153*030 ZIP-CODE
```

Result:

REPORT NO. 01 ANGELO	MICHAEL	EMPLOYEE ADDRESS LIST	09/29/99 PAGE	1
BABBIT	HERBIE	507 CISTINE DR WELLESLEY	MA	01568
BANK	DONTE	30 HERON AVE KINGSTON	NJ	21341
BLOOMER	DUDY	45 EAST GROVE DR HANIBAL	MA	02415
BREEZE	ALBERT	14 ZITHER TERR LEXINGTON	MA	01675
		100 BOARDWALK OCEAN CITY	NJ	03461

Chapter 5. Adding Programming Logic to a Report

5.1 How to code programming logic	5-4
5.2 Using counters	5-5
5.2.1 What you can do	5-5
5.2.2 How to do it	5-5
5.2.3 Demonstration (1): Printing the count of each occurrence	5-6
5.2.4 Demonstration (2): Printing a total count	5-7
5.2.5 Demonstration (3): Using the value of a work field	5-8
5.3 Performing arithmetic operations	5-9
5.3.1 What you can do	5-9
5.3.2 How to do it	5-9
5.3.3 Demonstration	5-9
5.4 Directing processing flow	5-12
5.4.1 What you can do	5-12
5.4.2 How to do it	5-12
5.4.3 Demonstration	5-12

Special CA-Culprit instructions allow you to produce reports that are more sophisticated than the formatted file listings we have done so far. You can count specific items, perform arithmetic operations, test data against specified conditions, and direct the flow of processing.

5.1 How to code programming logic

We are going to process data as it is read from the input file. To do this, we use **type 7** parameters on which to code our logic.

Type 7 parameters are coded from left to right, as follows:

1. A 2-digit report number
2. The parameter type (**7**)
3. A 3-digit sequence number indicating the order in which the parameter is executed
4. The instruction that will be executed

5.2 Using counters

5.2.1 What you can do

You can count occurrences, or the number of times an event happens during processing.

5.2.2 How to do it

- Conditionally, by coding:

1. A numeric work field:

010 COUNTER-A

2. A computation that adds 1 to the value of the work field whenever counting should occur:

017100 COUNTER-A + 1 COUNTER-A

- Automatically, by coding:

1. A numeric work field.

010 COUNTER-B

2. A value you want assigned each time counting occurs. Typically, this value is 1:

010 COUNTER-B 1

3. The name of the work field on a type 5 parameter:

0151*010 COUNTER-B (Prints the value each time)

or

0151*000 COUNTER-B (Does not print the value)

4. The name of the work field on a type 6 parameter when the total count should be printed on the report.

0161*010 COUNTER-B

Type 6 parameters are coded the same way as type 5.

►► For more information see Chapter 6, “Controlling Total Lines” on page 6-1.

5.2.3 Demonstration (1): Printing the count of each occurrence

Objective: This report assigns a number to each employee in the public relations department by adding 1 to the value of a numeric work field. The value of the work field is printed out each time a new employee is processed.

Parameters

Parameters	Description
01OUT D	01OUT D suppresses the automatic total of the employee number column.
010 COUNT	010 COUNT establishes the numeric work field COUNT.
0151*030 COUNT	0151*030 COUNT prints the value of the work field COUNT.
017010 COUNT + 1 COUNT	017010 COUNT + 1 COUNT adds 1 to the value contained in the work field COUNT.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   20
REC DEPARTMENT    115  25
SEL DEPARTMENT EQ 'PUBLIC RELATIONS'
013 EMPLOYEE COUNT
01OUT D
010 COUNT
0151*010 DEPARTMENT      HH 'DEPARTMENT'
0151*020 EMPLOYEE       HH 'EMPLOYEE NAME'
0151*030 COUNT SZ=3   HH 'EMPLOYEE NUMBER'
017010 COUNT + 1 COUNT

```

Result:

REPORT NO. 01 DEPARTMENT	EMPLOYEE COUNT EMPLOYEE NAME	11/13/99 PAGE 1	EMPLOYEE NUMBER
PUBLIC RELATIONS	MICHAEL ANGELO		1
PUBLIC RELATIONS	DONTE BANK		2
PUBLIC RELATIONS	JOCK JACKSON		3
PUBLIC RELATIONS	CAROL McDougall		4
PUBLIC RELATIONS	LAURA PENMAN		5
PUBLIC RELATIONS	BETSY ZEDI		6

5.2.4 Demonstration (2): Printing a total count

Objective: This report counts the number of employees processed by adding 1 to the value of the numeric work field. The final value of the work field is printed out by using a type 6 parameter.

- See Chapter 6, “Controlling Total Lines” on page 6-1 for details about type 6 parameters.

Parameters

Parameters	Description
010 COUNT	010 COUNT establishes the numeric work field.
0161*040 'NUMBER OF EMPLOYEES' 0161*050 COUNT	0161*040 and 0161*050 are type 6 parameters used to label the print line and obtain the last value in the work field COUNT.
017010 COUNT + 1 COUNT	017010 COUNT + 1 COUNT adds 1 to the value contained in the work field COUNT.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   20
REC DEPARTMENT    115  25
SEL DEPARTMENT EQ 'PUBLIC RELATIONS'
013 EMPLOYEE COUNT
010 COUNT
0151*010 DEPARTMENT     HH 'DEPARTMENT'
0151*020 EMPLOYEE       HH 'EMPLOYEE NAME'
0161*040 'NUMBER OF EMPLOYEES'
0161*050 COUNT
017010 COUNT + 1 COUNT

```

Result:

REPORT NO. 01 DEPARTMENT	EMPLOYEE COUNT EMPLOYEE NAME	11/13/99 PAGE 1
PUBLIC RELATIONS	MICHAEL ANGELO	
PUBLIC RELATIONS	DONTE BANK	
PUBLIC RELATIONS	JOCK JACKSON	
PUBLIC RELATIONS	CAROL McDougall	
PUBLIC RELATIONS	LAURA PENMAN	
PUBLIC RELATIONS	BETSY ZEDI	
	NUMBER OF EMPLOYEES	6

5.2.5 Demonstration (3): Using the value of a work field

Objective: This report counts the number of employees processed by adding 1 to the value of the numeric work field. The final value of the work field is printed out by using a type 6 parameter. See Chapter 6, “Controlling Total Lines” on page 6-1 for details about type 6 parameters.

Parameters

Parameters	Description
010 COUNT 1	010 COUNT 1 establishes a numeric work field having the value of 1.
0151*000 COUNT	0151 identifies the work field COUNT on a type 5 parameter. An automatic running total is kept. 000 suppresses the printing of the field value (1) on the report. (More than one such type 5 parameter can be used, if needed.)
0161*040 'NUMBER OF EMPLOYEES' 0161*050 COUNT	0161*040 and 0161*050 are type 6 parameters used to label the print line and obtain the last value in the work field COUNT.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   20
REC DEPARTMENT    115  25
SEL DEPARTMENT EQ 'PUBLIC RELATIONS'
013 EMPLOYEE COUNT
010 COUNT 1
0151*000 COUNT
0151*010 DEPARTMENT      HH 'DEPARTMENT'
0151*020 EMPLOYEE       HH 'EMPLOYEE NAME'
0161*040 'NUMBER OF EMPLOYEES'
0161*050 COUNT

```

Result:

REPORT NO. 01 DEPARTMENT	EMPLOYEE COUNT EMPLOYEE NAME	11/13/99 PAGE 1
PUBLIC RELATIONS	MICHAEL ANGELO	
PUBLIC RELATIONS	DONTE BANK	
PUBLIC RELATIONS	JOCK JACKSON	
PUBLIC RELATIONS	CAROL McDougall	
PUBLIC RELATIONS	LAURA PENMAN	
PUBLIC RELATIONS	BETSY ZEDI	
	NUMBER OF EMPLOYEES	6

5.3 Performing arithmetic operations

5.3.1 What you can do

You can perform arithmetic operations in your CA-Culprit report.

5.3.2 How to do it

For **simple, one-step operations**, code the following:

1. The numeric field being used as the operand
2. An operator that must have a space on either side
3. The work field that is to contain the results

```
017010 YR-SALARY / 52 WK-SALARY
```

For **complex (two or more operations) statements**, code the following:

1. The word **COMPUTE**

```
017010 COMPUTE (SALARY + 2000) X 0.10 BONUS
```

Arithmetic operations can be used

- Alone:

```
017100 SALARY / 52 WK-SALARY
```

- As the result of a conditional statement when the condition is met:

```
017010 IF SALARY LE 19000 100
```

```
017020 ...
```

```
017100 SALARY / 52 WK-SALARY
```

5.3.3 Demonstration

Objective: This report compares employee annual earnings to a base of \$19,000. Employees earning more than the base amount are considered salaried; those earning less are considered hourly. The report calculates an hourly wage for all nonsalaried employees.

Testing and calculations for this report are on type 7 parameters.

Parameters

Parameters	Description
01OUT D	D suppresses totals
010 AMOUNT DP=2	AMOUNT is a work field that holds computation results containing two decimal places.
017010 IF SALARY LE 19000 500	IF SALARY LE 19000 500 is a conditional statement that tests for a range less than or equal to 19000. If the test is true, processing skips to line 500 of the type 7 parameters. If not true, the type 7 statements are processed in sequence until TAKE is executed.
017 MOVE 'SALARIED' TO MESSAGE	MOVE 'SALARIED' TO MESSAGE places SALARIED in the MESSAGE work field when SALARY is greater than 19000.
017 MOVE SALARY TO AMOUNT	MOVE SALARY TO AMOUNT places the salary amount in the work field.
017 TAKE	TAKE causes all the type 5 parameter fields to be written to a work file (the extract file) for use in later processing.
017500 MOVE 'HOURLY ' TO MESSAGE	MOVE 'HOURLY ' TO MESSAGE places the word HOURS in the MESSAGE work field when the salary is less than 19000.
017 COMPUTE (SALARY / 52) / 40 AMOUNT	COMPUTE specifies a two-step computation. The result is placed in the AMOUNT work field.
TAKE	This is an implied TAKE that causes all type 5 fields to be written to a work file (the extract file) for use in later processing. This TAKE is supplied by CA-Culprit; it is not required in the code.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME    15   15
REC DEPARTMENT   115   25
REC SALARY       160   5   3   DP=2
01OUT D
01SORT LAST-NAME
013 EMPLOYEE COMPENSATION STATUS
010 MESSAGE '
010 AMOUNT DP=2
0151*010 EMPLOYEE SZ=20     HH 'EMPLOYEE' 'NAME'
0151*020 MESSAGE          HH 'STATUS'
0151*030 AMOUNT SZ=10  F$  HH 'SALARY'
017010 IF SALARY LE 19000 500
017    MOVE 'SALARIED' TO MESSAGE
017    MOVE SALARY TO AMOUNT
017    TAKE
017500 MOVE 'HOURLY' TO MESSAGE
017    COMPUTE (SALARY / 52) / 40 AMOUNT

```

Result:

REPORT NO. 01	EMPLOYEE	COMPENSATION STATUS	09/24/99 PAGE	1
	NAME		STATUS	SALARY
	ROY	ANDALE	SALARIED	\$33,500.00
	MICHAEL	ANGELO	HOURLY	\$8.65
	HARRY	ARM	SALARIED	\$46,000.00
	MONTE	BANK	SALARIED	\$80,000.00
	JUNE	BLOOMER	HOURLY	\$7.21
	CHARLES	BOWER	SALARIED	\$38,500.00
	C.	BREEZE	SALARIED	\$38,000.00
	RALPH	TYRO	SALARIED	\$20,000.00
	RICHARD	WAGNER	SALARIED	\$47,000.00
	ROGER	WILCO	SALARIED	\$80,000.00
	ROBBY	WILDER	SALARIED	\$90,000.00
	BETSY	ZEDI	SALARIED	\$37,000.00

5.4 Directing processing flow

5.4.1 What you can do

You can code a repeated procedure just once and direct CA-Culprit to repeat it whenever needed.

5.4.2 How to do it

To direct the processing flow, code:

- The word **PERFORM** on a processing parameter (type 7 or type 8). Type 8 parameters are discussed in Chapter 6, “Controlling Total Lines” on page 6-1.
- The parameter sequence number where processing of the repetitive statements starts.
- The word **RETURN** at the end of the block of repetitive statements. Keep the related PERFORM/RETURN within the same parameter type.

5.4.3 Demonstration

Objective: This report calculates employee bonuses based upon individual salary and longevity.

The repetitive computation based on longevity is executed by using PERFORM/RETURN statements.

Parameters

Parameters	Description
017 IF SALARY GT 15000 100 017 COMPUTE SALARY X 0.02 BONUS 017 PERFORM 500	100 directs processing to sequence line 100 when SALARY is greater than \$15,000. PERFORM 500 directs processing to sequence line 500 to calculate the longevity factor.
017 IF BONUS LT 500 010 017 MOVE 500 TO BONUS 017 010 TAKE	010 directs processing to sequence line 010 if the calculated bonus is less than \$500. TAKE causes the type 5 line to be extracted.
017100 IF SALARY GT 25000 200 017 COMPUTE SALARY X 0.03 BONUS 017 PERFORM 500	200 directs processing to sequence line 200 when SALARY is greater than \$25,000. PERFORM 500 directs processing to sequence line 500 to calculate the longevity factor.
017 IF BONUS LT 1200 110 017 MOVE 1200 TO BONUS 017 110 TAKE	110 directs processing to sequence line 110 if the calculated bonus is less than \$1200. TAKE causes the type 5 line to be extracted.
017 PERFORM 500	PERFORM 500 directs processing to sequence line 500 to calculate the longevity factor.
017 500 IF START-YEAR LT 81 510 017 BONUS + 100 BONUS 017 RETURN 017510 BONUS + 300 BONUS 017 520 RETURN	500 tests START-YEAR. Employees who started before 1981 receive an additional \$300 bonus. Those starting after 1981 receive an additional \$100 bonus. 017 520 RETURN directs processing back to the statement following the last PERFORM statement executed.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME    15   15
REC START-YEAR   97   2   2
REC SALARY       160   5   3   DP=2
01SORT LAST-NAME
013 EMPLOYEE BONUS REPORT
010 BONUS DP=2
0151*010 EMPLOYEE SZ=20   HH 'EMPLOYEE' 'NAME'
0151*020 SALARY   SZ=10   F$ HH '   SALARY'
0151*030 BONUS    SZ=7   F$ HH ' BONUS'
017   IF SALARY GT 15000 100
017   COMPUTE SALARY X 0.02  BONUS
017   PERFORM 500
017   IF BONUS LT 500 010
017   MOVE 500 TO BONUS
017010 TAKE
017100 IF SALARY GT 25000 200
017   COMPUTE SALARY X 0.03  BONUS
017   PERFORM 500
017   IF BONUS LT 1200 110
017   MOVE 1200 TO BONUS
017110 TAKE
017200 COMPUTE SALARY X 0.04  BONUS
017   PERFORM 500
017   IF BONUS LT 1800 210
017   MOVE 1800 TO BONUS
017210 TAKE
017500 IF START-YEAR LT 81 510
017   BONUS + 100 BONUS
017   RETURN
017510 BONUS + 300 BONUS
017520 RETURN

```

Result:

REPORT NO. 01	EMPLOYEE BONUS REPORT	09/24/99 PAGE	1
	EMPLOYEE NAME	SALARY	BONUS
ROY	ANDALE	\$33,500.00	\$1,640.00
MICHAEL	ANGELO	\$18,000.00	\$840.00
HARRY	ARM	\$46,000.00	\$1,800.00
MONTE	BANK	\$80,000.00	\$1,800.00
JUNE	BLOOMER	\$15,000.00	\$500.00
CHARLES	BOWER	\$38,500.00	\$1,800.00
C.	BREEZE	\$38,000.00	\$1,800.00
RALPH	TYRO	\$20,000.00	\$900.00
RICHARD	WAGNER	\$47,000.00	\$1,800.00
ROGER	WILCO	\$80,000.00	\$1,800.00
ROBBY	WILDER	\$90,000.00	\$1,800.00
BETSY	ZEDI	\$37,000.00	\$1,780.00
			\$2,522,500.00
			\$81,950.00

Chapter 6. Controlling Total Lines

6.1	The parameters to use with totals	6-4
6.2	Selectively printing total amounts	6-5
6.2.1	What you can do	6-5
6.2.2	How to do it	6-5
6.2.3	Demonstration	6-5
6.3	Labeling total lines	6-7
6.3.1	What you can do	6-7
6.3.2	How to do it	6-7
6.3.3	Demonstration	6-7
6.4	Generating total amounts only	6-9
6.4.1	What you can do	6-9
6.4.2	How to do it	6-9
6.4.3	Demonstration	6-9
6.5	Suppressing all totals	6-11
6.5.1	What you can do	6-11
6.5.2	How to do it	6-11
6.5.3	Demonstration	6-11
6.6	Generating subtotals	6-13
6.6.1	What you can do	6-13
6.6.2	How to do it	6-13
6.6.3	Demonstration	6-13
6.7	Modifying subtotal and total line placement	6-16
6.7.1	What you can do	6-16
6.7.2	How to do it	6-16
6.7.3	Demonstration	6-16
6.8	Selectively printing output lines	6-18
6.8.1	What you can do	6-18
6.8.2	How to do it	6-18
6.8.3	Demonstration (1): Selecting totals output	6-18
6.8.4	Demonstration (2): Using a work field to print total line labels	6-21
6.9	Obtaining sort field values	6-24
6.9.1	What you can do	6-24
6.9.2	How to do it	6-24
6.9.3	Demonstration	6-24
6.10	Obtaining sorted work field values	6-26
6.10.1	What you can do	6-26
6.10.2	How to do it	6-26
6.10.3	Demonstraton	6-26

CA-Culprit generates automatic total amounts for numeric fields that appear on type 5 lines. You can label and control the printing of the total lines by using parameters that act upon data before it is printed on the report (the output phase).

6.1 The parameters to use with totals

When working with total amounts, use:

1. **Type 6** parameters to control the print lines.
2. An OUT parameter with the **T** option to print total amounts only. (Use the **D** option to print detail lines only.)
3. A **SORT** parameter to obtain subtotals.
4. **Type 8** parameters to process the data.

6.2 Selectively printing total amounts

6.2.1 What you can do

You can print only the totals you want by using **type 6** parameters.

6.2.2 How to do it

Coding type 6 parameters is similar to coding type 5 parameters. That is, enter the following information, from left to right:

1. The 2-digit report number
2. The parameter type (**6**)
3. The number of the total line
4. The column placement consisting of either an asterisk (*) followed by a 3-digit number for relative placement or a 4-digit number for specific placement
5. The spacing specification
6. The name of the field you want to print on the total line
7. Additional options such as size, number of decimal places, and special formatting

6.2.3 Demonstration

Objective: This report contains the numeric fields START-YEAR, BIRTH-YEAR, and SALARY. Only the total for SALARY is printed.

Parameters

Parameters	Description
0161*0550SALARY SZ=11 F\$	01 is the number of the report. 6 specifies a type 6 parameter. 1 specifies the first total line. *055 specifies automatic page formatting with the salary total printed under the list of individual salaries. 0 specifies a blank line before the total is printed. SALARY specifies the field to be totaled. SZ=11 F\$ specifies the size of the total field to avoid truncation. The total is formatted with a dollar sign.

6.2 Selectively printing total amounts

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC FIRST-NAME    5   10
REC LAST-NAME     15  15
REC START-YEAR    97  2   2
REC BIRTH-YEAR    109  2   2
REC DEPARTMENT    115  25
REC SALARY        160  5   3  DP=2
013 EMPLOYEE SALARY LISTING BY DEPARTMENT
01SORT DEPARTMENT Ø SALARY D EMPLOYEE
0151*010 DEPARTMENT    HH 'DEPARTMENT'
0151*020 LAST-NAME SZ=10 HH 'EMPLOYEE NAME'
0151*030 FIRST-NAME
0151*040 BIRTH-YEAR    HH 'BIRTH' 'YEAR'
0151*050 START-YEAR    HH 'START' 'YEAR'
0151*055 SALARY SZ=11 F$  HH 'ANNUAL SALARY'
0161*0550SALARY SZ=11 F$
```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING BY DEPARTMENT	09/24/99	PAGE	1	BIRTH YEAR	START YEAR	ANNUAL SALARY
DEPARTMENT	EMPLOYEE NAME						
ACCOUNTING AND PAYROLL	JENSON	RUPERT	48	29			\$82,000.00
ACCOUNTING AND PAYROLL	KIMBALL	MARIANNE	49	19			\$45,000.00
ACCOUNTING AND PAYROLL	HUTTON	EDWARD	41	7			\$44,000.00
ACCOUNTING AND PAYROLL	BLOOMER	JUNE	60	5			\$15,000.00
ACCOUNTING AND PAYROLL	KING	DORIS	60	16			\$14,500.00
ACCOUNTING AND PAYROLL	NICEMAN	BRIAN	55	6			\$14,000.00
							\$214,500.00
THERMOREGULATION	CLOTH	TERRY	45	2			\$38,000.00
THERMOREGULATION	TIME	MARK	58	5			\$33,000.00
THERMOREGULATION	KASPAR	JOE	40	5			\$31,000.00
							\$227,000.00
							\$2,522,500.00

6.3 Labeling total lines

6.3.1 What you can do

You can label each printed total by using a type 6 parameter.

6.3.2 How to do it

To place a label on a total line:

1. Code a type 6 parameter.
2. Enter the text of the label instead of a field name.
3. Enclose the text in single quotation marks.

6.3.3 Demonstration

Objective: In the next report, a label is added to the salary grand total.

Parameters

Parameters	Description
0161*0400'SALARY TOTAL='	01 specifies report 01. 6 specifies a type 6 parameter. 1 specifies the first total line. *040 specifies automatic page formatting with the label starting to the left of the grand total. 0 specifies a blank line before the total line. 'SALARY TOTAL=' specifies the label used on the total line.

Complete code

6.3 Labeling total lines

```

IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC FIRST-NAME    5   10
REC LAST-NAME     15  15
REC START-YEAR    97   2   2
REC BIRTH-YEAR    109  2   2
REC DEPARTMENT    115  25
REC SALARY        160  5   3   DP=2
013 EMPLOYEE SALARY LISTING
0151*010 DEPARTMENT      HH 'DEPARTMENT'
0151*020 LAST-NAME SZ=10 HH '                           EMPLOYEE NAME'
0151*030 FIRST-NAME
0151*040 BIRTH-YEAR      HH 'BIRTH' 'YEAR'
0151*050 START-YEAR      HH 'START' 'YEAR'
0151*055 SALARY SZ=9 F$  HH '      ANNUAL SALARY'
0161*0400'SALARY TOTAL='
0161*055 SALARY SZ=9 F$

```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING	09/24/99 PAGE	1	BIRTH YEAR	START YEAR	ANNUAL SALARY
DEPARTMENT	EMPLOYEE NAME					
ACCOUNTING AND PAYROLL	BLOOMER	JUNE	60	5	\$15,000.00	
ACCOUNTING AND PAYROLL	HUTTON	EDWARD	41	7	\$44,000.00	
ACCOUNTING AND PAYROLL	JENSON	RUPERT	48	29	\$82,000.00	
INTERNAL SOFTWARE	LITERATA	LARRY	55	9	\$37,500.00	
INTERNAL SOFTWARE	O'HEARN	KATHERINE	54	4	\$42,500.00	
INTERNAL SOFTWARE	TYRO	RALPH	55	21	\$20,000.00	
EXECUTIVE ADMINISTRATION	HENDON	HENRIETTA	33	21	\$240,000.00	
EXECUTIVE ADMINISTRATION	PAPAZEUS	THEMIS	35	7	\$100,000.00	
EXECUTIVE ADMINISTRATION	RUPEE	JOHN	33	23	\$80,000.00	
SALARY TOTAL=						\$2,522,500.00

6.4 Generating total amounts only

6.4.1 What you can do

You can produce a report that contains only totals of numeric fields that appear on type 5 lines by specifying a **totals-only** report on the OUTPUT parameter.

6.4.2 How to do it

To obtain a totals-only report, code:

1. The OUTPUT parameter with the totals-only option (OUT T)
2. A type 5 parameter for each numeric field to be totaled
3. A type 6 parameter for totals to be printed

6.4.3 Demonstration

Objective: This example produces the total of all salaries in the Commonwealth Corporation.

Parameters

Parameters	Description
01OUT T	T specifies totals only.
0151*020 SALARY	0151*020 SALARY specifies the type 5 parameter that retrieves and causes the individual salaries to be totaled.
0161*010 'ALL DEPARTMENTS' 0161*020 SALARY	6 identifies the parameter type. 'ALL DEPARTMENTS' specifies the label for the total line.
	SALARY specifies the field whose total is to be printed.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC SALARY      160    5 3 DP=2  'SALARY' 'TOTAL'
01OUT T
0151*020 SALARY
0161*010 'ALL DEPARTMENTS' HH 'DEPARTMENT'
0161*020 SALARY SZ=10   HH 'TOTAL SALARIES'
```

6.4 Generating total amounts only

Result:

DEPARTMENT	TOTAL SALARIES
ALL DEPARTMENTS	2,522,500.00

6.5 Suppressing all totals

6.5.1 What you can do

You can suppress all totals by specifying a **details only** report on the OUT parameter.

6.5.2 How to do it

Code:

1. The 2-digit report number
2. The word OUT
3. D, after skipping one or more spaces

6.5.3 Demonstration

Objective: This example produces a listing of employee salaries, without totaling the amount.

Parameters

Parameters	Description
01OUT D	01 is the report number. OUT specifies an OUTPUT parameter. D specifies printing of detail lines only.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25      'EMPLOYEE NAME'
REC SALARY        160   5   3  DP=2  'ANNUAL SALARY'
01OUT D
013 EMPLOYEE SALARY LISTING
0151*010 EMPLOYEE SZ=20      HR
0151*020 SALARY   SZ=11  F$      HR
```

Result:

6.5 Suppressing all totals

REPORT NO. 01		EMPLOYEE SALARY LISTING	09/24/99 PAGE	1
		EMPLOYEE NAME	ANNUAL SALARY	
TERRY	JANENS		\$38,000.00	
JOE	NGUYA		\$31,000.00	
MARK	TIME		\$33,000.00	
ROGER	WILCO		\$80,000.00	
ALBERT	BREEZE		\$38,000.00	
CAROLYN	CROW		\$37,500.00	
BURT	LANCHESTER		\$54,500.00	
RENE	MAKER		\$85,000.00	
DUDY	BLOOMER		\$15,000.00	
EDWARD	HUTTON		\$44,000.00	

6.6 Generating subtotals

6.6.1 What you can do

You can obtain subtotals for your report by creating a pause in the output processing with a **control break** on a sorted field.

6.6.2 How to do it

Code from left to right:

1. The 2-digit report number.
2. The word SORT.
3. The name of the field to be sorted.
4. The ascending (blank or A) or descending (D) sort order.
5. The control break code that requests the subtotal and tells CA-Culprit what to do after the subtotal is printed:
 - **1** — Start a new report page
 - **0** — Print one blank line
 - **Dash (-)** — Print two blank lines
 - **Plus sign (+)** — Begin the next report line immediately after the subtotal line
6. Repeat steps 3, 4, and 5 as appropriate. Separate each entry by at least one space.

If your data file is sorted before you run your CA-Culprit code, use the NOSORT option with the SORT parameter.

►► NOSORT is explained in 13.2.4, “Demonstration (2): Using the SORT/NOSORT parameters” on page 13-12.

6.6.3 Demonstration

Objective: This example:

- Sorts the data by department.
- Obtains a salary subtotal each time the department changes.
- Inserts two blank lines after each subtotal.
- Lists salaries in descending order. Duplicate salaries print in alphabetical order by employee last name.

01SORT DEPARTMENT,
- SALARY, D LAST-NAME

SORT specifies the **SORT** parameter.

DEPARTMENT specifies a sort on **DEPARTMENT**.

- (Dash) specifies a control break, causing subtotaling and spacing to occur each time there is a new department.

SALARY specifies a sort on **SALARY** within each department.

D specifies a salary sort in descending order (that is, from largest to smallest).

LAST-NAME specifies a sort on employees within the same salary and department.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC FIRST-NAME    5   10
REC LAST-NAME     15  15
REC DEPARTMENT    115  25
REC SALARY        160  5   3   DP=2
REC TITLE         171  20
013 EMPLOYEE SALARY LISTING BY DEPARTMENT
01SORT DEPARTMENT,- SALARY,D LAST-NAME
0151*010 DEPARTMENT    HH 'DEPARTMENT NAME'
0151*020 LAST-NAME SZ=10 HH '                           EMPLOYEE NAME'
0151*025 FIRST-NAME
0151*030 TITLE       HH 'JOB TITLE'
0151*040 SALARY      HH '     ANNUAL SALARY'
```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING BY DEPARTMENT	09/24/99	PAGE	1	JOB TITLE	ANNUAL SALARY
DEPARTMENT NAME	EMPLOYEE NAME					
ACCOUNTING AND PAYROLL	JENSON	RUPERT	MGR ACCTNG/PAYROLL	82,000.00		
ACCOUNTING AND PAYROLL	KIMBALL	MARIANNE	ACCOUNTANT	45,000.00		
ACCOUNTING AND PAYROLL	HUTTON	EDWARD	FINANCIAL ANALYST	44,000.00		
ACCOUNTING AND PAYROLL	BLOOMER	JUNE	PAYROLL CLERK	15,000.00		
ACCOUNTING AND PAYROLL	KING	DORIS	AR CLERK	14,500.00		
ACCOUNTING AND PAYROLL	NICEMAN	BRIAN	AP CLERK	14,000.00		
				214,500.00		
BLUE SKIES	MOON	DANIEL	MGR BLUE SKIES	72,000.00		
BLUE SKIES	CLOUD	BETH	CUMULUS CARETAKER	52,750.00		
BLUE SKIES	DONOVAN	ALAN	SUNSHINE SUPERVISOR	33,500.00		
				158,250.00		
THERMOREGULATION	WILCO	ROGER	MGR THERMOREGULATION	80,000.00		
THERMOREGULATION	FINN	PHINEAS	KEEPER OF BALLOONS	45,000.00		
THERMOREGULATION	CLOTH	TERRY	HUMIDITY CONTROL CLK	38,000.00		
THERMOREGULATION	TIME	MARK	WINTERIZER	33,000.00		
THERMOREGULATION	KASPAR	JOE	WINTERIZER	31,000.00		
				227,000.00		
				2,522,500.00		

6.7 Modifying subtotal and total line placement

6.7.1 What you can do

Selected totals and labels can be placed in specific report columns.

6.7.2 How to do it

To control total lines, use **type 6** parameters in much the same way as you use type 5 parameters.

When the same numeric data item appears on both type 5 and type 6 parameters, both parameters must contain identical information, such as size (SZ=) and decimal point placement (DP=). Heading information can appear on either parameter, but not both.

6.7.3 Demonstration

Objective: The next report prints subtotal and grand total fields in their own column.

Parameters

Parameters	Description
0161*055 SALARY	01 is the report number. 6 identifies the parameter type. 1 specifies this field to appear on the first totals print line. *055 specifies automatic placement of the subtotal to the right of the individually listed salaries. SALARY specifies the field to be totaled.

Complete code

col. 2
 ↓
 IN 200 F 400 PS(TAPE)
 REC EMPLOYEE 5 25
 REC FIRST-NAME 5 10
 REC LAST-NAME 15 15
 REC DEPARTMENT 115 25
 REC SALARY 160 5 3 DP=2
 REC TITLE 171 20
 013 EMPLOYEE SALARY LISTING BY DEPARTMENT
 01SORT DEPARTMENT 0 SALARY D LAST-NAME
 0151*010 DEPARTMENT HH 'DEPARTMENT'
 0151*020 LAST-NAME SZ=10 HH 'EMPLOYEE NAME'
 0151*030 FIRST-NAME
 0151*040 TITLE HH 'JOB TITLE'
 0151*050 SALARY HH 'ANNUAL SALARY'
0161*055 SALARY

Result:

REPORT NO. 01 DEPARTMENT	EMPLOYEE SALARY LISTING BY DEPARTMENT EMPLOYEE NAME	09/24/99 PAGE	1 JOB TITLE	ANNUAL SALARY
ACCOUNTING AND PAYROLL	JENSON	RUPERT	MGR ACCTNG/PAYROLL	82,000.00
ACCOUNTING AND PAYROLL	KIMBALL	MARIANNE	ACCOUNTANT	45,000.00
ACCOUNTING AND PAYROLL	HUTTON	EDWARD	FINANCIAL ANALYST	44,000.00
ACCOUNTING AND PAYROLL	BLOOMER	JUNE	PAYROLL CLERK	15,000.00
ACCOUNTING AND PAYROLL	KING	DORIS	AR CLERK	14,500.00
ACCOUNTING AND PAYROLL	NICEMAN	BRIAN	AP CLERK	14,000.00
				214,500.00
BLUE SKIES	MOON	DANIEL	MGR BLUE SKIES	72,000.00
BLUE SKIES	CLOUD	BETH	CUMULUS CARETAKER	52,750.00
BLUE SKIES	DONOVAN	ALAN	SUNSHINE SUPERVISOR	33,500.00
				158,250.00
THERMOREGULATION	WILCO	ROGER	MGR THERMOREGULATION	80,000.00
THERMOREGULATION	FINN	PHINEAS	KEEPER OF BALLOONS	45,000.00
THERMOREGULATION	CLOTH	TERRY	HUMIDITY CONTROL CLK	38,000.00
THERMOREGULATION	TIME	MARK	WINTERIZER	33,000.00
THERMOREGULATION	KASPAR	JOE	WINTERIZER	31,000.00
				227,000.00
				2,522,500.00

6.8 Selectively printing output lines

6.8.1 What you can do

When working with multiple levels of output (one or more control breaks), you can control when the output of each level is printed.

6.8.2 How to do it

To control the printing of multiple levels of output, use:

- A **SORT** parameter, with one or more control breaks
- A **type 6** parameter, to print the line
- A **type 8** parameter, to act upon the data
- The **TAKE** instruction on a type 8 parameter, to specify which type 6 line to print and stop further processing for that control break
- The **DROP** instruction on a type 8 parameter, to stop processing for that control break

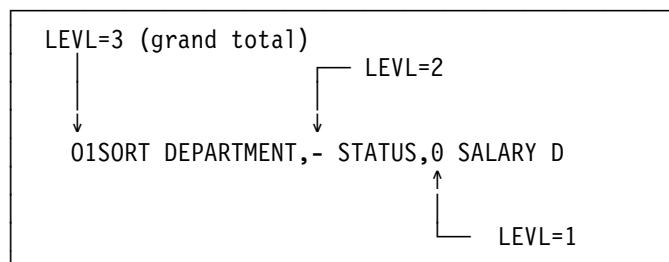
Coding for type 8 parameters is similar to that for type 7 parameters. Enter from left to right:

1. A 2-digit report number
2. The parameter type (8)
3. A 3-digit sequence number

6.8.3 Demonstration (1): Selecting totals output

Objective: This report prints three types of totals with appropriate labels:

- The total of employee salaries by job status within each department (STATUS CODE TOTAL)
- The total of department salaries (DEPARTMENT TOTAL)
- The grand total of all salaries (GRAND TOTAL)



Parameters

Parameters	Description
01SORT DEPARTMENT, - STATUS, 0 SALARY D	DEPARTMENT, - STATUS, 0 specify control breaks for subtotals.
0161*0300'STATUS CODE TOTAL' 0161*040 SALARY SZ=10 0162*0300'DEPARTMENT TOTAL' 0162*040 SALARY SZ=10 0163*0300'GRAND TOTAL' 0163*040 SALARY SZ=10	61, 62, and 63 specify groups of print lines used to label and print subtotal, total, and grand totals.
018010 IF LEVL EQ 1 100	<i>When the value in the STATUS field changes, go to line 100 of the type 8 code.</i> 8 specifies a type 8 parameter, which is used when testing control break levels. IF checks the value of LEVL. LEVL EQ 1 refers to the control break level on the SORT parameter. STATUS is the first control break (LEVL 1). 100 directs processing to line 100 of the type 8 parameter code.
018020 IF LEVL EQ 2 200	<i>When the value of DEPARTMENT changes, go to line 200 of the type 8 code.</i> LEVL EQ 2 specifies DEPARTMENT, the second (LEVL 2) control break field. 200 directs processing to line 200 of the type 8 parameter code.
018030 IF LEVL EQ 3 300	<i>When all the records are processed, go to line 300 of the type 8 code.</i> LEVL EQ 3 specifies grand total time; LEVL 3 is one level higher than the highest control break. 300 directs processing to line 300 of the type 8 parameter code.
018100 TAKE 1	<i>Print line 1 of the type 6 code.</i> TAKE 1 points to the first set of type 6 print lines (STATUS CODE TOTAL followed by total salary amount for the particular status) and stops processing of the current record.

Parameters	Description
018200 TAKE 2	<i>Print line 2 of the type 6 code.</i> TAKE 2 points to the second set of type 6 print lines (DEPARTMENT TOTAL followed by the department total amount) and stops processing of the current record.
018300 TAKE 3	<i>Print line 3 of the type 6 code.</i> TAKE 3 points to the third set of type 6 print lines (GRAND TOTAL followed by the grand total amount) and stops processing of the report.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC FIRST-NAME    5   10
REC LAST-NAME     15  15
REC STATUS        82   2
REC DEPARTMENT    115  25
REC SALARY        160   5 3 DP=2
REC TITLE         171  20
013 EMPLOYEE SALARY LISTING
01SORT DEPARTMENT - STATUS 0 SALARY D
0151*010 DEPARTMENT      HH 'DEPARTMENT'
0151*020 LAST-NAME SZ=10 HH 'EMPLOYEE NAME'
0151*025 FIRST-NAME
0151*030 TITLE          HH 'TITLE'
0151*035 STATUS          HH 'STATUS' 'CODE'
0151*040 SALARY SZ=10    HH 'ANNUAL SALARY'
0161*0300 'STATUS CODE TOTAL'
0161*040 SALARY SZ=10
0162*0300 'DEPARTMENT TOTAL'
0162*040 SALARY SZ=10
0163*0300 'GRAND TOTAL'
0163*040 SALARY SZ=10
018010  IF LEVL EQ 1 100
018020  IF LEVL EQ 2 200
018030  IF LEVL EQ 3 300
018100  TAKE 1
018200  TAKE 2
018300  TAKE 3

```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING	11/13/99 PAGE	1		
DEPARTMENT	EMPLOYEE NAME		TITLE	STATUS CODE	ANNUAL SALARY
PERSONNEL	PEOPLES	ELEANOR	MGR PERSONNEL	01	80,000.00
PERSONNEL	ORGRATZI	MADELINE	RECRUITER/INTERVWR	01	39,000.00
PERSONNEL	FITZHUGH	TOM	PERSONNEL CLERK	01	13,000.00
			STATUS CODE TOTAL		132,000.00
PERSONNEL	JOHNSON	CYNTHIA	PERSONNEL CLERK	05	13,500.00
			STATUS CODE TOTAL		13,500.00
			DEPARTMENT TOTAL		145,500.00
PUBLIC RELATIONS	BANK	DONTE	MGR PUBLIC RELATIONS	01	80,000.00
PUBLIC RELATIONS	PENMAN	LAURA	PR WRITER	01	39,000.00
PUBLIC RELATIONS	JACKSON	JOCK	SPORTS CONSULTANT	01	34,000.00
PUBLIC RELATIONS	ANGELO	MICHAEL	ILLUSTRATOR	01	18,000.00
PUBLIC RELATIONS	MCDougall	CAROL	PASTE-UP ARTIST	01	18,000.00
			STATUS CODE TOTAL		189,000.00
PUBLIC RELATIONS	ZEDI	BETSY	PR WRITER	05	37,000.00
			STATUS CODE TOTAL		37,000.00
			DEPARTMENT TOTAL		226,000.00
			GRAND TOTAL		2,554,718.00

6.8.4 Demonstration (2): Using a work field to print total line labels

Objective: This example uses a single type 6 parameter line, a MOVE instruction, and an alphanumeric work field to produce the same report shown in Demonstration (1).

An alphanumeric work field contains the following information from left to right:

- A 2-digit report number
- The work field type (0)
- The name of the work field
- Single quotation marks to enclose a space large enough to hold the text for the longest label that will be placed in the field

Setting up the work field

Parameters	Description
010 TOTAL-MESSAGE ' ... '	<p>01 is the report number.</p> <p>0 identifies a work field.</p> <p>TOTAL-MESSAGE is the name of the work field.</p> <p>' ' specifies the space allowed for the label text. Our work field is 17 characters long to accommodate STATUS CODE TOTAL, the longest label.</p>

Setting up the print line

Parameters	Description
0161*0400 TOTAL-MESSAGE . 0161*060 SALARY SZ=10	TOTAL-MESSAGE assigns the test of the work field to the print line. SALARY prints the salary amount to the right of the text.

Placing text into the work field

018010 IF LEVL EQ 1 100 ... 018100 MOVE 'STATUS CODE TOTAL' TO TOTAL-MESSAGE 018110 TAKE	<i>When the status code changes, go to line 100 of the type 8 code for the next instruction.</i> MOVE places the phrase STATUS CODE TOTAL into the work field. TAKE prints the phrase STATUS CODE TOTAL and the salary amount. CA-Culprit goes on to process the next record.
018020 IF LEVL EQ 2 200 ... 018200 MOVE 'DEPARTMENT TOTAL' TO TOTAL-MESSAGE 018210 TAKE	<i>When the department name changes, go to line 200 of the type 8 code for the next instruction.</i> MOVE places the phrase DEPARTMENT TOTAL into the work field. TAKE prints the phrase DEPARTMENT TOTAL and the salary totals for the department. CA-Culprit goes on to process the next record.
018030 IF LEVL EQ 3 300 ... 018300 MOVE 'GRAND TOTAL' TO TOTAL-MESSAGE 018310 TAKE.	<i>When all the records are processed, go to line 300 of the type 8 code for the next instruction.</i> MOVE places the phrase GRAND TOTAL into the work field. TAKE prints the phrase GRAND TOTAL and the salary totals for all departments.

Complete code

```

col. 2
↓
REC EMPLOYEE      5  25
REC FIRST-NAME    5  10
REC LAST-NAME     15 15
REC STATUS        82  2
REC DEPARTMENT    115 25
REC SALARY        160 5  3  DP=2
REC TITLE         171 20
013 EMPLOYEE SALARY LISTING
01SORT DEPARTMENT - STATUS 0 SALARY D
010 TOTAL-MESSAGE
0151*010 DEPARTMENT   HH 'DEPARTMENT'
0151*020 LAST-NAME SZ=10 HH '                           EMPLOYEE NAME'
0151*030 FIRST-NAME
0151*040 TITLE       HH 'TITLE'
0151*050 STATUS      HH 'STATUS' 'CODE'
0151*060 SALARY SZ=10 HH '      ANNUAL SALARY'
0161*0400 TOTAL-MESSAGE
0161*060 SALARY SZ=10
018010 IF LEVL EQ 1 100
018020 IF LEVL EQ 2 200
018030 IF LEVL EQ 3 300
018100 MOVE 'STATUS CODE TOTAL' TO TOTAL-MESSAGE
018110 TAKE
018200 MOVE 'DEPARTMENT TOTAL' TO TOTAL-MESSAGE
018210 TAKE
018300 MOVE 'GRAND TOTAL' TO TOTAL-MESSAGE
018310 TAKE

```

Result:

REPORT NO. 01	EMPLOYEE SALARY LISTING	11/13/99 PAGE	1				
DEPARTMENT	EMPLOYEE NAME			TITLE	STATUS CODE	ANNUAL SALARY	
PERSONNEL	PEOPLES	ELEANOR	MGR PERSONNEL	01	80,000.00		
PERSONNEL	ORGATZI	MADELINE	RECRUITER/INTERVWR	01	39,000.00		
PERSONNEL	FITZHUGH	TOM	PERSONNEL CLERK	01	13,000.00		
			STATUS CODE TOTAL		132,000.00		
PERSONNEL	JOHNSON	CYNTHIA	PERSONNEL CLERK	05	13,500.00		
			STATUS CODE TOTAL		13,500.00		
			DEPARTMENT TOTAL		145,500.00		
PUBLIC RELATIONS	BANK	DONTE	MGR PUBLIC RELATIONS	01	80,000.00		
PUBLIC RELATIONS	PENMAN	LAURA	PR WRITER	01	39,000.00		
PUBLIC RELATIONS	JACKSON	JOCK	SPORTS CONSULTANT	01	34,000.00		
PUBLIC RELATIONS	ANGELO	MICHAEL	ILLUSTRATOR	01	18,000.00		
PUBLIC RELATIONS	MCDougall	CAROL	PASTE-UP ARTIST	01	18,000.00		
			STATUS CODE TOTAL		189,000.00		
PUBLIC RELATIONS	ZEDI	BETSY	PR WRITER	05	37,000.00		
			STATUS CODE TOTAL		37,000.00		
			DEPARTMENT TOTAL		226,000.00		
			GRAND TOTAL		2,554,718.00		

6.9 Obtaining sort field values

6.9.1 What you can do

You can obtain the value of a numeric sort key when control breaks occur and when the last detail line for a control break is processed.

6.9.2 How to do it

You can obtain the value of a numeric key as long as the following conditions are true:

- The field appears on a SORT parameter
- The field does not appear on a type 5 edit line

6.9.3 Demonstration

Objective: The next report:

- Lists unsalaried employees (those that earn \$19000 or less in one year)
- Calculates the hourly wage of these employees
- Sorts these employees in ascending order on the hourly wage field
- Prints the highest hourly wage (obtained from the last sort field value) at the end of the report

Parameters

Parameters	Description
010 AMOUNT DP=2	010 AMOUNT defines a numeric work field to hold the hourly wage computation.
01SORT AMOUNT	SORT AMOUNT specifies sorting of AMOUNT in ascending order so that the last amount represents the highest hourly wage.
0162*015 'HIGHEST HOURLY WAGE' 0162*0300AMOUNT SZ=9 F\$	62 specifies the second type 6 line to label and print the last value for AMOUNT.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME    15   15
REC DEPARTMENT   115   25
REC SALARY       160   5   3   DP=2
SEL SALARY LE 19000
01OUT 80
01SORT AMOUNT
013 UNSALARIED EMPLOYEE COMPENSATION
010 AMOUNT DP=2
0151*010 EMPLOYEE          HH 'EMPLOYEE' 'NAME'
0151*020 SALARY SZ=9     F$  HH 'ANNUAL SALARY'
0161*020 SALARY SZ=9     F$ 
0162*0150'HIGHEST HOURLY WAGE'
0162*030 AMOUNT SZ=9     F$ 
017010 COMPUTE (SALARY / 52) / 40  AMOUNT

```

Result

REPORT NO. 01		UNSALARIED EMPLOYEE COMPENSATION	09/24/99 PAGE	1
	EMPLOYEE	NAME	ANNUAL SALARY	
NANCY	TERNER		\$13,000.00	
TOM	FITZHUGH		\$13,000.00	
CYNTHIA	JOHNSON		\$13,500.00	
BRIAN	NICEMAN		\$14,000.00	
ROBIN	GARDNER		\$14,000.00	
SANDY	KRAAMER		\$14,000.00	
DORIS	KING		\$14,500.00	
GEORGE	FONRAD		\$14,750.00	
JUNE	BLOOMER		\$15,000.00	
MICHAEL	ANGELO		\$18,000.00	
CAROL	MCDougall		\$18,000.00	
HERBERT	LIPSICH		\$18,500.00	
			\$180,250.00	
HIGHEST HOURLY WAGE			\$8.89	

6.10 Obtaining sorted work field values

6.10.1 What you can do

You can obtain the value of an alphanumeric sort key when a control break occurs and when the last detail line for a control break is processed.

6.10.2 How to do it

To access the contents of a work field, code the field on a SORT parameter.

6.10.3 Demonstration

This report sorts the message work field and produces a listing of employees organized into executive, salaried, and hourly categories.

Parameters

Parameters	Description
01SORT MESSAGE - SALARY 010 MESSAGE ' '.	MESSAGE specifies an alphanumeric work field sorted by salary group.
010 AMOUNT DP=2	AMOUNT is a numeric work field that holds the results of the hourly wage computation.
0152*030+AMOUNT SZ=9 F\$ HH ' HOURLY RATE'.	0152030+ specifies a second print line, which is used to select the hourly wage appearing on the report. + specifies printing AMOUNT on the first line (overprint).
	AMOUNT specifies printing the current contents of the work field AMOUNT .
0161*0400SALARY SZ=9 F\$	SALARY specifies printing the salary total at a control break.
017010 IF SALARY LE 19000 500 ... 017500 MOVE 'HOURLY ' TO MESSAGE ... 017 COMPUTE (SALARY / 52) / 40 AMOUNT ... 017 TAKE (1 2).	IF tests for salaries less than or equal to \$19,000. MOVE places HOURLY in the MESSAGE work field. COMPUTE calculates hourly salary. TAKE (1 2) specifies extracting the first and second detail lines.
017020 IF SALARY GT 50000 600 ... 017 MOVE 'SALARIED ' TO MESSAGE ... 017 TAKE 1 ... 017600 MOVE 'EXECUTIVE' TO MESSAGE ... 017 TAKE 1.	IF tests for salaries greater than \$50,000. If true, line 600 is executed. MOVE places SALARIED in the MESSAGE work field. TAKE 1 specifies printing the first detail line. MOVE places EXECUTIVE in the MESSAGE work field if SALARY is greater than \$50,000. TAKE 1 specifies printing the first detail line.

Complete Code

```

IN 200 F 400 PS(TAPE)
REC EMPLOYEE      5   25
REC LAST-NAME    15   10
REC DEPARTMENT   115   25
REC SALARY       160   5   3  DP=2
01SORT MESSAGE - SALARY
013 EMPLOYEE COMPENSATION STATUS
010 MESSAGE '
010 AMOUNT DP=2
0151*010 EMPLOYEE          HH 'EMPLOYEE NAME'
0151*020 MESSAGE           HH 'STATUS'
0152*030+AMOUNT SZ=9     F$  HH '      HOURLY RATE'
0151*040 SALARY SZ=9      F$  HH 'ANNUAL SALARY'
0161*0400SALARY SZ=9     F$
017010 IF SALARY LE 19000 500
017020 IF SALARY GT 50000 600
017   MOVE 'SALARIED ' TO MESSAGE
017   TAKE 1
017500 MOVE 'HOURLY ' TO MESSAGE
017   COMPUTE (SALARY / 52) / 40  AMOUNT
017   TAKE (1 2)
017600 MOVE 'EXECUTIVE' TO MESSAGE
017   TAKE 1

```

Result:

REPORT NO. 01	EMPLOYEE NAME	EMPLOYEE COMPENSATION STATUS	09/26/99 PAGE	1	HOURLY RATE	ANNUAL SALARY
BETH BURT	CLOUD LANCHESTER	EXECUTIVE EXECUTIVE			\$52,750.00 \$54,500.00	
ROBBY THEMIS HENRIETTA	WILDER PAPAZEUS HENDON	EXECUTIVE EXECUTIVE EXECUTIVE			\$90,000.00 \$100,000.00 \$240,000.00	
						\$1,291,250.00
DORIS GEORGE JUNE MICHAEL CAROL HERBERT	KING FONRAD BLOOMER ANGELO MCDougall LIPSICH	HOURLY HOURLY HOURLY HOURLY HOURLY HOURLY			\$14,500.00 \$14,750.00 \$15,000.00 \$18,000.00 \$18,000.00 \$18,500.00	
						\$180,250.00
TERENCE EDWARD MARIANNE PHINEAS HARRY RICHARD	KLWELLEN HUTTON KIMBALL FINN ARM WAGNER	SALARIED SALARIED SALARIED SALARIED SALARIED SALARIED			\$43,000.00 \$44,000.00 \$45,000.00 \$45,000.00 \$46,000.00 \$47,000.00	
						\$1,051,000.00
						\$2,522,500.00

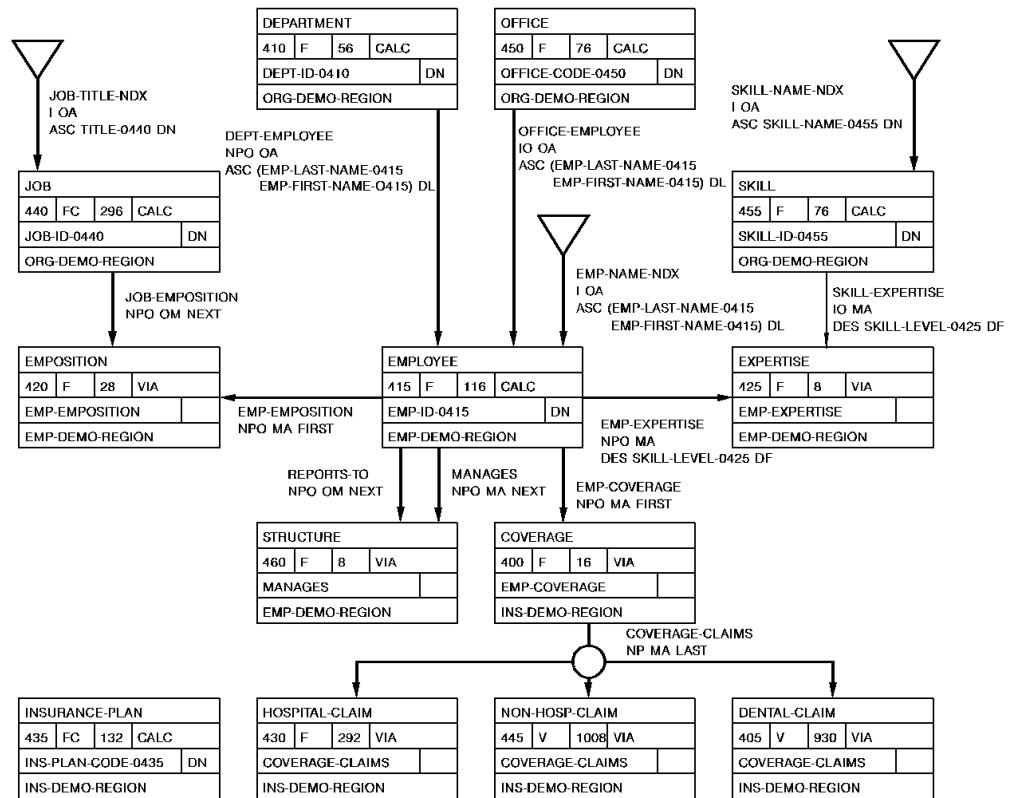
Chapter 7. Generating Reports From Database Files

7.1 Listing data	7-4
7.1.1 What you can do	7-4
7.1.2 How to do it	7-4
7.1.3 Demonstration	7-4
7.2 Using logical records	7-6
7.2.1 What you can do	7-6
7.2.2 How to do it	7-6
7.2.3 Demonstration	7-6
7.3 Selective processing	7-8
7.3.1 What you can do	7-8
7.3.2 How to do it	7-8
7.3.3 Demonstration	7-8

CA-Culprit reports can be written from data stored in a database as well as from data stored in standard files.

You need to obtain the following information about your CA-IDMS/DB database before you code your report:

- The **data dictionary name**, if you are not using the primary dictionary of your installation
- The name of the **sbschema** that describes the database record types and fields that CA-Culprit accesses to build the report
- A **data structure diagram**, similar to that shown in the figure below



7.1 Listing data

7.1.1 What you can do

You can list related occurrences of records in the database.

7.1.2 How to do it

To access the database, code:

1. A **DATABASE** parameter with the **DICTNAME=** option if you are using an alternate dictionary
2. An **INPUT** parameter with the following information:
 - a. **DB** to specify that the report is accessing a database
 - b. **SS=** with the name of the subschema you are using
 - a. The path name (PATH-ID)
 - b. The name of each record as it is encountered on the path
3. **Type 5** parameters with appropriate field names (as listed in the subschema)

Note: CA-Culprit automatically generates REC parameters from the definitions stored in the data dictionary. REC parameters do not have to be coded unless you want field names or data types that are different from those given in the subschema.

7.1.3 Demonstration

Objective: This report lists all employee name, ID, start date, and job title information stored in the Personnel database. Any employee having more than one job title in the course of employment is listed each time the job title changes. Numeric fields are not edited.

Parameters

Parameters	Description
DATABASE DICTNAME=DOCUDICT	DATABASE specifies the database parameter. If used, it must always be the first parameter. DICTNAME=DOCUDICT specifies DOCUDICT as the dictionary used for the report.
IN DB SS=EMPSS01	DB specifies a database as the data source. SS=EMPSS01 specifies EMPSS01 as the subschema used.
PATHAA EMPLOYEE EMPOSITION JOB	PATH specifies the PATH parameter. AA names the path. EMPLOYEE EMPOSITION JOB specifies the related database records used as a path through the database.
0151*010 EMP-NAME-0415 0151*020 EMP-ID-0415 0151*030 START-DATE-0420 0151*040 TITLE-0440	The field names are the same as those used in the database.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
PATHAA EMPLOYEE EMPOSITION JOB
01OUT 80 D
013 EMPLOYEE LISTING
0151*010 EMP-NAME-0415          HH ' ' 'EMPLOYEE' 'NAME'
0151*020 EMP-ID-0415            HH 'EMPLOYEE' 'ID'
0151*030 START-DATE-0420        HH 'START' 'DATE'
0151*040 TITLE-0440             HH 'JOB' 'TITLE'
```

Result

REPORT NO. 01	EMPLOYEE LISTING	09/24/99	PAGE	1
EMPLOYEE NAME	EMPLOYEE ID	START DATE	JOB TITLE	
KATHERINE O'HEARN	23	790505	PROGRAMMER/ANALYST	
KATHERINE O'HEARN	23	780504	PROGRAMMER TRAINEE	
ROBBY WILDER	472	790716	DIR CORP CONFUSION	
BURT LANCHESTER	301	800203	RAINMAKER	
BURT LANCHESTER	301	770203	RAINMAKER	
BURT LANCHESTER	301	750203	RAINMAKER	
VLADIMIR HEAROWITZ	27	810909	PROGRAMMER/ANALYST	
THEMIS PAPAZEUS	471	820101	DIR WEATHER	
THEMIS PAPAZEUS	471	780907	MGR BRAINSTORMING	
MONTE BANK	7	780430	MGR PUBLIC RELATIONS	
CAROLYN CROW	334	790617	RAINDANCE CONSULTANT	
CAROL McDougall	127	800607	PASTE-UP ARTIST	
JULIE JENSEN	19	820929	PROGRAMMER/ANALYST	

7.2 Using logical records

7.2.1 What you can do

Logical records allow reports to be created from data stored in the database with no concern for where the information is located. All you need to know is the name of the logical record and the fields contained in the record.

7.2.2 How to do it

When using logical records, code:

1. A **DATABASE** parameter with the DICTNAME= option if an alternate dictionary is being used
2. An **INPUT DB** parameter with the SS= keyword and the subschema name
3. **Type 5** parameters with appropriate field names

7.2.3 Demonstration

Objective: This report lists employee information from data contained in the logical record EMP-JOB-LR.

Parameters

Parameters	Description
REC START-DATE 79 6 2	REC START-DATE redefines the date field as numeric to allow automatic formatting.
PATHAA EMP-JOB-LR	EMP-JOB-LR is the logical record containing data from the database.
01OUT D	D suppresses totaling of the EMP-ID-0415 field.

Complete code

```
col. 2
↓
REC START-DATE 79 6 2
PATHAA EMP-JOB-LR
01OUT D
013 EMPLOYEE LISTING
0151*005 EMP-ID-0415          HH 'ID'
0151*010 EMP-NAME-0415        HH ' ' 'EMPLOYEE' 'NAME'
0151*020 TITLE-0440           HH 'JOB TITLE'
0151*030 START-DATE           FD   HH 'START DATE'
```

Result:

REPORT NO. 01	EMPLOYEE LISTING	09/25/99 PAGE	1
ID	EMPLOYEE NAME	JOB TITLE	START DATE
23	KATHERINE O'HEARN	PROGRAMMER/ANALYST	79/05/05
23	KATHERINE O'HEARN	PROGRAMMER TRAINEE	78/05/04
472	ROBBY WILDER	DIR CORP CONFUSION	79/07/16
301	BURT LANCHESTER	RAINMAKER	80/02/03
301	BURT LANCHESTER	RAINMAKER	77/02/03
301	BURT LANCHESTER	RAINMAKER	75/02/03
27	VLADIMIR HEAROWITZ	PROGRAMMER/ANALYST	81/09/09
471	THEMIS PAPAZEUS	DIR WEATHER	82/01/01
471	THEMIS PAPAZEUS	MGR BRAINSTORMING	78/09/07
7	MONTE BANK	MGR PUBLIC RELATION	78/04/30
334	CAROLYN CROW	RAINDANCE CONSULTAN	79/06/17
127	CAROL McDougall	PASTE-UP ARTIST	80/06/07
19	JULIE JENSEN	PROGRAMMER/ANALYST	82/09/29

7.3 Selective processing

7.3.1 What you can do

Data that meets specified criteria can be selected for a report by applying selection criteria to logical records.

7.3.2 How to do it

Data can be selected:

1. With conditional statements set up by a **WHERE** clause:
retrieves only those records with a job ID equal to 3051
2. By searching for defined values with **CONTAINS** or **MATCHES** patterns:
retrieves employee records where the character set CAS appears somewhere in the name
retrieves employee records where the first three characters of the employee name are CAS
retrieves employee records where names start with CA followed by any alphabetic character
retrieves employee records where ids start with 02 followed by any two digits

7.3.3 Demonstration

Objective: This report lists information about those employees with both the word ENTRY in their job title and ids in the ranges 0001 to 0048 and 0999 to 9999.

Parameters

Parameters	Description
<pre>PATHAA EMPLOYEE-TABLE WHERE (EMP-ID-0415 LE '0048' OR * EMP-ID-0415 GT '0999') * AND DEPT-ID-0410 GT '1000'</pre>	<p>EMPLOYEE-TABLE names the logical record containing database data.</p> <p>WHERE forms a compound conditional statement to specify the criteria for record retrieval.</p> <p>* specifies a continuation line.</p>

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS09
PATHAA EMPLOYEE-TABLE WHERE (EMP-ID-0415 LE '0048' OR
*                           EMP-ID-0415 GT '0999')
*                           AND DEPT-ID-0410 GT '1000'
010UT D
013 EMPLOYEE LISTING
0151*005 EMP-ID-0415          HH 'ID'
0151*010 EMP-NAME-0415        HH ' ' 'EMPLOYEE' 'NAME'
0151*020 DEPT-ID-0410         HH 'DEPT ID'

```

Result:

REPORT NO. 01	EMPLOYEE LISTING	09/25/99 PAGE	1
ID	EMPLOYEE NAME		DEPT ID
7	MONTE	BANK	4000
4	HERBERT	CRANE	3200
24	JANE	DOUGH	3100
32	JANE	FERNDALE	3200
45	GEORGE	FONRAD	3200
29	JAMES	GALLWAY	3100
3	JENNIFER	GARFIELD	3100
28	PERCY	GRANGER	3100
27	VLADIMIR	HEAROWITZ	3100
20	JAMES	JACOBI	3100
19	JULIE	JENSEN	3100
11	RUPERT	JENSON	2000
16	TERENCE	KLWELLEN	3200
31	HERBERT	LIPSICH	3200
35	LARRY	LITERATA	3100
15	RENE	MAKER	5100
23	KATHERINE	O'HEARN	3100
48	NANCY	TERNER	3200
21	RALPH	TYRO	3100

Chapter 8. Building and Using Data Tables

8.1	Creating tables	8-4
8.1.1	What you can do	8-4
8.1.2	How to do it	8-4
8.1.3	Demonstration (1): Creating a table from a sequential file	8-6
8.1.4	Demonstration (2): Creating a table from the database	8-8
8.1.5	Demonstration (3): Creating a table from an existing table	8-10
8.1.6	Demonstration (4): Creating a totals-only table	8-12
8.2	Retrieving data tables	8-13
8.2.1	What you can do	8-13
8.2.2	How to do it	8-13
8.2.3	Demonstration	8-13
8.3	Modifying data tables	8-15
8.3.1	What you can do	8-15
8.3.2	How to do it	8-15
8.3.3	Demonstration (1): Adding rows to an existing table	8-15
8.3.4	Demonstration (2): Replacing rows of an existing table	8-17
8.3.5	Demonstration (3): Deleting a table	8-18
8.3.6	Demonstration (4): Regenerating a table	8-19
8.4	Consolidating tables	8-23
8.4.1	What you can do	8-23
8.4.2	How to do it	8-23
8.4.3	Demonstration	8-25

CA-Culprit is used in an CA-IDMS/DB environment to create, retrieve, and manipulate data tables. CA-Culprit:

- Creates tables from:
 - Conventional files
 - The database
 - Other tables
- Retrieves tables
- Updates and regenerates tables
- Consolidates tables
- Deletes tables

►► For a general discussion about tables, refer to the *CA-IDMS ASF User Guide*.

8.1 Creating tables

8.1.1 What you can do

You can:

- Create a table from data stored
 - On conventional files (sequential, ISAM, or VSAM)
 - On a database (CA-IDMS/DB, IMS, RDMS, or TOTAL)
 - In an already existing table
- Store rows in ascending or descending order
- Find errors in your code by using the data table reports that are automatically generated in the Input Parameter and Run-Time Message Listings of the CA-Culprit run

8.1.2 How to do it

To create a table:

1. **Define the incoming data** using input definition parameters (shown in the table below).
2. **Define the table** using the OUTPUT parameter and the keywords listed in the table below.
3. **Define the columns of the table** using type 5 or type 6 parameters.
4. **Specify special options**, as needed, using the options listed in the table below.

►► You can find complete descriptions of the parameters and keywords in the *CA-Culprit Reference Guide*.

Required parameters and keywords for creating a table

If the data source is...	the required parameters and keywords are...
A conventional file (sequential, ISAM, and VSAM)	An INPUT parameter, including the file type (PS, IS, CARD, or VS) REC parameters (at least one) An OUTPUT parameter with: TABLE= TYPE=CREATE USER= PW= CATALOG= Type 5 parameters (at least one) Type 6 parameters to create a totals-only table
An CA-IDMS/DB database	A DATABASE PARAMETER with DICTNAME= An INPUT DB SS= parameter A PATH parameter An OUTPUT parameter with: TABLE= TYPE=CREATE USER= PW= CATALOG= Type 5 parameters (at least one) Type 6 parameters to create a totals-only table
An existing table	An INPUT parameter with: TABLE= TYPE=COPY USER= PW= CATALOG= An OUTPUT parameter with: TABLE= TYPE=CREATE USER= PW= CATALOG= Type 5 parameters (at least one) Type 6 parameters to generate a totals-only table

Options available when creating a table

The option ...	used on this parameter ...	specifies ...
IX=	type 5 or type 6	table row storage indexed in ascending (A) or descending (D) order 0151*001 CUST-NUM IX=A
COLUMN=	type 5 or type 6	absolute field placement for tables extending beyond column 9999 0151 NAME COL=13001
T/D	OUTPUT	a totals-only (T) or a details-only (D) report
COMMENT=' '	OUTPUT	ASF comment field update
AREA=	OUTPUT	ASF area name field update
ONLINE=YES/NO	OUTPUT	Whether to create dialogs and maps for table viewing
DISPLAY=YES/NO	OUTPUT	The ability to retrieve data from the table
LOAD=YES/NO	OUTPUT	The ability to store data on the table
CHANGE=YES/NO	OUTPUT	The ability to update data from the table
ERASE=YES/NO	OUTPUT	The ability to erase rows of data

►► For more information about ASF, refer to the *CA-IDMS ASF User Guide*.

8.1.3 Demonstration (1): Creating a table from a sequential file

Objective: This example creates a table from a sequential file containing customer information. The rows in the table are placed in ascending order based on the customer number (CUST-NUM). Generated data table reports that appear in the Input Parameter and Run Time Message Listing are shown.

Parameters

Parameters	Description
IN 80 F 4000 PS(TAPE)	IN 80 F 4000 PS(TAPE) defines a sequential input file with fixed length 80- byte records, a block size of 4000 bytes, which is stored on tape.
REC CUST-NUM 1 5 REC CUST-NAME 19 20	REC parameters define the fields of the sequential file.
01OUTPUT TABLE=CUSTOMER-FILE TYPE=CREATE USER=DOC1 PW=DOC1 *CATALOG=ASFDICT ONLINE=YES	OUTPUT identifies the OUTPUT parameter, which specifies options to define the output table.
	TABLE=CUSTOMER-FILE names the table. TABLE= must be the first keyword on the OUTPUT parameter.
	TYPE=CREATE defines a new table. TYPE= must be the second keyword on the OUTPUT parameter.
	USER=DOC1 PW=DOC1 identifies the user and password for signing on to the catalog.
	CATALOG=ASFDICT identifies the catalog (dictionary) containing the table definition.
	ONLINE=YES creates dialogs and maps to view the table online through ASF.
0151*001 CUST-NUM IX=A 0151*002 CUST-NAME	*001 and *002 , specify the sequence of column placement.
	CUST-NUM and CUST-NAME are the column names. The fields named on the type 5 parameters determine the data type and column size.
	IX=A indexes the rows in ascending order on customer number.

Complete code

```
col. 2
↓
REC CUST-NAME 19 20
01OUTPUT TABLE=CUSTOMER-FILE TYPE=CREATE USER=DOC1 PW=DOC1
*CATALOG=ASFDICT ONLINE=YES
0151*001 CUST-NUM IX=A
0151*002 CUST-NAME
```

Results:

8.1 Creating tables

```
12/18/99          INPUT PARAMETER LISTING C8607M V15.0 PAGE   2
*****
OUTPUT  DATA TABLE
*****
01 OUT TABLE-NAME: CUSTOMER-FILE
        FUNCTION TYPE: CREATE DETAILS ONLY      COMMIT:    100
        LR-NAME: CUSTOMER-FILE                 SUBSCHEMA: RU000276
        USER: QAC                           OWNER: QAC
        CATALOG: QASFDICT                  SYSCTL:   SYSCTL     LOCATION:
        AREA: IDMSR-AREA2                COMMENT:
        SIZE- PRIMARY:                   SECONDARY: MAXIMUM:
        ONLINE: Y                      DISPLAY: Y      CHANGE: Y      ERASE: Y
*****
EDIT  LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS...
*****
01 5   1      *001  CUST-NUM
01 5   1      *002  CUST-NAME
I TABLE SUCCESSFULLY GENERATED: CUSTOMER-FILE
EXTRACT WILL BE PERFORMED
PROFILE OPTION IN EFFECT: RELEASE = 6
```

```
12/18/99          RUN TIME MESSAGES      C8607M V15.0 PAGE   1
*****
***** END OF FILE *****
17 INPUT RECORDS READ
DATA TABLE UPDATE STATISTICS


| REPORT | FUNCTION | LR-NAME       | TABLE DEFINITION | ROWS STORED |
|--------|----------|---------------|------------------|-------------|
| 01     | CRE      | CUSTOMER-FILE | GENERATED        | 17          |


```

8.1.4 Demonstration (2): Creating a table from the database

Objective: This example creates a table from the employee database. Column headings different from the database field names are created by moving field names to work fields. The work field names are specified on the type 5 lines.

Parameters

Parameters	Description
DATABASE DICTNAME=TSTDICTIONARY	DICTNAME=TSTDICTIONARY defines the dictionary containing the subschema definition.
IN DB SS=EMPSS01	IN DB SS=EMPSS01 specifies database input and identifies the subschema.
PATHAA EMPLOYEE	PATHAA EMPLOYEE defines the database path.
<pre>01OUTPUT TABLE=EMPLOYEES TYPE=CREATE USER=DOC1 PW=DOC1 *CATALOG=ASFDICT ONLINE=YES</pre>	OUTPUT identifies the OUTPUT parameter, which specifies options to define the output table. TABLE=EMPLOYEES names the table. TABLE= must be the first keyword on the OUTPUT parameter. TYPE=CREATE specifies that a new table is to be defined. TYPE= must be the second keyword on the OUTPUT parameter. USER=DOC1 PW=DOC1 identifies the authorized user and password for signing on to the catalog. CATALOG=ASFDICT identifies the catalog containing the table definition. ONLINE=YES creates dialogs and maps to view the table online through ASF.
<pre>077010 MOVE EMP-ID-0415 TO ID 077 MOVE EMP-NAME-0415 TO NAME</pre>	MOVE places the database fields into work fields having different names. (The work-field names match those listed on type 5 parameters.)

Complete code

```
col. 2
↓
DATABASE DICTNAME=TSTDICT
IN DB SS=EMPSS01,EMPSCHM,100
PATHAA EMPLOYEE
070OUTPUT TABLE=EMPLOYEES TYPE=CREATE USER=DOC1 PW=DOC1
*CATALOG=ASFDIR ONLINE=YES
070 ID
070 NAME '
0751*001 ID
0751*002 NAME
077010 MOVE EMP-ID-0415 TO ID
077      MOVE EMP-NAME-0415 TO NAME
```

8.1.5 Demonstration (3): Creating a table from an existing table

Objective: This example selects all part-time workers from a table (ALL-EMPLOYEES) and creates a new table (PART-TIME) for part-time employees.

The required keywords for the INPUT parameter are similar to those shown earlier in this chapter for the OUTPUT parameter.

Parameters

Parameters	Description
INPUT TABLE=ALL-EMPLOYEES TYPE=COPY USER=DOC1 PW=DOC1 *CATALOG=ASFDICT	INPUT identifies the incoming data. TABLE=ALL-EMPLOYEES names the table. TABLE= must be the first keyword on the INPUT parameter. TYPE=COPY copies the EMPLOYEE table. TYPE= must be the second keyword on the INPUT parameter. USER=DOC1 PW=DOC1 identifies the user and password for signing on to the catalog (dictionary). CATALOG=ASFDICT identifies the catalog containing the table definition.
SELECT EMPLOYEES WHEN STATUS EQ 'PT'	SELECT retrieves only part-time employees.
02OUTPUT TABLE=PART-TIME TYPE=CREATE USER=DOC1 PW=DOC1 *CATALOG=ASFDICT ONLINE=YES	OUTPUT identifies the OUTPUT parameter, which includes the CREATE instruction. TABLE=PART-TIME names the new table. TABLE= must be the first keyword on the OUTPUT parameter. TYPE=CREATE specifies a new table. USER=DOC1 PW=DOC1 identifies the authorized user and password for signing on to the catalog. CATALOG=ASFDICT identifies the catalog (dictionary) containing the table definition. ONLINE=YES creates dialogs and maps to view the table online through ASF.
02SORT NAME	SORT alphabetizes the names.

Complete code

```
DATABASE DICTNAME=TSTDICT
INPUT TABLE=ALL-EMPLOYEES TYPE=COPY USER=DOC1 PW=DOC1
*CATALOG=ASFDICT
SELECT EMPLOYEES WHEN STATUS EQ 'PT'
02OUTPUT TABLE=PART-TIME TYPE=CREATE USER=DOC1 PW=DOC1
*CATALOG=ASFDICT ONLINE=YES
02SORT NAME
0251*001 ID
0251*002 NAME
```

8.1.6 Demonstration (4): Creating a totals-only table

Objective: In this example, the output table contains a summary of monthly receipts for each branch of a bank. Type 6 parameters define the name, size, and position of the table columns. One type 5 parameter accumulates daily receipts.

01OUT T ...	T specifies a totals-only report.
* ONLINE=YES	ONLINE=YES directs ASF to create dialogs and maps for this data table.
* COMMENT='TOTAL MONTHLY RECEIPTS FOR EACH BRANCH'	COMMENT=' ... ' updates the comment field on the ASF Table Definition screen.
0161*001 BRANCH FZ	0161*... parameters define the name, size, and position of the table columns.
0161*002 MONTH FZ	
0161*003 YEAR FZ	
0161*004 MONTHLY-RECEIPTS SZ=5 FP DP=2	FZ and FP specify zoned and packed numeric fields respectively.

Complete code

```
IN    80  F   80
REC   DAILY-RECEIPTS 8 10  2  DP=2
REC   DAY      18  2  2
REC   MONTH    20  2  2
REC   YEAR     22  2  2
REC   BRANCH   24  2  2
01OUT T TABLE=RECEIPT-TOTALS TYPE=CREATE USER=DOC1 PW=DOC1
*          CATALOG=ASFDIR ONLINE=YES
*          COMMENT='TOTAL MONTHLY RECEIPTS FOR EACH BRANCH'
01SORT BRANCH YEAR MONTH +
010   MONTHLY-RECEIPTS
0151*000 DAILY-RECEIPTS
0161*001 BRANCH    FZ
0161*002 MONTH     FZ
0161*003 YEAR      FZ
0161*004 MONTHLY-RECEIPTS SZ=5  FP  DP=2
018001  IF LEVL EQ 2  DROP
018     MOVE DAILY-RECEIPTS TO MONTHLY-RECEIPTS
```

8.2 Retrieving data tables

8.2.1 What you can do

You can retrieve a copy of your data table and print a listing or a report based on the table contents. The table can also be viewed or modified with ASF.

8.2.2 How to do it

A data table is retrieved by coding the following statements:

1. An **INPUT** parameter that specifies at least the following information:
 - a. The name of the table being copied (**TABLE=**)
 - b. The copy function (**TYPE=COPY**)
 - c. A valid user ID (**USER=**)
If you are not the table owner, you must have authorization to access the table.
 - d. A valid password (**PW=**)
 - e. The catalog (dictionary) in which the table definition resides (**CATALOG=**)

8.2.3 Demonstration

Objective: This report retrieves and prints a listing of the part-time employees from the PART-TIME table, which was created earlier in this chapter.

Parameters

Parameters	Description
INPUT TABLE=PART-TIME TYPE=COPY USER=DOC1 PW=DOC1 *CATALOG=ASFDICT	INPUT identifies the INPUT parameter, which includes the COPY instruction. TABLE=PART-TIME specifies the table used for input. TABLE= must be the first keyword on the INPUT parameter. TYPE=COPY specifies copying the PART-TIME table. TYPE= must be the second keyword on the INPUT parameter. USER=DOC1 PW=DOC1 identifies the user and password for signing on to the catalog. CATALOG=ASFDICT identifies the catalog that contains the table definition.
01OUTPUT 60	OUTPUT 60 specifies printed output with 60 character lines.

Complete code

```

col. 2
↓
INPUT TABLE=PART-TIME TYPE=COPY USER=DOC1 PW=DOC1
*CATALOG=ASFDICT
01OUTPUT 60
013 PART-TIME EMPLOYEE LISTING
0151*010 ID          HH 'EMPLOYEE ID'
0151*020 NAME        HH 'EMPLOYEE NAME'

```

Result:

REPORT NO. 01	PART-TIME EMPLOYEE LIST 12/19/99 PAGE	1
EMPLOYEE ID	EMPLOYEE NAME	
476	BETSY	ZEDI
51	CYNTHIA	JOHNSON
49	DOUGLAS	KAHALLY
457	HARRY	ARM

8.3 Modifying data tables

8.3.1 What you can do

You can:

- **Add** rows of data to an existing table
- **Replace** rows of data in an existing table
- **Delete** a table
- **Generate** a modified table
- Use tables located in different dictionaries, DDS nodes, or central versions

8.3.2 How to do it

When modifying an existing table, specify:

1. **The source of the input data** on the INPUT parameter
2. **The modified table name** and the function to be performed (TYPE=ADD/REPLACE/DELETE/GENERATE) on the OUTPUT parameter
3. **The columns of the table** with type 5 or type 6 parameters

Note: At least one type 5 parameter is required. A dummy type 5 parameter can be used when necessary.

8.3.3 Demonstration (1): Adding rows to an existing table

Objective: This example adds the contents of the NEW-CUSTOMER table to the CUSTOMER-LIST table. Note that the type 5 parameters must match the column definitions for the CUSTOMER-LIST table.

Parameters

Parameters	Description
INPUT TABLE=NEW-CUSTOMERS TYPE=COPY USER=DOC1 PW=DOC1 * CATALOG=ASFDICT	INPUT identifies the INPUT parameter, which includes the COPY instruction. TABLE=NEW-CUSTOMERS specifies the table that will be added to CUSTOMER-LIST. TYPE=COPY specifies copying rows from the input table NEW-CUSTOMERS.
050OUTPUT TABLE=CUSTOMER-LIST TYPE=ADD USER=DOC1 PW=DOC1 * CATALOG=ASFDICT	OUTPUT identifies the OUTPUT parameter, which includes the ADD instruction. TABLE=CUSTOMER-LIST specifies the table receiving the additional rows. TYPE=ADD adds new rows to the CUSTOMER-LIST table.
0551*001 CUST-NUMBER 0551*002 CUST-NAME 0551*003 CUST-ADDRESS	1, 2, and 3 define three table columns.

Complete code

```

col. 2
↓
INPUT TABLE=NEW-CUSTOMERS TYPE=COPY USER=DOC1 PW=DOC1
* CATALOG=ASFDICT
050OUTPUT TABLE=CUSTOMER-LIST TYPE=ADD USER=DOC1 PW=DOC1
* CATALOG=ASFDICT
0551*001 CUST-NUMBER FZ SZ=5
0551*002 CUST-NAME
0551*003 CUST-ADDRESS

```

Results: The contents of NEW-CUSTOMER:

25060SUSAN	ARMITAGE	56725 OAK STREET	PITTSFIELD	MA02956
27056ALLEN	KOHN	22651 POLK STREET	SAN FRANCISCO	CA09809
39557HENRY	TRUMBLE	2102 WASHINGTON ST	BROOKLINE	MA02147
30415MARY	SMYTH	5999 SANDY LANE	LONG BEACH	CA09743
33480VICKY	KNIGHT	5678 PINE ROAD	PORTLAND	ME06895
69879BRUNO	THOR	22002 PEACHES AVE	FRESNO	CA96543
99983ELLEN	SANDS	1 APPLE ORCHARD	BISCOE	NC64321

The contents of CUSTOMER-LIST before the ADD modification:

15060SHARON	ARMSTRONG	25 CEDAR STREET	WAYLAND	MA02756
21056AMOS	JOHNSON	22651 MASS AVENUE	SAN FRANCISCO	CA09801
29557IRWIN	TRIMBLE	102 COLBOURNE ST	BROOKLINE	MA02147
30115IRMA	DOONES	5656 SANDHILL ROAD	OCEAN CITY	CA09743
33470VICTORIA	DAY	1234 OAK HILL ROAD	EVERGREEN	MI07895
69876BRUCE	THORPE	11002 PEACHTREE LA	ATLANTA	GA76543
99083HELEN	SANTOVEC	3 APPLE ORCHARD	CHARLOTTE	NC64321

The contents of CUSTOMER-LIST after the ADD modification:

15060SHARON	ARMSTRONG	25 CEDAR STREET	WAYLAND	MA02756
21056AMOS	JOHNSON	22651 MASS AVENUE	SAN FRANCISCO	CA09801
29557IRWIN	TRIMBLE	102 COLBOURNE ST	BROOKLINE	MA02147
30115IRMA	DOONES	5656 SANDHILL ROAD	OCEAN CITY	CA09743
33470VICTORIA	DAY	1234 OAK HILL ROAD	EVERGREEN	MI07895
69876BRUCE	THORPE	11002 PEACHTREE LA	ATLANTA	GA76543
99083HELEN	SANTOVEC	3 APPLE ORCHARD	CHARLOTTE	NC64321
25060SUSAN	ARMITAGE	56725 OAK STREET	PITTSFIELD	MA02956
27056ALLEN	KOHN	22651 POLK STREET	SAN FRANCISCO	CA09809
39557HENRY	TRUMBLE	2102 WASHINGTON ST	BROOKLINE	MA02147
30415MARY	SMYTH	5999 SANDY LANE	LONG BEACH	CA09743
33480VICKY	KNIGHT	5678 PINE ROAD	PORTLAND	ME06895
69879BRUNO	THOR	2002 PEACHES AVE	FRESNO	CA96543
99983ELLEN	SANDS	1 APPLE ORCHARD	BISCOE	NC64321

8.3.4 Demonstration (2): Replacing rows of an existing table

Objective: This example updates customer numbers by replacing the rows in the NEW-CUSTOMERS table with data read in from a sequential file. Note that the type 5 parameters must match the column definitions for the CUSTOMER-LIST table.

Parameters

Parameters	Description
IN 80	IN 80 defines an 80-byte sequential file containing the new customer numbers.
05OUTPUT TABLE=NEW-CUSTOMERS TYPE=REPLACE USER=DOC1 PW=DOC1 * CATALOG=ASFDICT	OUTPUT identifies the OUTPUT parameter, which includes the REPLACE instruction. TABLE=NEW-CUSTOMERS names the table in which data is replaced. TYPE=REPLACE replaces existing table rows with new data.
0551*001 CUST-NUMBER 0551*002 CUST-NAME 0551*003 CUST-ADDRESS	1, 2, and 3 define three table columns.

Complete code

```

col. 2
↓
IN 80
REC CUST-NUMBER    1   5   2
REC CUST-NAME      6   20
REC CUST-ADDRESS   26   42
05OUTPUT TABLE=NEW-CUSTOMERS TYPE=REPLACE USER=DOC1 PW=DOC1
*      CATALOG=ASFDICT
0551*001 CUST-NUMBER
0551*002 CUST-NAME
0551*003 CUST-ADDRESS

```

Results: Previous contents of NEW-CUSTOMER:

25060SUSAN	ARMITAGE	56725 OAK STREET	PITTSFIELD	MA02956
27056ALLEN	KOHN	22651 POLK STREET	SAN FRANCISCO	CA09809
39557HENRY	TRUMBLE	2102 WASHINGTON ST	BROOKLINE	MA02147
30415MARY	SMYTH	5999 SANDY LANE	LONG BEACH	CA09743
33480VICKY	KNIGHT	5678 PINE ROAD	PORTLAND	ME06895
69879BRUNO	THOR	22002 PEACHES AVE	FRESNO	CA96543
99983ELLEN	SANDS	1 APPLE ORCHARD	BISCOE	NC64321

The contents of NEW-CUSTOMERS after REPLACE:

New data	↓			
251 SUSAN	ARMITAGE	56725 OAK STREET	PITTSFIELD	MA02956
272 ALLEN	KOHN	22651 POLK STREET	SAN FRANCISCO	CA09809
393 HENRY	TRUMBLE	2102 WASHINGTON ST	BROOKLINE	MA02147
304 MARY	SMYTH	5999 SANDY LANE	LONG BEACH	CA09743
335 VICKY	KNIGHT	5678 PINE ROAD	PORTLAND	ME06895
696 BRUNO	THOR	22002 PEACH AVE	FRESNO	CA96543
997 ELLEN	SANDS	1 APPLE ORCHARD	BISCOE	NC64321

8.3.5 Demonstration (3): Deleting a table

Objective: This example deletes the definition and data of a table containing old customer information.

IN 80

IN 80 is used to meet the CA-Culprit INPUT parameter requirement.

05OUTPUT TABLE=OLD-CUSTOMERS TYPE=DELETE
USER=DOC1 PW=DOC1
CATALOG=ASFdict

OUTPUT identifies the OUTPUT parameter, which includes the DELETE instruction.

TABLE=OLD-CUSTOMERS names the table to be deleted.

TYPE=DELETE specifies deleting the the table.

0551*001 ''

5 specifies a type 5 parameter, which is used to meet CA-Culprit's coding requirements.

057001 DROP

DROP on the type 7 parameter stops unnecessary input processing.

Complete code

```

col. 2
↓
IN 80
REC INFIELD 1 80
05OUTPUT TABLE=OLD-CUSTOMERS TYPE=DELETE USER=DOC1 PW=DOC1
*          CATALOG=ASFDICT
0551*001  '
057001   DROP

```

Results:

12/18/99		RUN TIME MESSAGE0	C8607M V15.0	PAGE	1
***** END OF FILE *****					
7 INPUT RECORDS READ					
DATA TABLE UPDATE STATISTICS					
REPORT	FUNCTION	LR-NAME	TABLE DEFINITION	ROWS STORED	
05	DEL	OLD-CUSTOMERS	DELETED	0	

8.3.6 Demonstration (4): Regenerating a table

Objective: This example modifies the existing NEW-CUSTOMERS table so that it can be viewed online through ASF. The ONLINE=YES and the TYPE=GENERATE instructions cause ASF to construct a dialog and regenerate the table.

Parameters

Parameters	Description
INPUT 80	INPUT is used to meet CA-Culprit's INPUT parameter requirement.
<pre>05OUTPUT TABLE=NEW-CUSTOMERS TYPE=GENERATE USER=DOC1 PW=DOC1 * CATALOG=ASFDICT ONLINE=YES COMMENT='REGENERATED WITH CA-Culprit'</pre>	OUTPUT identifies the OUTPUT parameter, which includes the GENERATE instruction. TABLE=NEW-CUSTOMERS identifies the table to be generated. TYPE=GENERATE regenerates the table. ONLINE=YES creates a dialog to allow viewing of NEW-CUSTOMERS online with ASF. COMMENT='REGENERATED WITH CA-Culprit' specifies the ASF comment line text for the table.
0551*001 ''	5 specifies a type 5 parameter to meet the CA-Culprit coding requirements.
057001 DROP	DROP on the type 7 parameter stops unnecessary input processing.

Complete code

```
col. 2
↓
INPUT 80
REC FIELD A 1 1
05OUTPUT TABLE=NEW-CUSTOMERS TYPE=GENERATE USER=DOC1 PW=DOC1
*      CATALOG=ASFDICT ONLINE=YES COMMENT='REGENERATED WITH CA-Culprit'
0551*001 ''
057001 DROP
```

Results: Viewing the table through ASF: Before regenerating the table:

The ASF ASEL screen

```

Computer Associates International          CAGJF0
ASEL CA - Automatic System Facility 15.0    ** Activity Selection **
DC560017 NO DIALOG EXISTS FOR GENERATED TABLE

User Name: DOC1

- PF1 - Help      - PF5 - Select Data   - PF13 - Query
- PF2 - Define Table - PF7 - Page Backward - PF14 - Signon
- PF3 - Load Data  - PF8 - Page Forward  - PA1 - Prior Level
- PF4 - Display/Change Data - PF9 - Passkey - CLEAR - Leave ASF
                                         - PF15 - Administrator

Table Name.:
Table Owner:                               Page: 1 of 1

- APPLICANT
- EMPLOYEES
- JOB
- JOBS AND APPLICANTS

```

After regenerating the table:

The ASF TDEF screen

```

Computer Associates International          CAGJF0
TDEF CA - Automatic System Facility 15.0    ** Table Definition **
DC560005 MODIFY AND/OR SELECT NEXT ACTIVITY

- PF1 - Help      - PF4 - Extended Table Definition
- PF2 - Define Columns - PF5 - Delete Table Definition
- PF3 - Generate   - PF6 - Message Screen

Table Name.: NEW-CUSTOMERS
Table Owner: DOC1                         Defn Number: 190
View/Stored: STORED                        Status.....: GENERATED
Comments...: REGENERATED WITH Culprit

Table Derivation
Source Table #1
Table Name.:
Table Owner:

Source Table #2
Table Name.:
Table Owner:

Column #1:                                 Where Column #1 EQ Column #2
Column #2

```

The first row of data in the table as viewed online through ASF:

Output online

8.3 Modifying data tables

```
COMPUTER ASSOCIATES INTERNATIONAL
PF1 = ADD; PF2 = CHANGE; PF3 = DELETE; ENTER = NEXT; CLEAR = EXIT
NEW CUSTOMERS
CUST-NUMBER          251
CUST-NAME            SUSAN    ARMITAGE
CUST-ADDRESS          56725 OAK STREET    PITTSFIELD    MA02956
```

8.4 Consolidating tables

8.4.1 What you can do

You can process data from more than one input table in a single CA-Culprit run by reading the data tables as one logical file with the TYPE=CONSOLIDATION option. In addition, rows can be selected with WHERE or SELECT/BYPASS criteria.

Table data can be consolidated from:

- The same dictionary
- Different dictionaries
- Different DDS nodes
- Different central versions

8.4.2 How to do it

To consolidate tables:

1. **Check the column definitions** of each table to be used. The definitions must be identical.
2. **Define each table on an INPUT parameter** using the keywords listed in the table below.
 - TYPE=COPY must appear on the first INPUT parameter to identify the primary data table.

The primary data table is the source for:

- The generated PATH parameter and all necessary REC parameters for the run
- The specification of selection criteria (SELECT/BYPASS or the WHERE clause)
- TYPE=CONSOL must appear on subsequent INPUT parameters to identify the secondary data tables.

The order of appearance of the INPUT parameters determines the processing order of the tables.

3. **Specify the output** by using:

- An OUTPUT parameter, if needed
- At least one type 5 parameter
- Type 6, type 7, and type 8 parameters as needed.

4. **Reference tables**, if needed, in a table consolidation run by using the reserved words shown in the table below. These reserved words are valid on SELECT/BYPASS, edit, SORT, and process parameters.

Keyword	Specifies
Required Keywords:	
TABLE=	The table name
TYPE=COPY/CONSOL	The primary and secondary tables to be consolidated
USER=	The individual having authority to sign on to the catalog (dictionary)
PW=	The user's password
OWNER=	The owner of the table
CATALOG=	The catalog or dictionary containing the table definition
Optional Keywords:	
LOCATION=	Overrides DATABASE DICTNODE=
SYSCTL= (OS/390 Users)	Names the SYSCTL file that controls the system accessing the table.
VALIDATE=FIRST/ALL	Compares column definitions of secondary tables to definitions of the primary table.
WHERE	Applies WHERE clause criteria to all tables being consolidated.

Reserved words to reference tables

Reserved word	What it is	What it does
TABLE-ID	A 3-byte zoned decimal field	Indicates the data table currently being accessed
TABLE-NAME	A 56-byte alphanumeric field	Contains the name of the table currently being accessed
SUBSCHEMA-NAME	An 8-byte alphanumeric field	Contains the subschema name of the table being accessed

8.4.3 Demonstration

Objective: This example reads employee data from three tables that reside in different central versions and different dictionaries to produce a selected listing of employees in the data processing departments of three offices.

The SYSCTL= keyword values refer to ddnames that appear in an OS/390 job control language stream:

```
//SYSTEM84 DD DSN=DBDC.SYSTEM84.SYSCTL,DISP=SHR
//SYSTEM85 DD DSN=DBDC.SYSTEM85.SYSCTL,DISP=SHR
//SYSTEM86 DD DSN=DBDC.SYSTEM86.SYSCTL,DISP=SHR
```

Parameters

Parameters	Description
<pre>INPUT TABLE=BOSTON-EMPLOYEES TYPE=COPY USER=DOC1 PW=DOC1 * OWNER=DRH CATALOG=ASFDICT SYSCTL=SYSTEM84 * WHERE DEPT-ID EQ '1234'</pre>	<p>TABLE=BOSTON-EMPLOYEES is the first required keyword on the INPUT parameter and specifies BOSTON-EMPLOYEES as the first table to be retrieved.</p> <p>TYPE=COPY copies the BOSTON-EMPLOYEES table.</p> <p>CATALOG=ASFDICT identifies the catalog that contains the table definition.</p> <p>SYSCTL=SYSTEM84 specifies the central version (OS/390) that contains ASFDICT. (VM/ESA users should use CVMACH= option.)</p> <p>WHERE DEPT-ID EQ '1234' specifies selection criteria applicable to all tables read.</p>
<pre>INPUT TABLE=CHICAGO-EMPLOYEES TYPE=CONSOL USER=DOC1 PW=DOC1 * OWNER=DDR CATALOG=TSTDICT SYSCTL=SYSTEM85</pre>	<p>TABLE=CHICAGO-EMPLOYEES is the first required keyword on the INPUT parameter and specifies CHICAGO-EMPLOYEES as a secondary table.</p> <p>TYPE=CONSOL specifies consolidation of CHICAGO-EMPLOYEES with BOSTON-EMPLOYEES.</p> <p>CATALOG=TSTDICT identifies the catalog that contains the table definition.</p> <p>SYSCTL=SYSTEM85 specifies the central version that contains TSTDICT.</p>
<pre>INPUT TABLE=DENVER-EMPLOYEES TYPE=CONSOL USER=DOC1 PW=DOC1 * OWNER=ADR CATALOG=PRODICT SYSCTL=SYSTEM86</pre>	<p>TABLE=DENVER-EMPLOYEES is the first required keyword on the INPUT parameter and specifies DENVER-EMPLOYEES as a secondary table.</p> <p>TYPE=CONSOL specifies consolidation of DENVER-EMPLOYEES with CHICAGO-EMPLOYEES.</p> <p>CATALOG=PRODICT identifies the catalog that contains the table definition.</p> <p>SYSCTL=SYSTEM86 specifies the central version that contains PRODICT.</p>

Complete code

```

col. 2
↓
INPUT TABLE=BOSTON-EMPLOYEES TYPE=COPY USER=DOC1 PW=DOC1
*      OWNER=DRH CATALOG=ASFICT SYSCTL=SYSTEM84
*      WHERE DEPT-ID EQ '1234'
INPUT TABLE=CHICAGO-EMPLOYEES TYPE=CONSOL USER=DOC1 PW=DOC1
*      OWNER=DDR CATALOG=TSTDICT SYSCTL=SYSTEM85
INPUT TABLE=DENVER-EMPLOYEES TYPE=CONSOL USER=DOC1 PW=DOC1
*      OWNER=ADR CATALOG=PRODICT SYSCTL=SYSTEM86
01OUTPUT D
013EMPLOYEES IN DATA PROCESSING
0151*010 TABLE-ID  HH 'TABLE-ID'
0151*020 TABLE-NAME SZ=20 HH 'TABLE-NAME'
0151*030 EMP-NAME  HH 'EMPLOYEE'
0151*040 JOB-TITLE HH 'TITLE'

```

Result:

REPORT NO. 01	EMPLOYEES IN DATA PROCESSING	01/14/99 PAGE	1	EMPLOYEE	TITLE
TABLE-ID	TABLE-NAME				
1	BOSTON-EMPLOYEES			JOHN SMYTH	PROGRAMMER
1	BOSTON-EMPLOYEES			MARY JONES	DBA
1	BOSTON-EMPLOYEES			JOE GREEN	PROGRAMMER
2	CHICAGO-EMPLOYEES			JOAN WHITE	DATA ENTRY CLERK
2	CHICAGO-EMPLOYEES			JAN HUBBARD	DBA
2	CHICAGO-EMPLOYEES			DAVID KELLY	REGIONAL MGR
3	DENVER-EMPLOYEES			MEL SMITH	DATA ENTRY CLERK
3	DENVER-EMPLOYEES			MARIO JENI	DBA

Table extraction statistics:

01/14/99	RUN TIME MESSAGES	C8612B V15.0 PAGE	1	
DATA TABLE EXTRACTION STATISTICS				ROWS OBTAINED FROM IDMS/DB
TABLE-NAME		TABLE-ID		
BOSTON-EMPLOYEES		001		3
CHICAGO-EMPLOYEES		002		3
DENVER-EMPLOYEES		003		2
CONSOLIDATION TOTALS				8

***** END OF FILE *****
8 INPUT RECORDS READ

8.4 Consolidating tables

Advanced Topics

Chapter 9. Totals Processing Techniques

9.1 Calculations on accumulated totals	9-4
9.1.1 What you can do	9-4
9.1.2 How to do it	9-4
9.1.3 Demonstration	9-4
9.2 Multiple-level subtotals	9-6
9.2.1 What you can do	9-6
9.2.2 How to do it	9-6
9.2.3 Demonstration	9-6
9.3 Creating a sparse listing	9-10
9.3.1 What you can do	9-10
9.3.2 How to do it	9-10
9.3.3 Demonstration	9-10
9.4 Obtaining work field values	9-13
9.4.1 Demonstration	9-13

CA-Culprit processes data in two input/output phases. The first phase executes type 7 logic against the input data and outputs a temporary work file (the extract file). The second phase reads the extract file after it has been sorted and performs final calculations on accumulated totals, executes control breaks, and generates report output.

Within the CA-Culprit final processing phase you can perform calculations on accumulated totals, obtain multiple level subtotals, and access current values of work fields that do not appear on SORT or type 5 parameters.

A detailed description of the CA-Culprit final processing phase is presented in Appendix G, “How Totals Processing Works” on page G-1.

9.1 Calculations on accumulated totals

9.1.1 What you can do

You can obtain subtotals, perform calculations on totals, and perform logical operations on totals.

9.1.2 How to do it

Use:

- **Control breaks** on the SORT parameter to obtain subtotals
- **Type 8** logic to:
 - Test control break levels
 - Issue TAKE or DROP instructions
 - Perform total-time computations and logical operations
- **Type 6** parameters to select the total lines to be printed

9.1.3 Demonstration

Objective: This report prints account balances for several branch offices and computes an average branch balance.

Procedure

- The branch number is the sort key value obtained when a control break occurs on **BRANCH** (LEVL=1).
- Branch account totals are automatically obtained by the control break.
- An average balance is computed in type 8 logic from the automatic grand total amount and a count of the branch control breaks.

Complete code

```

col. 2
↓
REC CURRENT-BAL    4   8   2 DP=2
01OUT T
01SORT BRANCH 0
010 PRINT1 'BRANCH' $Control break for branch number
010 COUNT
010 AVERAGE DP=2
013 SUMMARY REPORT OF AVERAGE TRANSACTIONS
0141*001 '
0151*010 CURRENT-BAL
01610020 PRINT1
01610027 BRANCH $Prints BRANCH value
01610031 'TOTAL BALANCE'
01610045 CURRENT-BAL SZ=11
016200310 'TOTAL BALANCE'
01620045 CURRENT-BAL SZ=11 $Prints branch totals
016300310 'AVERAGE PER BRANCH'
01630050 AVERAGE SZ=11 $Prints the average
018 IF LEVL EQ 2 200 $Test for grand-total time
018 COUNT + 1 COUNT
018 TAKE 1 $Prints branch information
018200 CURRENT-BAL / COUNT AVERAGE $Uses the grand total of CURRENT-BAL
018     TAKE (2 3) $Prints type 6 lines 2 and 3

```

Result:

REPORT NO. 01	SUMMARY REPORT OF AVERAGE TRANSACTIONS	09/27/99 PAGE 1
	BRANCH 001 TOTAL BALANCE	999,999.99
	BRANCH 010 TOTAL BALANCE	1,358,349.11
	BRANCH 011 TOTAL BALANCE	345,678.90
	BRANCH 020 TOTAL BALANCE	345,575.18
	TOTAL BALANCE	3,049,603.18
	AVERAGE PER BRANCH	762,400.80

9.2 Multiple-level subtotals

9.2.1 What you can do

You can obtain automatically calculated subtotals for each control-break level.

After the extract file is sorted, records are read into the output (CULE) phase of CA-Culprit. When a control break occurs, the type 8 logic is executed and the subtotal accumulators for the level with the control break are reinitialized after the break. If no control break occurs, the values of the numeric fields appearing on type 5 lines are added to the accumulators.

9.2.2 How to do it

1. Code the **control break** on the **SORT** parameter.
2. Specify the type 5 parameter field name on a **type 6** parameter.
3. Test control break levels in **type 8** logic.
4. Specify the type 6 line to be printed with the **TAKE** command in type 8 logic.

9.2.3 Demonstration

Objective: This report lists the sales for each plant and prints subtotals by plant and division.

Procedure

- **Control break levels** are printed on the report to show where accumulator initialization occurs. (The contents of the accumulators during processing are shown in the table below.)
- **COUNT**, a numeric work field on a type 5 line, produces a subtotal of the number of records processed:
 - The subtotal for each plant is obtained on the first type 6 line.
 - The subtotal for each division is obtained on the second type 6 line.
 - The total number of records for all divisions is obtained on the third type 6 line.
- **DIVISION** and **PLANT**, are alphanumeric sort keys:
 - The sort key value (division and plant number) is obtained at the appropriate control break levels.
 - Omission of DIVISION on the third type 6 line results in the division number on the grand total line being left out.
- **AMOUNT** is a numeric input item:

- On a type 5 line, AMOUNT prints on detail lines.
- On the first type 6 line, AMOUNT prints the subtotal for each plant.
- On the second type 6 line, AMOUNT prints the subtotal for each division.
- On the third type 6 line, AMOUNT prints the grand total.

Accumulated total values: The figure below shows the total values that occur during the processing of the code for the sales report:

Accumulated total values for sales report

SORT KEY VALUES		TYPE 5 FIELDS			
LEVEL 2 DIVISION	LEVEL 1 PLANT	AMT INPUT	LEVEL 3	LEVEL 2	LEVEL 1
01	01	50	50	50	50
01	01	20	70	70	70
01	01	10	80	80	80*
01	02	40	120	120	40
01	02	50	170	170*	90*
02	02	20	190	20	20
02	02	0	190	20	20
02	02	40	230	60*	60*
06	03	40	270	40	40
* denotes initialization of the bucket to zero after the control break					
LEVEL 1 -- PLANT LEVEL 2 -- DIVISION LEVEL 3 -- GRAND TOTAL					

The values of the accumulators during the output processing of the Sales by Division and Plant code are incremented until a control break occurs. A control break causes initialization of the accumulators for the level having the control break.

Complete code

```

col. 2
↓
REC PLANT      3  2
REC AMOUNT     5  4  2  DP=2
REC TYPE       9  1
REC COMPL-DATE 10  6  2
01SORT DIVISION 1 PLANT 0
013 SALES BY DIVISION AND PLANT
010 COUNT 1
014100200 'DIVISION NUMBER:'
01410037  DIVISION
0151*0000 COUNT                               $Printing on detail line suppressed
0151*010  PLANT
0151*020 AMOUNT F2 SZ=7
0161*0050 DIVISION
0161*010 PLANT          HH 'PLANT'
0161*020 AMOUNT SZ=7 HH 'AMOUNT'
0161*021 COUNT SZ=2 HH 'COUNT TOTALS'
0162*0050 DIVISION
0162*020 AMOUNT SZ=7
0162*021 COUNT SZ=2
0163*005 'GRAND TOTAL:'
0163*020 AMOUNT SZ=7
0163*021 COUNT SZ=2
018    IF LEVL EQ 3  300      $Grand total test
018    IF LEVL EQ 2  200      $Division total test
018100 TAKE 1                $Print first type 6 line
018200 TAKE 2                $Print second type 6 line
018300 TAKE 3                $Print third type 6 line

```

Result:

REPORT NO. 01	SALES BY DIVISION AND PLANT	09/27/99 PAGE	1
DIVISION NUMBER: 01			
	PLANT	AMOUNT	COUNT TOTALS
	01	50.00	
	01	20.00	
	01	20.00	
01	01	90.00	3
	02	40.00	
	02	50.00	
01	02	90.00	2
01		180.00	5

REPORT NO. 01	SALES BY DIVISION AND PLANT	09/27/99 PAGE	2
DIVISION NUMBER: 02			
	PLANT	AMOUNT	COUNT TOTALS
	02	20.00	
	02	.00	
	02	40.00	
02	02	60.00	3
02		60.00	3

REPORT NO. 01	SALES BY DIVISION AND PLANT	09/27/99 PAGE	3
DIVISION NUMBER: 06			
	PLANT	AMOUNT	COUNT TOTALS
	03	40.00	
06	03	40.00	1
06		40.00	1
GRAND TOTAL:		280.00	9

9.3 Creating a sparse listing

9.3.1 What you can do

You can eliminate repeated occurrences of selected fields on printed output by forcing a control break on a work field that is not tied to input data.

9.3.2 How to do it

1. Define a numeric **work field**.
2. Code the work field on a **SORT** parameter as a low-order sort key with a control break.
3. Code a type 7 statement that increments the control-break work field by 1. (A control break will then occur on each extract record when the value of the work field changes.)
4. Code **type 8** instructions for processing the control break.

9.3.3 Demonstration

Objective: This report uses a forced control break to make individual plant and division codes available for labeling print lines.

Procedure

- A numeric work field (**WORK-X**) is placed on a **SORT** parameter to force a control break each time the value of the work field changes.
- The value of WORK-X is incremented in type 7 logic.
- The control break on WORK-X causes:
 - **Type 8** statements to be processed for every extract record. Type 8 logic checks for a control break at LEVL 4, 3, or 2.
 - If one of the type 8 conditions is true, processing branches to the appropriate **type 6** subtotal or total print line.

Complete code

```

col. 2
↓
REC PLANT      3 2
REC AMOUNT     5 4 2 DP=2
01SORT DIVISION 0 PLANT 0 WORK-X + $Forced control break
013 SALES BY PLANT AND DIVISION
010 COUNT 1
010 PLANT-SAVE ' '
010 DIV-SAVE ' '
010 WORK-X
010UT T
0151*0000 COUNT
0151*020 AMOUNT F2 SZ=7
017100 WORK-X + 1 WORK-X           $Increment work field value
0161*005 DIVISION    HH 'DIVISION'   $Headings and first detail line
0161*010 PLANT       HH 'PLANT'
0161*020 AMOUNT     SZ=7  HH 'AMOUNT'
0161*025 ' '          HH 'COUNT'
0162*0100 PLANT                      $Plant subtotal
0162*020 AMOUNT SZ=7
0162*025 COUNT SZ=2                  $Plant records processed
0163*0050 DIVISION                   $Division subtotal
0163*021 AMOUNT SZ=7
0163*025 COUNT SZ=2                  $Division records processed
0164*0050 'GRAND TOTAL'            $Grand total
0164*022 AMOUNT SZ=7
0164*025 COUNT SZ=2                  $Total number of record processed
0165*020 AMOUNT SZ=7                  $Unlabeled detail lines
018010 IF LEVL = 4 400             $Go to grand total print routine
018020 IF LEVL = 3 300             $Go to the division print routine
018030 IF LEVL = 2 200             $Go to the plant print routine
018040 IF DIVISION = DIV-SAVE AND PLANT = PLANT-SAVE 100
018050 MOVE PLANT TO PLANT-SAVE
018055 MOVE DIVISION TO DIV-SAVE
018060 TAKE 1
018100 TAKE 5
018200 TAKE 2
018300 TAKE 3
018400 TAKE 4

```

Result:

9.3 Creating a sparse listing

REPORT NO. 01	SALES BY PLANT AND DIVISION	09/27/99 PAGE	1	COUNT
DIVISION	PLANT	AMOUNT		
01	01	50.00 20.00 20.00		
	01	90.00		3
01	02	40.00 50.00		
	02	90.00		2
01			180.00	5
02	02	20.00 40.00		
	02	60.00		3
02			60.00	3
06	03	40.00		
	03	40.00		1
06			40.00	1
GRAND TOTAL			280.00	9

9.4 Obtaining work field values

Work fields that do not appear on SORT or type 5 parameters are not part of the extract record. Any references made to such work fields during output processing return only the most current value of the work field.

To obtain the value of a subscripted work field, refer to 10.8, “Obtaining sort-key values” on page 10-22.

9.4.1 Demonstration

Objective: The Summary Report of Account Totals is repeated here to show the use of the work field AVERAGE.

Procedure

- AVERAGE is computed in type 8 logic.
- The computation results are stored in the work field AVERAGE.
- The current value of AVERAGE is printed by using a type 6 parameter reference.

Complete code

```

col. 2
↓
IN 80 F 320 PS(TAPE)
REC BRANCH      1   3
REC CURRENT-BAL 4   8   2  DP=2
01OUT T
01SORT BRANCH,0
010 PRINT1  'BRANCH'
010 COUNT
010 AVERAGE DP=2          $Work field holding computation results
013 SUMMARY REPORT OF AVERAGE TRANSACTIONS
0141*001  '
0151*010  CURRENT-BAL
01610020  PRINT1
01610027  BRANCH
01610031  'TOTAL BALANCE'
01610045  CURRENT-BAL SZ=11
016200310 'TOTAL BALANCE'
01620045  CURRENT-BAL SZ=11
016300260 'AVERAGE PER BRANCH'
01630045  AVERAGE SZ=11          $Print the value of the work field
018     IF LEVL EQ 2  200
018     COUNT + 1 COUNT
018     TAKE 1
018200  CURRENT-BAL / COUNT AVERAGE $Compute the work field value
018     TAKE (2 3)

```

Result:

9.4 Obtaining work field values

REPORT NO. 01	SUMMARY REPORT OF AVERAGE TRANSACTIONS	09/27/99 PAGE	1
BRANCH 001 TOTAL BALANCE 999,999.99			
BRANCH 010 TOTAL BALANCE 1,358,349.11			
BRANCH 011 TOTAL BALANCE 345,678.90			
BRANCH 020 TOTAL BALANCE 345,575.18			
TOTAL BALANCE 3,049,603.18			
AVERAGE PER BRANCH 762,400.80			

Chapter 10. Using Subscripts

10.1	Explicit subscripts	10-4
10.1.1	What you can do	10-4
10.1.2	How to do it	10-4
10.1.3	Demonstration (1): Using a literal subscript	10-4
10.1.4	Demonstration (2): Using a counter as a subscript	10-6
10.2	Zero subscripts	10-8
10.2.1	What you can do	10-8
10.2.2	How to do it	10-8
10.2.3	Demonstration	10-8
10.3	Fixed repeating fields	10-11
10.3.1	What you can do	10-11
10.3.2	How to do it	10-11
10.3.3	Demonstration	10-11
10.4	Variable repeating groups	10-13
10.4.1	What you can do	10-13
10.4.2	How to do it	10-13
10.4.3	Demonstration	10-13
10.5	Floating fields	10-15
10.5.1	What you can do	10-15
10.5.2	How to do it	10-15
10.5.3	Demonstration	10-15
10.6	Obtaining accumulated totals	10-17
10.6.1	What you can do	10-17
10.6.2	How to do it	10-17
10.6.3	Demonstration	10-17
10.7	Obtaining specific field values	10-19
10.7.1	What you can do	10-19
10.7.2	How to do it	10-19
10.7.3	Demonstration	10-19
10.8	Obtaining sort-key values	10-22
10.8.1	What you can do	10-22
10.8.2	How to do it	10-22
10.8.3	Demonstration	10-22

Subscripts define and provide access to particular occurrences of repeating data fields. References can be made to work field occurrences, segments of a field, and fixed or variable repeating input fields.

10.1 Explicit subscripts

10.1.1 What you can do

You can reference a field occurrence directly by:

- Defining an explicit number of occurrences in a given work field:

`010 CHARACTERS.3 'A' 'B' 'C'`

- Referencing a specific work field occurrence by a number or integer name:

`0151*010 CHARACTERS.2` (By a number)

`010 INDEX` (By an integer)
`0151*020 CUSTOMER-NAME.INDEX`

10.1.2 How to do it

Code a work field parameter with:

1. The name of the repeating field
2. A period (.) immediately followed by a numeric literal that specifies the number of repeating elements
3. Optional initialization of each occurrence of the field

10.1.3 Demonstration (1): Using a literal subscript

Objective: This report lists the interest due on loans. Based on the size of the loan, there are three possible interest rates. Literal subscripts are used to reference each specific interest rate occurrence.

Procedure

- Interest rates are stored in a work field **INTEREST.3**.
- **Type 7** logic references each occurrence of the work field.
- **INTEREST.3** has a **DP=4** specification, which translates each occurrence into a 16-byte packed decimal.
- The size of the printed interest rate is controlled by using **SZ=4** on line **0151*030**.

Complete code

```
col. 2
↓
IN 200 F 400 PS(TAPE)
REC NAME      5 20
REC LOAN     160 7 3 DP=2
01OUT D
010 INTEREST.3 DP=4 0.0525  0.0550  0.0575 $The subscripted work field
010 AMOUNT DP=2
010 INT DP=4
0151*010 NAME           HH 'CUSTOMER'
0151*020 LOAN   SZ=8    HH 'LOAN AMOUNT'
0151*030 INT   SZ=4    DP=4  HH 'INTEREST' 'RATE'
0151*040 AMOUNT SZ=7    DP=2  HH 'INTEREST' 'AMOUNT'
017   IF LOAN GT 50000.00 100
017   IF LOAN EQ (20000.00 TO 49999.99) 200
017   IF LOAN LT 20000.00 300
017100 COMPUTE ROUND (LOAN X INTEREST.1) AMOUNT $Use the first occurrence
017   MOVE INTEREST.1 TO INT
017   TAKE
017200 COMPUTE ROUND (LOAN X INTEREST.2) AMOUNT $Use the second occurrence
017   MOVE INTEREST.2 TO INT
017   TAKE
017300 COMPUTE ROUND (LOAN X INTEREST.3) AMOUNT $Use the third occurrence
017   MOVE INTEREST.3 TO INT
017   TAKE
```

Result:

CUSTOMER		LOAN AMOUNT	RATE	AMOUNT
JUNE	BLOOMER	15,000.00	.0575	862.50
EDWARD	HUTTON	44,000.00	.0550	2,420.00
RUPERT	JENSON	82,000.00	.0525	4,305.00
MARIANNE	KIMBALL	45,000.00	.0550	2,475.00
DORIS	KING	14,500.00	.0575	833.75
BRIAN	NICEMAN	14,000.00	.0575	805.00
HERBERT	CRANE	75,000.00	.0525	3,937.50
JANE	FERNDALE	22,500.00	.0550	1,237.50
GEORGE	FONRAD	14,750.00	.0575	848.13
ROBIN	GARDNER	14,000.00	.0575	805.00
DOUGLAS	KAHALLY	20,000.00	.0550	1,100.00
TERENCE	KLWELLEN	43,000.00	.0550	2,365.00
SANDY	KRAMMER	14,000.00	.0575	805.00
HERBERT	LIPSICH	18,500.00	.0575	1,063.75
NANCY	TERNER	13,000.00	.0575	747.50
BETH	CLOUD	52,750.00	.0525	2,769.38
ALAN	DONOVAN	33,500.00	.0550	1,842.50
DANIEL	MOON	72,000.00	.0525	3,780.00
ROY	ANDALE	33,500.00	.0550	1,842.50
HARRY	ARM	46,000.00	.0550	2,530.00
C.	BREEZE	38,000.00	.0550	2,090.00
CAROLYN	CROW	37,500.00	.0550	2,062.50
BURT	LANCHESTER	54,500.00	.0525	2,861.25
RENE	MAKER	85,000.00	.0525	4,462.50
RICHARD	MUNYON	36,000.00	.0550	1,980.00
RICHARD	WAGNER	47,000.00	.0550	2,585.00
TOM	FITZHUGH	13,000.00	.0575	747.50
CYNTHIA	JOHNSON	13,500.00	.0575	776.25
MADELINE	ORGRATZI	39,000.00	.0550	2,145.00
ELEANOR	PEOPLES	80,000.00	.0525	4,200.00
MICHAEL	ANGELO	18,000.00	.0575	1,035.00
MONTE	BANK	80,000.00	.0525	4,200.00
CHARLES	BOWER	38,500.00	.0550	2,117.50
JOCK	JACKSON	34,000.00	.0550	1,870.00
CAROL	MCDougall	18,000.00	.0575	1,035.00
LAURA	PENMAN	39,000.00	.0550	2,145.00
BETSY	ZEDI	37,000.00	.0550	2,035.00
TERRY	CLOTH	38,000.00	.0550	2,090.00
PHINEAS	FINN	45,000.00	.0550	2,475.00
JOE	KASPAR	31,000.00	.0550	1,705.00
MARK	TIME	33,000.00	.0550	1,815.00
ROGER	WILCO	80,000.00	.0525	4,200.00
JANE	DOUGH	33,000.00	.0550	1,815.00
JAMES	GALLWAY	33,000.00	.0550	1,815.00
JENNIFER	GARFIELD	65,000.00	.0525	3,412.50
PERCY	GRANGER	34,500.00	.0550	1,897.50
VLADIMIR	HEAROWITZ	33,000.00	.0550	1,815.00
JAMES	JACOBI	55,000.00	.0525	2,887.50
JULIE	JENSEN	37,000.00	.0550	2,035.00
LARRY	LITERATA	37,500.00	.0550	2,062.50
KATHERINE	O'HEARN	42,500.00	.0550	2,337.50
RALPH	TYRO	20,000.00	.0550	1,100.00
HENRIETTA	HENDON	240,000.00	.0525	12,600.00
THEMIS	PAPAZEUS	100,000.00	.0525	5,250.00
JOHN	RUPEE	80,000.00	.0525	4,200.00
ROBBY	WILDER	90,000.00	.0525	4,725.00

10.1.4 Demonstration (2): Using a counter as a subscript

Objective: This report lists data occurrences that are stored in a work field.

Procedure: The value of the counter **INDEX** is used as a subscript on a type 5 parameter.

Complete code

```
col. 2
↓
IN 80
REC CODE 1 2 2
020 FOREIGN-MAKES.30  'ROLLS ROYCE' 'ALPHA ROMEO' 'FIAT'
*'VOLKSWAGEN' 'PORCHE' 'LANCIA' 'BMW' 'AUDI' 'DATSUN' 'TOYOTA'
*'MERCEDES' 'MASERATI' 'LADA' 'SAAB' 'HONDA' 'MAZDA' 'RENAULT'
*'VOLVO' 'LAMBORGHINI' 'TRIUMPH' 'LOTUS' 'FERRARI' '
020 INDEX                      $Counter work field
02OUT 80
0251*010 FOREIGN-MAKES.INDEX
027010  INDEX + 1 INDEX          $Increment counter
027020  IF INDEX GT 30  STOP-RPT   $Check for subscript value
027    RELS
027    B 10
```

Result:

```
ROLLS ROYCE
ALPHA ROMEO
FIAT
VOLKSWAGEN
PORCHE
LANCIA
BMW
DATSUN
MERCEDES
SAAB
HONDA
MAZDA
VOLVO
LAMBORGHINI
TRIUMPH
LOTUS
FERRARI
```

10.2 Zero subscripts

10.2.1 What you can do

You can define an alphanumeric work field, redefine adjacent storage to contain segments of the work field, and reference the segments with explicit subscripts.

10.2.2 How to do it

1. Define the subscripted work field by coding a work field parameter (0) with:
 - a. The name of the primary work field that redefines the adjacent fields
 - b. A period (.) followed immediately by zero (0)
 - c. Spaces (enclosed in single quotation marks) to indicate the length of the primary work field

```
010 NAME-0.0 '
```
2. Redefine the storage area that immediately follows the zero-subscripted work field by coding one or more work field parameters (in alphabetical or numerical order):
 - a. The prefix of the field name being defined
 - b. Spaces (enclosed in single quotation marks) to indicate the length of the redefined field:

```
010 NAME-0.0 '
010 NAME-1 '
```
3. Reference a zero-subscripted work field by using:
 - a. The name of the work field
 - b. An explicit subscript

```
010 FIELD-0.0 '
010 FIELD-1 '
010 FIELD-2 '
010 FIELD-3 '
0151*010 FIELD-0.1 $References the first occurrence of FIELD-0.0
```

10.2.3 Demonstration

Objective: This report lists the name, password, and address of employees. The passwords are created from the first, third, and fifth letters of the employees' first names.

Procedure

- The NAME-1 work field is redefined into 1-byte segments.
- WRKFLD-1, -2, and -3 are combined into a single field (WRKFLD-0.0).
- The five address segments are combined into one field (ADDRESS-0.0).

Complete code

```

col. 2
↓
IN 200
REC NAME      5    20
REC FIRST-NAME 5    10
REC LAST-NAME 15   10
REC STREET     30   20
REC CITY       50   15
REC STATE      65   2
REC ZIP-CODE   67   5
SEL STATE EQ 'MA'
01SORT LAST-NAME
010 NAME-0.0 ' '          $Redefines NAME-1 into 1-byte occurrences
010 NAME-1 ' '           $Holds the first name
010 WRKFLD-0.0 ' '       $3 bytes to hold the letters for the password
010 WRKFLD-1 ' '         $Holds a single letter
010 WRKFLD-2 ' '
010 WRKFLD-3 ' '
010 ADDRESS-0.0 ' '
010 ADDRESS-1 'STREET'
010 ADDRESS-2 'CITY'
010 ADDRESS-3 ' '
010 ADDRESS-4 ' '
010 ADDRESS-5 'ZIP '
0151*010 NAME      HH 'NAME'
0151*020 WRKFLD-0.1 HH 'PASSWORD'
0151*030 ADDRESS-0.1 HH ' ' ADDRESS'
017020 MOVE FIRST-NAME TO NAME-1
017    MOVE NAME-0.1 TO WRKFLD-1 $First letter of the password
017    MOVE NAME-0.3 TO WRKFLD-2 $Second letter of the password
017    MOVE NAME-0.5 TO WRKFLD-3 $Third letter of the password
017    MOVE STREET TO ADDRESS-1
017    MOVE CITY TO ADDRESS-2
017    MOVE STATE TO ADDRESS-3
017    MOVE ZIP-CODE TO ADDRESS-5

```

Result:

NAME	PASSWORD	ADDRESS
ROY ANDALE	RY	44 TRIGGER RD FRAMINGHAM MA 03461
MICHAEL ANGELO	MCA	507 CISTINE DR WELLESLEY MA 01568
HARRY ARM	HRY	77 SUNSET STRIP NATICK MA 02178
MONTE BANK	MNE	45 EAST GROVE DR HANIBAL MA 02415
JUNE BLOOMER	JN	14 ZITHER TERR LEXINGTON MA 01675
CHARLES BOWER	CAL	30 RALPH ST WELLESLEY MA 01568
C. BREEZE	C	200 NIGHTINGALE ST FRAMINGHAM MA 03461
TERRY CLOTH	TRY	5 ASPHALT ST EASTON MA 05491
BETH CLOUD	BT	3456 PINKY LN NATICK MA 02178
CAROLYN CROW	CRL	891 SUMMER ST WESTWOOD MA 02090
ALAN DONOVAN	AA	6781 CORNWALL AVE MELROSE MA 02176
JANE DOUGH	JN	15 LOCATION DR NEWTON MA 02456
JANE FERNDALE	JN	60 FOREST AVE NEWTON MA 02576
PHINEAS FINN	PIE	79 HIGH ST WESTON MA 04371
TOM FITZHUGH	TM	450 THRUWAY ST MANSFIELD MA 03458
GEORGE FONRAD	GOG	99 VERDE ST STOUGHTON MA 02456
JAMES GALLWAY	JMS	15 DAWSON ST MEDFORD MA 02432
ROBIN GARDNER	RBN	68 75TH ST LOWELL MA 02945
JENNIFER GARFIELD	JNI	110A FIRTH ST STONEHAM MA 02928
PERCY GRANGER	PRY	14 BYTE WAY TAUNTON MA 02678
VLADIMIR HEAROWITZ	VAI	19 TERRACE TERR TAUNTON MA 02678
HENRIETTA HENDON	HNI	16 HENDON DR WELLESLEY MA 02198
EDWARD HUTTON	EWR	781 CROSS ST MELROSE MA 02176
JOCK JACKSON	JC	65 BROWN ST WALTHAM MA 01476
JAMES JACOBI	JMS	555 JAKAS DR GROVER MA 02976
JULIE JENSEN	JLE	15 THINGER AVE SAUGUS MA 02276
RUPERT JENSON	RPR	999 HARVEY ST MELROSE MA 02176
CYNTHIA JOHNSON	CNH	17 MANIFESTO DR WALPOLE MA 02546
DOUGLAS KAHALLY	DUL	56 SPELLING AVE READING MA 02317
JOE KASPAR	JE	133 NORTH ST WESTBORO MA 03156
MARIANNE KIMBALL	MRA	561 LEXINGTON AVE LITTLETON MA 01239
DORIS KING	DRS	716 MORRIS ST MELROSE MA 02176
TERENCE KLWELLEN	TRN	12 RAUNCH ST., APT1 DEDHAM MA 02034
SANDY KRAAMER	SNY	56 NASTY WAY WESTWOOD MA 02090
BURT LANCHESTER	BR	45 PINKERTON AVE WALTHAM MA 01476
HERBERT LIPSICH	HRE	6 FORENOON ST WALDEN MA 02986
LARRY LITERATA	LRY	123 SATURDAY TERR WILMINGTON MA 02476
RENE MAKER	RN	10 DROVER DR BOSTON MA 02123
CAROL McDougall	CRL	19 URITOP DR WELLESLEY MA 01568
DANIEL MOON	DNE	16 SKYHIGH DR WESTON MA 04371
RICHARD MUNYON	RCA	17 BLACKHILL DR WESTWOOD MA 02090
BRIAN NICEMAN	BIN	60 FLORENCE AVE MELROSE MA 02176
KATHERINE O'HEARN	KTE	12 EAST SPEEN ST NATICK MA 02364
MADELINE ORGRATZI	MDL	67 RAINBOW DR KENDON MA 06182
THEMIS PAPAZEUS	TEI	234 TRANSWORLD ST NORTHBORO MA 03256
LAURA PENMAN	LUA	45 THRUSH LN WALTHAM MA 01476
ELEANOR PEOPLES	EEN	756 YELLOWSTONE DR BOSTON MA 02834
JOHN RUPEE	JH	114 WEST INDIA ST METHUEN MA 02312
NANCY TERNER	NNY	14 TYPO TERR READING MA 02317
MARK TIME	MR	44 CLOCK ST MALDEN MA 01776
RALPH TYRO	RLH	888 FORTIHE ST SINGER MA 02254
RICHARD WAGNER	RCA	677 GERMANY LN NATICK MA 02178
ROGER WILCO	RGR	671A SNOWBANK RD TYNGSBORO MA 01879
ROBBY WILDER	RBY	4567 E. GROWTH ST SOUTHBORO MA 03145
BETSY ZEDI	BTY	34 VALE AVE SOUTHBORO MA 03145

10.3 Fixed repeating fields

10.3.1 What you can do

You can retrieve input data items that occur a fixed number of times.

10.3.2 How to do it

Define the related items as a group of similar elements before processing takes place.

1. Define the **group** of repeating fields by coding one REC parameter with:
 - a. A group name for the repeating fields
 - b. The start position of the group
 - c. The keyword **GROUP**
 - d. A 2-character group identifier
 - e. The total length (in bytes) of the group, followed by a period (.)
 - f. The number of occurrences in the group
2. Define the **elements** of the group by coding a REC parameter for each element of the group with:
 - a. The name of the element
 - b. The start position *within the group*
 - c. The length of the element
 - d. The keyword **ELMNT**
 - e. The same 2-character identifier as the group to which the element belongs
3. Reference the elements by using explicit subscripts.

10.3.3 Demonstration

Objective: This report prints interest rates in three columns.

Procedure: The input records contain three repeating rates:

- **INTEREST-RATE-DATA** defines the group.
- **INTEREST-RATE** defines the individual elements.
- Explicit subscripts reference the elements on type 5 parameters.

Complete code

```
col. 2
↓
IN 80
REC INTEREST-RATE-DATA 1      GROUP AA 4.3
REC INTEREST-RATE      1 4 2  DP=4 ELMNT AA
000UT 80 D
003 INTEREST RATE TABLE
0051*010 INTEREST-RATE.1 HH '' RATE CODE 1' '(ACCOUNT TYPE 1)'
0051*020 INTEREST-RATE.2 HH 'RATE CODE 2' '(ACCOUNT TYPE 2)'
0051*030 INTEREST-RATE.3 HH 'RATE CODE 3' '(ACCOUNT TYPE 3)'
```

Result

REPORT NO. 00	INTEREST RATE TABLE	09/27/90 PAGE 1
RATE CODE 1 (ACCOUNT TYPE 1)	RATE CODE 2 (ACCOUNT TYPE 2)	RATE CODE 3 (ACCOUNT TYPE 3)
.0500	.0525	.0550
.0525	.0550	.0575
.0550	.0575	.0600
.0550	.0575	.0625

10.4 Variable repeating groups

10.4.1 What you can do

You can retrieve occurrences of input fields that repeat a variable number of times.

10.4.2 How to do it

1. Define the **group** by coding one REC parameter with:
 - a. An identifying name for the group of fields
 - b. The start position
 - c. The keyword **GROUP**
 - d. A group identifier
 - e. The total length of one occurrence of the group (in bytes), followed by a period (.)
 - f. The name of the field in the fixed portion of the record that contains the number of occurrences of the group
2. Define the **elements** of the group by coding a REC parameter for each element of the group with:
 - a. The name of the element
 - b. The start position *within the group*
 - c. The length of the element
 - d. The keyword **ELMNT**
 - e. The identifier of the group
3. Reference the elements by using explicit subscripts.

10.4.3 Demonstration

Objective: This report lists insurance policy numbers and coverage dates for several store locations. The number of policies varies for each store.

Procedure

- Three fields (LOCATION-ID, LOC-EFF-DATE, and LOC-EXP-DATE) in a variable repeating data group are defined on REC parameters:
 - **LOCATION-INFO** is the assigned group name.
 - **LOCATION-NUMBER** is the input record field containing the number of occurrences.

- **INDEX**, which is used as an explicit subscript, is a work field that contains the current occurrence count.

Complete code

```
col. 2
↓
IN 80 F 320 PS(TAPE)
REC POLICY-NUMBER      5 10  2
REC COVERAGE-CODE      15   3
REC EFFECTIVE-DATE     18   6  2
REC LOCATION-NUMBER    24   4  2
REC LOCATION-INFO 28          GROUP AA 14.LOCATION-NUMBER
REC LOCATION-ID       1 2 2      ELMNT AA
REC LOC-EFF-DATE     3 6 2      ELMNT AA
REC LOC-EXP-DATE     9 6 2      ELMNT AA
01OUT D
010 INDEX 1           $Counter
0141*010 'POLICY NUMBER'
0141*020 'COVERAGE'
0141*030 'EFFECTIVE DATE'
0141*040 'LOCATION ID'
0141*050 'LOC-EFF-DATE'
0141*060 'LOC-EXP-DATE'
01420001  ''
0151*010 POLICY-NUMBER F0
0151*020 COVERAGE-CODE
0151*030 EFFECTIVE-DATE FD
0152*040 LOCATION-ID. INDEX
0152*050 LOC-EFF-DATE. INDEX FD
0152*060 LOC-EXP-DATE. INDEX FD
017    RELS 1           $Print first detail line; continue processing
017100  RELS 2           $Print second detail line; continue processing
017    INDEX + 1 INDEX
017    IF INDEX LE LOCATION-NUMBER 100$Test subscript value
017    MOVE 1 TO INDEX      $Reset subscript value
017    DROP
```

Result:

POLICY NUMBER	COVERAGE	EFFECTIVE DATE	LOCATION ID	LOC-EFF-DATE	LOC-EXP-DATE
111111	010	12/01/87	1 5	12/01/87 12/01/87	12/01/92 12/15/93
222222	011	01/30/85	10	09/15/88	09/15/98
333333	005	05/30/84	5 3 1	08/01/83 02/01/85 01/01/86	08/01/92 02/01/92 01/01/96
4444444	002	11/15/86	4 2	07/01/82 02/01/88	07/01/92 02/01/98

10.5 Floating fields

10.5.1 What you can do

You can retrieve a field that occurs immediately after a variable repeating group (floating field) by creating code that is relative to the end of the variable group.

10.5.2 How to do it

1. Define the floating group by coding one **REC** parameter with:
 - a. The assigned group name
 - b. A start position of 1
 - c. The keyword **GROUP**
 - d. A unique group identifier
 - e. The group identifier of the last variable repeating group
 - f. The length of the floating field in bytes
2. Define the field by coding a **REC** parameter with:
 - a. The name of the floating field
 - b. A start position of 1
 - c. The length of the field
 - d. The keyword **ELMNT**
 - e. The identifier of the group
3. Reference the field directly.

10.5.3 Demonstration

Objective: This report lists insurance coverage and risk level for a series of stores from an input file containing the floating field RISK-LEVEL.

Procedure

- **OTHER-DATA** defines the floating group.
- **RISK-LEVEL** defines the floating field.
- RISK-LEVEL is referenced directly on a type 5 parameter.

Complete code

```

col. 2
↓
IN 80
REC POLICY-NUMBER      5 10 2
REC COVERAGE-CODE       15 3
REC EFFECTIVE-DATE      18 6 2
REC LOCATION-NUMBER     24 4 2
REC LOCATION-INFO        28
                           GROUP AA 14.LOCATION-NUMBER
REC LOCATION-ID          1 2 2
                           ELMNT AA
REC LOC-EFF-DATE         3 6 2
                           ELMNT AA
REC LOC-EXP-DATE         9 6 2
                           ELMNT AA
REC OTHER-DATA           1
                           GROUP BB AA 1
REC RISK-LEVEL           1 1
                           ELMNT BB
010UT D
010 INDEX 1
0141*010 'POLICY NUMBER'
0141*020 'COVERAGE'
0141*030 'EFFECTIVE DATE'
0141*035 'RISK LEVEL'
0141*040 'LOCATION ID'
0141*050 'LOC-EFF-DATE'
0141*060 'LOC-EXP-DATE'
01420001 ' '
0151*010 POLICY-NUMBER F0
0151*020 COVERAGE-CODE
0151*030 EFFECTIVE-DATE FD
0151*035 RISK-LEVEL           $The floating field
0152*040 LOCATION-ID.INDEX
0152*050 LOC-EFF-DATE.INDEX FD
0152*060 LOC-EXP-DATE.INDEX FD
017 RELS 1
017100 RELS 2
017 INDEX + 1 INDEX
017 IF INDEX LE LOCATION-NUMBER 100
017 MOVE 1 TO INDEX
017 DROP

```

Result:

POLICY NUMBER	COVERAGE	EFFECTIVE DATE	RISK LEVEL	LOCATION ID	LOC-EFF-DATE	LOC-EXP-DATE
111111	010	12/01/87		1	12/01/87	12/01/92
				5	12/01/87	12/15/93
222222	011	01/30/85		10	09/15/88	09/15/98
333333	005	05/30/84		5	08/01/83	08/01/92
				3	02/01/85	02/01/92
				1	01/01/86	01/01/96
444444	002	11/15/86		4	07/01/82	07/01/92
				2	02/01/88	02/01/98

10.6 Obtaining accumulated totals

10.6.1 What you can do

You can obtain the accumulated totals value for each occurrence of a subscripted field.

10.6.2 How to do it

1. Name the occurrences of the subscripted field on **type 5** parameters.
2. Print the specific totals values with **type 6** parameters.

In the example below, AMT.5 is defined as a work field consisting of five repeating fields. The type 5 parameters extract the value of each occurrence of AMT.1 and AMT.2. The type 6 parameters return the sum of the occurrences of AMT.1 and the sum of the occurrences of AMT.2:

```
010 AMT.5  
0151*005 AMT.1  
0151*010 AMT.2  
0161*005 AMT.1  
0161*010 AMT.2
```

10.6.3 Demonstration

Objective: This report lists the quarterly gain or loss for a series of stores by retrieving the accumulated totals value for each occurrence of the subscripted work field GAIN-OR-LOSS.4.

Procedure

- Two repeating input fields are defined:
 - INCOME and EXPEND are the group names.
 - QTR and AMT are the field names.
- Gain or loss statistics for each fiscal quarter are computed from the elements (QTR - AMT), which are placed in the work field **GAIN-OR-LOSS.4**.
- Individual occurrences of the GAIN-OR-LOSS work field are retrieved by type 5 parameters.
- References to each occurrence of the GAIN-OR-LOSS work field on type 6 lines obtain the accumulated totals for each quarter.

Complete code

```

col. 2
↓
IN 80 F 320 PS(TAPE)
REC STORE      1 3 2
REC INCOME     4
REC QTR        1 8 2 DP=2      GROUP AA 8.4
REC EXPEND     36           ELMNT AA
REC AMT        1 8 2 DP=2      GROUP BB 8.4
                           ELMNT BB
010 GAIN-OR-LOSS.4 DP=2
010 INDEX 1
013 PROFIT/LOSS REPORT — 1990
0151*010 STORE          HH 'STORE'
0151*020 GAIN-OR-LOSS.1 SZ=11 F3 HH 'FIRST QUARTER'
0151*030 GAIN-OR-LOSS.2 SZ=11 F3 HH 'SECOND QUARTER'
0151*040 GAIN-OR-LOSS.3 SZ=11 F3 HH 'THIRD QUARTER'
0151*050 GAIN-OR-LOSS.4 SZ=11 F3 HH 'FOURTH QUARTER'
0161*0200GAIN-OR-LOSS.1 SZ=11 F3           $Quarterly totals
0161*030 GAIN-OR-LOSS.2 SZ=11 F3
0161*040 GAIN-OR-LOSS.3 SZ=11 F3
0161*050 GAIN-OR-LOSS.4 SZ=11 F3
017100 COMPUTE QTR.INDEX - AMT.INDEX GAIN-OR-LOSS.INDEX
017    INDEX + 1 INDEX
017    IF INDEX LE 4 100
017    MOVE 1 TO INDEX
017    TAKE

```

Result:

REPORT NO. 01 STORE	PROFIT/LOSS REPORT — 1995 FIRST QUARTER	09/27/99 PAGE SECOND QUARTER	1 THIRD QUARTER	FOURTH QUARTER
1	19,900.21	10,788.50	53,899.98	19,895.00
10	0.21	1,388.41	99.98	2,895.76
11	5,099.79-	34,788.50	17,900.00	19,895.00
21	44,900.21	10,211.50-	123,899.98	70,104.50-
	59,700.84	36,753.91	195,799.94	27,418.74-

10.7 Obtaining specific field values

10.7.1 What you can do

You can obtain the current value of a subscripted numeric work field in output processing when the work field is specified on type 6 or 8 parameters only.

10.7.2 How to do it

Code:

- The **work field**
- A **type 6** or a **type 8** parameter with the name of the work field and the subscript indicating the occurrence to be retrieved

In the following example, AMT.1 and AMT.2 on the type 6 parameter return the current values of the first and second work field occurrences:

```
010 AMT.5  
0161*005 AMT.1  
0161*010 AMT.2
```

10.7.3 Demonstration

Objective: The following report lists the quarterly gain or loss for each of four stores. The annual gain or loss, by store, is printed across the bottom of the report.

Procedure

- **GAIN-OR-LOSS.4** is a subscripted work field that holds the quarterly income computation for each store.
- **STORE-TOTAL** is a subscripted work field that holds the annual income of each store.
- Each occurrence of STORE-TOTAL is printed out by references on **type 6** parameters.

Complete code

```
col. 2
↓
IN 80 F 320 PS(TAPE)
REC STORE      1 3 2
REC INCOME     4          GROUP AA 8.4
REC QTR        1 8 2 DP=2   ELMNT AA
REC EXPEND     36         GROUP BB 8.4
REC AMT        1 8 2 DP=2   ELMNT BB
010 GAIN-OR-LOSS.4 DP=2           $Quarterly income
010 INDEX 1
010 TOTAL 0
010 STORE-TOTAL.4                  $Annual income
010 COUNT 1
013 PROFIT/LOSS REPORT — 1990
0151*010 STORE SZ=5      HH 'STORE'
0151*020 GAIN-OR-LOSS.INDEX SZ=11 F3 HH 'QUARTERLY' 'GAIN/LOSS'
0152*001 ''
0161*0200GAIN-OR-LOSS.INDEX SZ=11 F3
0162*010-'STORE TOTAL SUMMARY:'
0163*010 'STORE 1:'
0163*015 STORE-TOTAL.1 SZ=11 F3 $Current value of the first occurrence
0163*020 'STORE 10:'
0163*025 STORE-TOTAL.2 SZ=11 F3 $Current value of the second occurrence
0163*030 'STORE 11:'
0163*035 STORE-TOTAL.3 SZ=11 F3 $Current value of the third occurrence
0163*050 'STORE 20:'
0163*055 STORE-TOTAL.4 SZ=11 F3 $Current value of the fourth occurrence
017100 COMPUTE QTR.INDEX - AMT.INDEX GAIN-OR-LOSS.INDEX
017 COMPUTE TOTAL + GAIN-OR-LOSS.INDEX TOTAL
017 RELS (1)
017175 IF INDEX EQ 4 200
017 INDEX + 1 INDEX
017 IF INDEX LE 4 100
017200 MOVE 1 TO INDEX
017 MOVE TOTAL TO STORE-TOTAL.COUNT
017 COUNT + 1 COUNT
017 RELS (2)
017 TOTAL - TOTAL TOTAL
017 DROP
```

Result:

REPORT NO. 01	PROFIT/LOSS REPORT — 1995	09/27/99 PAGE	1
QUARTERLY GAIN/LOSS			
STORE			
1	19,900.21		
1	10,788.50		
1	53,899.98		
1	19,895.00		
10	0.21		
10	1,388.41		
10	99.98		
10	2,895.76		
11	5,099.79-		
11	34,788.50		
11	17,900.00		
11	19,895.00		
21	44,900.21		
21	10,211.50-		
21	123,899.98		
21	70,104.50-		
264,835.95			
- STORE TOTAL SUMMARY:			
STORE 1:	104,484	STORE 10:	
		4,384	STORE 11:
			67,484
			STORE 20:
			88,485

10.8 Obtaining sort-key values

10.8.1 What you can do

You can use one element of a subscripted field as a sort key. The value of the sort key associated with the last detail line processed before the control break is returned when type 6 or type 8 parameters reference the field.

10.8.2 How to do it

Code:

1. The subscripted field on a **SORT** parameter
2. The subscripted field on a **type 6** or **type 8** parameter

The following example uses a subscripted work field, AMT.5, on the SORT parameter. The type 6 reference to the field returns the current value of AMT.5:

```
010 AMT.5  
01SORT AMT.5  
0151*010 ''  
0161*010 AMT.5
```

10.8.3 Demonstration

Objective: This report prints the third quarter profits for each of six stores.

Procedure

- The work field **GAIN-OR-LOSS.4** holds the results of quarterly income calculations.
- An occurrence of the subscripted work field **GAIN-OR-LOSS.3** is used as the sort key. The value of that particular field is written to the extract file when a control break occurs.
- A **type 6** parameter references GAIN-OR-LOSS.3 to retrieve the subscripted field value from the extract file.

Complete code

```

col. 2
↓
IN 80 F 320 PS(TAPE)
REC STORE      1 3 2
REC INCOME     4          GROUP AA 8.4
REC QTR        1 8 2 DP=2   ELMNT AA
REC EXPEND     36         GROUP BB 8.4
REC AMT        1 8 2 DP=2   ELMNT BB
01OUT T
010 GAIN-OR-LOSS.4 DP=2      $Four quarters
010 INDEX 1
01SORT STORE 0 GAIN-OR-LOSS.3      $Third occurrence in work field
013 THIRD QUARTER EARNINGS — 1990
0151*010 ''           $One type 5 line is always required
0161*010 STORE        HH 'STORE'
* $                   Retrieve the value:
0161*020 GAIN-OR-LOSS.3 SZ=11 F3 HH 'THIRD QUARTER PROFIT'
017100 COMPUTE QTR.INDEX - AMT.INDEX GAIN-OR-LOSS.INDEX
017175 IF INDEX EQ 4 200      $Load subscripted work field
017    INDEX + 1 INDEX
017    IF INDEX LE 4 100
017200 MOVE 1 TO INDEX
018010 IF LEVL EQ 1 100
018020 IF LEVL EQ 2 200
018100 TAKE 1
018200 DROP

```

Result:

REPORT NO. 01	THIRD QUARTER EARNINGS — 1995 STORE	09/27/99 PAGE	1 THIRD QUARTER PROFIT
	1		53,899.98
	10		99.98
	11		17,900.00
	21		123,899.98

Chapter 11. File Matching

11.1	What file matching is	11-3
11.1.1	How file matching works	11-3
11.1.2	How to code match-file runs	11-7
11.2	Matching single-occurrence files	11-8
11.2.1	Demonstration	11-8
11.3	Matching multiple transactions with a single-entry master	11-10
11.3.1	Demonstration (1): dropping unmatched records	11-10
11.3.2	Demonstration (2): selecting matching account numbers	11-13
11.4	Listing accounts without transactions	11-15
11.4.1	Demonstration	11-15
11.5	Listing transactions not on the master file	11-18
11.5.1	Demonstration	11-18
11.6	Using the match-file facility for table initialization	11-21
11.6.1	What you can do	11-21
11.6.2	How to do it	11-21
11.6.3	Demonstration	11-21
11.7	Using files defined to the data dictionary	11-25
11.7.1	What you can do	11-25
11.7.2	How to do it	11-25
11.7.3	Demonstration	11-25
11.8	Creating unique match-key names	11-26
11.8.1	What you can do	11-26
11.8.2	How to do it	11-26
11.8.3	Demonstration	11-26
11.9	Qualified fields	11-27
11.9.1	What you can do	11-27
11.9.2	How to do it	11-27
11.9.3	Demonstration	11-27

11.1 What file matching is

Two or more *presorted* conventional files having at least one common data item between them can be compared. When a match occurs, the matching records are brought into the input buffer as a single record.

11.1.1 How file matching works

When CA-Culprit encounters more than one INPUT parameter, it recognizes a match run and proceeds as follows:

1. Searches the first record on all files for a match:

If a SELECT statement is used on any file, the SELECT logic is performed *before* any match-file processing.

If a SELECT BUFFER statement is used, the SELECT logic is performed *after* the match processing and before type 7 logic is completed.

2. Searches the remaining file for a match:

If a match among all files is found, the matching records are delivered to the input buffer and type 7 processing is performed.

If a match is not found,

- a. The record with the lowest key value is moved into the input buffer.
- b. The remainder of the input buffer is nulled out by setting numeric key fields to zeros, alphanumeric key fields to spaces, and the rest of the record to spaces.
- c. Type 7 processing is performed.

The figure below shows the contents of the input buffer during a simple match-file run:

Buffer contents during a match-file run

PHYSICAL RECORD SEQUENCE		INPUT BUFFER		
FILE 1 (SYS010)	FILE 2 (SYS011)	SYS010 AREA	SYS011 AREA	INPUT BUFFER NUMBER
A	A	A	A	1
C	B	*	B	2
D1	D	C	*	3
D2	E1	D1	D	4
E1	E2	D2	*	5
E2	E3	E1	E	6
G	F	E2	E	7
H	G1	*	E	8
	G2	*	F	9
		G	G	10
		*	G	11
		H	*	12

Note: D₁, D₂, E₁, and so on, represent records with duplicate key values. An asterisk (*) indicates a null chapter of the buffer.

Keeping a Record in the Buffer: When a record needs to be processed against more than one record with a matching key value, the **MB=KEEP** option of the INPUT parameter is used. MB=KEEP is most commonly used on one file only.

The contents of the input buffer when the MB=KEEP option is used is shown in the figure below:

Buffer contents with the KEEP option

PHYSICAL RECORD SEQUENCE		INPUT BUFFER		
FILE 1 (SYS010)	FILE 2 (SYS011)	SYS010 AREA	SYS011 AREA	INPUT BUFFER NUMBER
A	A	A	A	1
C	B	*	B	2
D1	D	C	B	3
D2	E	D1	D	4
E1	F	D2	D	5
E2	G	E1	E	6
G		E2	E	7
H		*	F	8
		G	G	9
		H	G	10

Note: D₁, D₂, E₁, and E₂ represent records with duplicate key values. An asterisk (*) indicates a null chapter of the buffer.

Checking the File Status: When CA-Culprit reformats the input buffer for a match file run, it adjusts the starting position of the input fields by inserting two additional bytes for use in checking the status of the files:

- The first byte is an input file status byte specific to each file in the run.
- The second byte, called M*ID, is a composite input file status indicator consisting of a binary combination of the file-specific status bytes for all files in the run.

If the return value of M*ID is not 0, test the file-specific status bit to locate the file or files containing an error condition:

1. Define the file-specific status bit on a REC parameter:

If binary: REC FILE4-STATUS 0 1 1

If bit: REC FILE4-STATUS 0 81 5
 (Bit format uses a 2-digit length specification,
 indicating FILE4-STATUS is a single bit that occurs
 in the eighth bit position of start position 0.)

2. Test the value of the bit in type 7 logic for a non-zero (0) value.

The bit layout within the M*ID and the file-specific status bit fields are shown in the table below.

The Bit Layout within M*ID

Bit position	Condition	Decimal value of M*ID
5	File(s) out of sequence	8
6	Duplicate key value	4
8	End of file(s) value	1
0	Off	0

Demonstration: The following examples list the values of the file-specific status byte and the M*ID under error conditions:

- The first example attempts a match-file run with an unsorted transaction file.
- The second example attempts a match-file run with an unsorted transaction file and an unsorted master file.

Complete code

```
col. 2
↓
IN 80 MK=M-ACCOUNT
REC M-FILE-STATUS 0    1  1      'M-FILE' 'STATUS'
REC M-ACCOUNT          1   5      'ACCOUNT'
REC M-NAME             620
IN 80 MK=T-ACCOUNT
REC T-FILE-STATUS 0    1  1      'T-FILE' 'STATUS'
REC T-ACCOUNT          1   5      'ACCOUNT'
REC T-PAYMENT          6   8      2  DP=2 'PAYMENT'
020OUT D
0251*005 M*ID          HH 'M*ID'
0251*010 T-FILE-STATUS
0251*015 M-FILE-STATUS
0251*020 M-ACCOUNT      HR
```

Unsorted transaction file

REPORT NO. 02			JANUARY PAYMENTS RECEIVED	5/23/99 PAGE 1
M*ID	T-FILE STATUS	M-FILE STATUS	ACCOUNT	
1	1		15060	
1	1		21056	
1	1		29557	
1	1		30115	
1	1		33470	
1	1		69876	
1	1		99083	
8	8			
8	8			
.	.			
.	.			
.	.			

Unsorted master and transaction files

REPORT NO. 02 JANUARY PAYMENTS RECEIVED 5/23/99 PAGE 1			
M*ID	T-FILE STATUS	M-FILE STATUS	ACCOUNT
8	8	8	
8	8	8	
8	8	8	
4	4	4	
4	4	4	
.	.	.	
.	.	.	
.	.	.	

11.1.2 How to code match-file runs

Step 1: Sort all files in ascending match-key sequence.

►► See Chapter 13, “Additional Standard File Facilities” on page 13-1 for creating sorted files.

Step 2: Code:

1. One **INPUT** parameter with the **MK=** option for each file to be matched.

Match keys must be either all numeric or all alphanumeric. Numeric types can be mixed:

- Specify the **DD=** option on all or none of the **IN** parameters.
- Specify **MB=KEEP** if a record is to be matched against multiple occurrences of another file.
- Correlate the coding sequence of the CA-Culprit file definitions with the sequence of JCL statements that define the files (SYS010, SYS011, and additional increments, as needed).

2. **REC** parameters immediately following the associated **INPUT** parameter:

- Use unique names for all fields.
- Use start positions that are relative to the beginning of the record (not to the beginning of the input buffer).

3. Optional **SELECT/BYPASS** parameters:

- After the **REC** parameters of the specific file to be processed.
- With the **BUFFER** option immediately after the last input file definition if the processing applies to all files after matching takes place.

11.2 Matching single-occurrence files

11.2.1 Demonstration

Objective: This report contains account information from both a master file and a matching transaction file. Each file has a match key occurrence.

Procedure: The contents of the input buffer, represented by account number, show how the files are matched:

(SYS010)	(SYS011)
15060	15060
21056	21056
29557	29557
30115	30115
33470	33470
69876	69876
99083	99083

Complete code

```
col. 2
↓
IN 80 F 320 PS(TAPE) MK=M-ACCOUNT      $Master file
REC FILE-STATUS 0 1 1                   $Input file status
REC M-ACCOUNT      1 5                 'ACCOUNT'
REC M-NAME        6 20                'NAME'
REC M-ADDRESS      26 37               '          ADDRESS'
REC M-ZIP         63 5                 '          ZIP'

IN 80 F 320 PS(TAPE) MK=T-ACCOUNT      $Transaction file
REC T-ACCOUNT      1 5                 'ACCOUNT'
REC T-PAYMENT      6 8 2 DP=2          'PAYMENT'
REC T-DATE        14 6 2               'DATE'
020UT D
023 JANUARY PAYMENTS RECEIVED
0251*010 M-ACCOUNT      HR
0251*020 M-NAME        HR
0251*030 M-ADDRESS      HR
0251*035 M-ZIP         HR
0251*040 T-PAYMENT     SZ=10      HR
0251*050 T-DATE       FD         HR
027010 IF FILE-STATUS EQ 1  DROP      $EOF test
```

Result:

REPORT NO. 02 ACCOUNT	JANUARY PAYMENTS RECEIVED NAME	09/29/99 PAGE ADDRESS	1	PAYMENT	DATE
15060	SHARON ARMSTRONG	25 CEDAR STREET WAYLAND MA	02756	100.99	01/15/91
21056	AMOS JOHNSON	22651 MASS AVENUE SAN FRANCISCO CA	09801	1,000.00	01/20/91
29557	IRWIN TRIMBLE	102 COLBOURNE ST BROOKLINE MA	09743	50.00	01/04/91
30115	IRMA DOONES	5656 SANDHILL ROAD OCEAN CITY CA	09743	1,009.90	01/04/91
33470	VICTORIA DAY	1234 OAK HILL ROAD EVERGREEN MI	07895	759.00	01/13/91
69876	BRUCE THORPE	11002 PEACHTREE LA ATLANTA GA	76543	900.99	01/01/91
99083	HELEN SANTOVEC	3 APPLE ORCHARD CHARLOTTE NC	64321	5.00	01/20/91

11.3 Matching multiple transactions with a single-entry master

11.3.1 Demonstration (1): dropping unmatched records

Objective: This report lists customer transactions and current balances by matching a multiple-occurrence transaction file against a single-entry master file. Unmatched records are dropped.

Procedure

- **MB=KEEP** retains a master record in the input buffer.
- The **BYPASS** parameter eliminates unwanted master file records.
- **Type 7** logic drops all unmatched records.

The contents of the input buffer, represented by account number, show how the files are matched:

(SYS010)	(SYS011)
15060	15060
	15060
	15060
15999	
16070	16070
19235	19235
21056	21056
23055	23055
	23055
	23055
27777	
29557	29557
30115	30115
	30115
31113	31113
	31113
31275	31275
32115	
33470	33470
	33470
34440	34440
	34440
	34440
36682	36682
69876	69876
	69876
99083	99083
	99083

Complete code

```

col. 2
↓
IN 80 MK=M-ACCOUNT MB=KEEP $Keep the master record
REC M-ACCOUNT    1   5      'ACCOUNT'
REC M-BRANCH     6   2      'BRANCH'
REC M-NAME       19  20     'NAME'
BYP M-BRANCH NE ('32' '35' '45' '46') $Select records

IN 80 MK=T-ACCOUNT
REC T-ACCOUNT    1   5      'ACCOUNT'
REC T-TRANS-IND  6   1      'DEPOSIT/' 'WITHDRAWAL'
REC T-TRANS-AMT  7  11  2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC T-DATE       18  6   2      'DATE'
REC T-BRANCH     24  2      'BRANCH'
REC T-NAME       26  20     'NAME'
800UT D
80SORT T-BRANCH - T-ACCOUNT T-DATE
803 MASTER FILE MATCHED WITH MULTIPLE TRANSACTIONS
8051*010 M-BRANCH          HR
8051*020 T-ACCOUNT         HR
8051*030 M-NAME            HR
8051*050 T-TRANS-AMT       HR
8051*060 T-TRANS-IND       HR
8051*070 T-DATE            FD
807010 IF M-ACCOUNT NE T-ACCOUNT DROP $Drop unmatched records

```

Result:

REPORT NO. 80 MASTER FILE MATCHED WITH MULTIPLE TRANSACTIONS			09/28/99	PAGE 1	AMOUNT OF TRANSACTION	DEPOSIT/ WITHDRAWAL	DATE
BRANCH	ACCOUNT	NAME					
32	15060	SHARON ARMSTRONG			10,099.01	D	01/15/91
32	15060	SHARON ARMSTRONG			990.11	W	01/16/91
32	15060	SHARON ARMSTRONG			100,990.11	D	01/20/91
32	16070	ARTHUR LINK			1,080.04	D	01/01/91
32	19235	GARY NOBLES			80.04	W	01/02/91
32	21056	AMOS JOHNSON			1.02	W	01/20/91
35	23055	JACK JACKSON			500.00	W	01/02/91
35	23055	JACK JACKSON			50,000.00	D	01/25/91
35	23055	JACK JACKSON			30.00	D	01/26/91
35	29557	IRWIN TRIMBLE			500,001.00	D	01/04/91
45	33470	VICTORIA DAY			7,590,001.30	D	01/13/91
45	33470	VICTORIA DAY			1.30	D	01/26/91
45	34440	HELEN WRIGHT			195.01	W	01/12/91
45	34440	HELEN WRIGHT			2,000.01	D	01/24/91
45	34440	HELEN WRIGHT			95.01	D	01/27/91
45	34440	HELEN WRIGHT			100.00	D	01/28/91
45	36682	JEAN WREN			4,000.78	W	01/12/91
46	69876	BRUCE THORPE			9,009,901.15	D	01/01/91
46	69876	BRUCE THORPE			637.55	D	01/02/91
46	99083	HELEN SANTOVEC			50,001.30	D	01/20/91
46	99083	HELEN SANTOVEC			6,468.52	D	01/22/91

11.3.2 Demonstration (2): selecting matching account numbers

This is the same report presented in Demonstration (1). Only this time records with matching account numbers are selected.

Procedure

- **MB=KEEP** retains a master record in the input buffer.
- The **SELECT** parameter selects master file records.
- The **SELECT** parameter with the **BUFFER** option selects only records that have matching account numbers.

Complete code

```

col. 2
↓
IN 80 F 320 PS(TAPE) MK=M-ACCOUNT MB=KEEP      $Keep the master record
REC M-ACCOUNT    1   5          'ACCOUNT'
REC M-BRANCH     6   2          'BRANCH'
REC M-NAME       19  20         'NAME'
SEL M-BRANCH EQ ('32' '35' '45' '46')           $Select records

IN 80 F 320 PS(TAPE) MK=T-ACCOUNT
REC T-ACCOUNT    1   5          'ACCOUNT'
REC T-TRANS-IND  6   1          'DEPOSIT/' 'WITHDRAWAL'
REC T-TRANS-AMT  7  11 2 DP=2  'AMOUNT OF' 'TRANSACTION'
REC T-DATE       18  6 2        'DATE'
REC T-BRANCH     24   2          'BRANCH'
REC T-NAME       26  20         'NAME'
SEL BUFFER WHEN M-ACCOUNT EQ T-ACCOUNT $Select matching records
800UT D
80SORT T-BRANCH - T-ACCOUNT T-DATE
803 MASTER FILE MATCHED WITH MULTIPLE TRANSACTIONS
8051*010 M-BRANCH          HR
8051*020 T-ACCOUNT         HR
8051*030 M-NAME            HR
8051*050 T-TRANS-AMT       HR
8051*060 T-TRANS-IND       HR
8051*070 T-DATE            FD

```

Result:

11.3 Matching multiple transactions with a single-entry master

REPORT NO. 80 MASTER FILE MATCHED WITH MULTIPLE TRANSACTIONS 09/28/99 PAGE 1				DEPOSIT/ WITHDRAWAL	DATE
BRANCH	ACCOUNT	NAME			
32	15060	SHARON ARMSTRONG	10,099.01	D	01/15/91
32	15060	SHARON ARMSTRONG	990.11	W	01/16/91
32	15060	SHARON ARMSTRONG	100,990.11	D	01/20/91
32	16070	ARTHUR LINK	1,080.04	D	01/01/91
32	19235	GARY NOBLES	80.04	W	01/02/91
32	21056	AMOS JOHNSON	1.02	W	01/20/91
35	23055	JACK JACKSON	500.00	W	01/02/91
35	23055	JACK JACKSON	50,000.00	D	01/25/91
35	23055	JACK JACKSON	30.00	D	01/26/91
35	29557	IRWIN TRIMBLE	500,001.00	D	01/04/91
45	33470	VICTORIA DAY	7,590,001.30	D	01/13/91
45	33470	VICTORIA DAY	1.30	D	01/26/91
45	34440	HELEN WRIGHT	195.01	W	01/12/91
45	34440	HELEN WRIGHT	2,000.01	D	01/24/91
45	34440	HELEN WRIGHT	95.01	D	01/27/91
45	34440	HELEN WRIGHT	100.00	D	01/28/91
45	36682	JEAN WREN	4,000.78	W	01/12/91
46	69876	BRUCE THORPE	9,009,901.15	D	01/01/91
46	69876	BRUCE THORPE	637.55	D	01/02/91
46	99083	HELEN SANTOVEC	50,001.30	D	01/20/91
46	99083	HELEN SANTOVEC	6,468.52	D	01/22/91

11.4 Listing accounts without transactions

11.4.1 Demonstration

Objective: This report lists accounts that do not have transactions. The match-file run matches a multiple-transaction file against a single-occurrence master file.

Procedure

- **MB=KEEP** retains a master file in the input buffer.
- A **SELECT BUFFER** statement selects those accounts without transactions.

Note: Type 7 logic (IF M-ACCOUNT LE T-ACCOUNT DROP) can be used instead of the SELECT BUFFER statement.

The contents of the input buffer:

(SYS010)	(SYS011)
15060	15060
15060	15060
15060	15060
15999	
16070	16070
19235	19235
21056	21056
23055	23055
23055	23055
23055	23055
27777	
29557	29557
30115	30115
30115	30115
31113	31113
31113	31113
31275	31275
32115	
33470	33470
33470	33470
34440	34440
34440	34440
34440	34440
36682	36682
69876	69876
69876	69876
99083	99083
99083	99083

Complete code

```

col. 2
↓
IN 80 MK=M-ACCOUNT MB=KEEP $Keep master record
REC M-ACCOUNT      1    5      'ACCOUNT'
REC M-BRANCH       6    2      'BRANCH'
REC M-NAME         19   20     'NAME'

IN 80 MK=T-ACCOUNT
REC T-ACCOUNT      1    5      'ACCOUNT'
SEL BUFFER WHEN T-ACCOUNT EQ ' '$Accounts wihout transactions
800UT D
80SORT M-BRANCH -
803 LIST OF ACCOUNTS WITHOUT TRANSACTIONS
8051*010 M-BRANCH      HR
8051*020 M-ACCOUNT     HR
8051*030 M-NAME        HR

```

Result:

REPORT NO. 80	LIST OF ACCOUNTS WITHOUT TRANSACTIONS	09/28/99	PAGE	1	NAME
BRANCH		ACCOUNT			
32		15999			SHELLY BROWN
35		27777			ANDY PIGGOTT
40		32115			BOB DATO

11.5 Listing transactions not on the master file

11.5.1 Demonstration

Objective: This report lists transactions not on the master file. Multiple transactions are matched against a single-entry master file.

Procedure

- **MB=KEEP** retains a master file in the input buffer.
- A **SELECT BUFFER** statement selects transactions that cannot be matched with the master file.

Note: Type 7 logic (IF M-ACCOUNT GE T-ACCOUNT DROP) can be used instead of the SELECT BUFFER statement.

The contents of the input buffer for the run (K denotes master records kept in the input buffer):

(SYS010)	(SYS011)
	15060
	15060
	15060
15999	
16070	16070
19235	19235
21056	21056
23055	23055
23055K	23055
23055K	23055
27777	
29557	29557
30115K	30115
30115K	30115
30115K	31113
30115K	31113
31275	31275
32115	
33470	33470
33470K	33470
34440	34440
34440K	36682
69876	69876
69876K	69876
99083	99083
99083K	99083

Complete code

```

col. 2
↓
IN 80 MK=M-ACCOUNT MB=KEEP
REC M-ACCOUNT    1      5          'ACCOUNT'

IN 80 MK=T-ACCOUNT
REC T-ACCOUNT    1      5          'ACCOUNT'
REC T-TRANS-IND  6      1          'DEPOSIT/' 'WITHDRAWAL'
REC T-TRANS-AMT  7      11 2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC T-DATE       18     6 2          'DATE'
REC T-BRANCH     24     2          'BRANCH'
REC T-NAME       26     20         'NAME'
SEL BUFFER WHEN T-ACCOUNT GT M-ACCOUNT $Transactions without masters
800UT D
80SORT T-BRANCH - T-DATE D T-ACCOUNT
803 LISTING OF TRANSACTIONS WITHOUT MASTERS
8051*010 T-BRANCH           HR
8051*020 T-ACCOUNT          HR
8051*030 T-NAME             HR
8051*050 T-TRANS-AMT        HR
8051*060 T-TRANS-IND        HR
8051*070 T-DATE             FD   HR

```

Result:

REPORT NO. 80 LISTING OF TRANSACTIONS WITHOUT MASTERS			09/28/99 PAGE	1	AMOUNT OF TRANSACTION	DEPOSIT/ WITHDRAWAL	DATE
BRANCH	ACCOUNT	NAME					
32	15060	SHARON ARMSTRONG		100,990.11		D	01/20/91
32	15060	SHARON ARMSTRONG		990.11		W	01/16/91
32	15060	SHARON ARMSTRONG		10,099.01		D	01/15/91
40	31113	MARY CATREY		36.00			01/24/91
40	31113	MARY CATREY		36.00		D	01/21/91
45	36682	JEAN WREN		4,000.78		W	01/12/91

11.6 Using the match-file facility for table initialization

11.6.1 What you can do

You can use match keys to control the order in which files are read in a match run. If you do not specify match keys for a file, CA-Culprit assumes low key values for the missing match keys on that file and reads it first.

11.6.2 How to do it

1. Omit the MK= specification on the IN parameter for the file you want read first.
2. Code the MK= specification on the IN parameter for subsequent files.

11.6.3 Demonstration

Objective: This example produces two reports:

- An interest rate table from the first file processed
- A listing of the first month's interest for new accounts, using the interest rate table and the input data from the second file

Procedure

- No match key is used for the interest rate file.
- A match key on ACCOUNT-NO is used for the second file.

The contents of the input buffer for this run:

File 1 (SYS010)	File 2 (SYS011)
.0500 .0525 .0550	
.0525 .0550 .0575	
.0550 .0575 .0600	
.0550 .0575 .0625	
	15060
	15999
	16070
	19235
	21056
	23055
	27777
	29557
	.
	.
	.

Complete code

```

col. 2
↓
IN 80                                     $Read first, no match key
REC FILE-TEST          0 1 1
REC INTEREST-RATE-DATA 1                   GROUP AA 4.3
REC INTEREST-RATE      1 4 2 DP=4 ELMNT AA

IN 150 MK=ACCOUNT-NO                     $Read second, one match key
REC ACCOUNT-NO   1 5                      'ACCOUNT' 'NUMBER'
REC ACCT-TYPE    6 2 2                  'ACCOUNT' 'TYPE'
REC OPEN-BAL     8 11 2 DP=2            'OPENING' 'BALANCE'
REC NAME         19 20                 'NAME'
REC DATE-OPENED 39 6 2                'DATE' 'OPENED'
GWO RATE.12 DP=4 COUNT    ZERO
000UT 65 D

003 INTEREST RATE TABLE                 $First report
0051*010 INTEREST-RATE.1 HH ' ' RATE CODE 1' '(ACCOUNT TYPE 1)'
0051*020 INTEREST-RATE.2 HH 'RATE CODE 2' '(ACCOUNT TYPE 2)'
0051*030 INTEREST-RATE.3 HH 'RATE CODE 3' '(ACCOUNT TYPE 3)'
92SORT ACCT-TYPE 0 OPEN-BAL D
92OUT 120 D
920 NTH
920 FIRST-MO-INT DP=3
923 FIRST MONTH'S INTEREST             $Second report
9241*010 ' '
9251*010 ACCOUNT-NO        HR
9251*020 NAME              HR
9251*030 DATE-OPENED FD    HR
9251*040 ACCT-TYPE        HR
9251*050 RATE.NTH SZ=4 HH 'INTEREST' 'RATE'
9251*060 OPEN-BAL          HR
9251*070 FIRST-MO-INT SZ=8 HH 'INTEREST' '(FIRST' 'MONTH)'
007010 IF FILE-TEST EQ 1 DROP
007 COUNT + 1 COUNT
007 MOVE INTEREST-RATE.1 TO RATE.COUNT
007 COUNT + 1 COUNT
007 MOVE INTEREST-RATE.2 TO RATE.COUNT
007 COUNT + 1 COUNT
007 MOVE INTEREST-RATE.3 TO RATE.COUNT
007 IF COUNT GT 12 040
007 TAKE
007040 ZERO DIVIDE ZERO ZERO $Forced buffer dump if more than 12 entries
007 STOP-RUN                 $Error, stop processing
927 IF FILE-TEST NE 1 DROP
927 IF DATE-OPENED GE 100186 100
927 IF DATE-OPENED GE 070186 110
927 IF DATE-OPENED GE 040186 120
927 IF DATE-OPENED GE 010186 130
927 DROP

927100 ACCT-TYPE + 9 NTH
927 B 150
927110 ACCT-TYPE + 6 NTH
927 B 150
927120 ACCT-TYPE + 3 NTH
927 B 150
927130 ACCT-TYPE + 0 NTH
927150 COMPUTE (OPEN-BAL X RATE.NTH) / 12 FIRST-MO-INT

```

Result

REPORT NO. 00	INTEREST RATE TABLE	09/29/99 PAGE	1
RATE CODE 1 (ACCOUNT TYPE 1)	RATE CODE 2 (ACCOUNT TYPE 2)	RATE CODE 3 (ACCOUNT TYPE 3)	
.0500	.0525	.0550	
.0525	.0550	.0575	
.0550	.0575	.0600	
.0550	.0575	.0625	

REPORT NO. 92	FIRST MONTH'S INTEREST	09/29/99 PAGE	1			
ACCOUNT NUMBER	NAME	DATE OPENED	ACCOUNT TYPE	INTEREST RATE	OPENING BALANCE	INTEREST (FIRST MONTH)
33470	VICTORIA DAY	10/30/91	1	.0550	51,954.10	238.123
16070	ARTHUR LINK	10/10/91	1	.0550	10,800.45	49.502
99083	HELEN SANTOVEC	10/12/91	1	.0550	4,685.28	21.474
69876	BRUCE THORPE	08/15/91	2	.0575	6,375.58	30.550
19235	GARY NOBLES	12/01/91	2	.0575	3,800.45	18.210
34440	HELEN WRIGHT	09/10/91	2	.0575	1,950.10	9.344
36682	JEAN WREN	05/27/91	3	.0575	40,007.80	191.704
15999	SHELLY BROWN	03/30/91	3	.0550	852.45	3.907
30115	IRMA DOONES	12/24/91	3	.0625	366.29	1.908
31113	MARY CATREY	07/07/91	3	.0600	360.00	1.800
15060	SHARON ARMSTRONG	10/30/91	3	.0625		
21056	AMOS JOHNSON	10/30/91	3	.0625		
23055	JACK JACKSON	10/30/91	3	.0625		
27777	ANDY PIGGOTT	10/30/91	3	.0625		
29557	IRWIN TRIMBLE	10/30/91	3	.0625		
31275	FOR ANOTHER	10/30/91	3	.0625		
32115	BOB DATO	10/30/91	3	.0625		

11.7 Using files defined to the data dictionary

11.7.1 What you can do

During match file runs, you can reference files defined to the data dictionary.

11.7.2 How to do it

Code:

1. A **DATABASE** parameter with the **DICTNAME=** option if an alternate dictionary is used.
2. An **INPUT** parameter with the **FN=** option to specify the file name.
3. **INPUT parameter options** and **REC** parameters if redefinition or supplementary definitions are needed. See 13.3, “Accessing non-database files defined in the data dictionary” on page 13-14.

11.7.3 Demonstration

Objective: This report matches a student master file (STUDENT-MAST), which is defined to the data dictionary, with a VSAM file containing course descriptions and prerequisite courses.

Procedure

- A **DATABASE** parameter with the **DICTNAME=** option specifies the alternate dictionary DOCUDICT.
- The **FN=** option on the **IN** parameter specifies the STUDENT-MAST file.

Partial code

```
col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN 257      FN=STUDENT-MAST  MK=COURSE-CODE

IN 96 96 VS MK=COURSE-NUM
REC COURSE-NUM    1   3
REC DESC          4   30
REC PREQ          34          GROUP AA 3.4
REC COURSE        1   3   ELMNT AA
```

11.8 Creating unique match-key names

11.8.1 What you can do

You can assign unique names to reference entities in the data dictionary. You may use either the primary (data dictionary) name or the assigned name. Just be consistent.

11.8.2 How to do it

Use a **REC** parameter to assign a unique name to a field.

11.8.3 Demonstration

Objective: This example matches two files that are defined to the data dictionary. The match key for both files is PART-NUMBER. PART-NUMBER in the PARTS-INVENTORY file is assigned STOCK-REF-NUM as a unique name.

Procedure

- A **DD=** specification on the IN parameter provides alternative file assignments for each file.
- A **REC** parameter renames the PART-NUMBER field of the PARTS-INVENTORY file.
- An **MK=** specification of the new name of the PART-NUMBER (STOCK-REF-NUM) is the match key for the PARTS-INVENTORY file.

Partial code

```
col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN 130 DD=SYS011 FN=PARTS-INVENTORY MK=STOCK-REF-NUM
*          WAREHOUSE-CODE BIN-POINTER
REC STOCK-REF-NUM 1 10

IN 170 DD=SYS012 FN=PART-DESC MK=PART-NUMBER WAREHOUSE-ID
*          BIN-LOCATION
```

11.9 Qualified fields

11.9.1 What you can do

You can associate one record name in the data dictionary with fields that are specified on two or more INPUT parameters.

11.9.2 How to do it

Unique field names can be provided in a particular CA-Culprit run by coding:

- The **FN=** option of the INPUT parameter with the data dictionary file name
- The **MK=** option of the INPUT parameter with the field name followed by the record name known to the data dictionary, a comma, and a level number enclosed in parentheses
- **SYS010** in the JCL for the first file (level 1); **SYS011** in the JCL for the second file (level 2), and so on

11.9.3 Demonstration

Objective: This example matches three files that are defined in the data dictionary. Two files have MON-TRAN-FILE as a file name and TRAN-CODE as a key field name. All three files have ACCT-NO as a key field.

Procedure: The code uses level numbers to distinguish between field references (ACCOUNT,**1**, ACCOUNT,**2**, TRANSACTION,**1**, and TRANSACTION,**2**).

Partial code

```
col. 2
↓
DATABASE DICT NAME=DOCUDICT

IN 100 FN=MON-TRAN-FILE MK=ACCT-NO(ACCOUNT,1) TRAN-CODE
*      (TRANSACTION,1)
IN 100 FN=MON-TRAN-FILE MK=ACCT-NO(ACCOUNT,2) TRAN-CODE
*      (TRANSACTION,2)
IN 150 FN=CUST-FILE      MK=ACCT-NO(ACC-REC) MB=KEEP
```


Chapter 12. Using and Modifying Copied Code

12.1	Copying stored code (USE parameter)	12-4
12.1.1	What you can do	12-4
12.1.2	How to do it	12-4
12.2	Assigning values to symbolic fields (USE parameter)	12-5
12.2.1	What you can do	12-5
12.2.2	How to do it	12-5
12.2.3	Demonstration	12-5
12.3	Assigning default values to symbolic parameters (USE parameter)	12-8
12.3.1	What you can do	12-8
12.3.2	How to do it	12-8
12.3.3	Demonstration	12-8
12.4	Nesting the USE parameter	12-10
12.4.1	What you can do	12-10
12.4.2	How to do it	12-10
12.4.3	Demonstration	12-10
12.5	Modifying code (USE parameter)	12-11
12.5.1	What you can do	12-11
12.5.2	How to do it	12-11
12.5.3	Demonstration (1): Changing report numbers and character strings .	12-11
12.5.4	Demonstration (2): Using a KEEP clause	12-13
12.5.5	Demonstration (3): Using a DROP clause	12-14
12.5.6	Demonstration (4): Using the RENUMBER clause	12-15
12.6	Copying and modifying code (=COPY parameter)	12-18
12.6.1	What you can do	12-18
12.6.2	How to do it	12-18
12.6.3	Demonstration	12-18
12.7	Copying and modifying code (=MACRO parameter)	12-20
12.7.1	What you can do	12-20
12.7.2	How to do it	12-20
12.7.3	Demonstration (1): Providing symbolic parameter values	12-20
12.7.4	Demonstration (2): Modifying parameters and the report number .	12-22
12.8	Listing the contents of a data file	12-24
12.8.1	What you can do	12-24
12.8.2	How to do it	12-24
12.8.3	Demonstration (1): Printing fields with AMLIST3	12-24
12.8.4	Procedure	12-24
12.8.5	Demonstration (2): Modifying code using WITH VALUES and CHANGE	12-26

Frequently used chapters of CA-Culprit code can be stored for use by several reports or users. Use of stored parameters helps to establish standard file definitions, procedures, and reports. For example, you can store INPUT and REC parameters that define a shared input file, type 7 or 8 procedural code that performs an operation needed by several reports, or a set of commonly used extract functions.

Parameters to be inserted are maintained in card-image format and in data dictionaries, partitioned data sets (OS/390), source statement libraries (VSE/ESA), PANVALET libraries, and LIBRARIAN and sub-libraries. Stored code can be copied and modified to meet the requirements of a particular report by using one of the copied code parameters:

- USE
- =COPY
- =MACRO

12.1 Copying stored code (USE parameter)

12.1.1 What you can do

The USE parameter allows you to copy code that is stored on 80-byte records.

12.1.2 How to do it

Code a basic USE parameter by entering the following in the job stream:

1. The keyword **USE**
2. The name of the file or module that contains the CA-Culprit source code:

```
USE CULCODE  
/*
```

12.2 Assigning values to symbolic fields (USE parameter)

12.2.1 What you can do

You can use symbolic fields on the parameters in your stored or inline CA-Culprit code for values that are likely to change.

12.2.2 How to do it

Code:

1. Source code that incorporates symbolic parameter references:

```
010 START &&1
```

2. A standard USE parameter that names the source code to be copied:

```
USE CULCODE  
/*
```

3. The WITH VALUES keyword phrase:

```
USE CULCODE  
WITH VALUES ...  
/*
```

4. Keyword arguments (data values) or keyword expressions (labels), as needed, on the same line as WITH VALUES. The argument sequence must correspond to the parameter sequence:

```
USE CULCODE  
WITH VALUES (012191,MONTHLY REPORT)  
/*
```

12.2.3 Demonstration

Objective: This is a monthly report of branch office transactions created from code stored as RPT624. The report title and the dates printed in the heading of the report change each time the report code is run.

Procedure

- The **USE** parameter copies stored code (RPT624) containing symbolic parameters for values likely to change.
- The **WITH VALUES** clause provides arguments that are substituted for the symbolic parameters.

Complete code: Below follows the job stream and the stored code:

The job stream

```
USE RPT624
WITH VALUES (011591,021591,030191,MONTHLY REPORT)
/*
```

The stored code

```
col. 2
↓
IN 80 F
REC ACCOUNT    1      5      'ACCOUNT'
REC TRANS-IND   6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT   7      11     2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC TRANS-DATE  18     6      2      'DATE'
REC BRANCH      24     2      'BRANCH'
REC NAME        26     20     'NAME'
800UT 80
80SORT ACCOUNT 1 BRANCH  NAME  RPT-DAY  START  END
800  START &&1                      $First date (011591)
800  END   &&2                      $Second date (021591)
800  RPT-DAY &&3                    $Third date (030191)
803  &&4                      $Title (MONTHLY REPORT)
80410010 NAME
80410030 ACCOUNT
80410040 BRANCH
80410050 RPT-DAY      FD
80420010 'PERIOD: FROM'
80420024 START       FD
80420034 'TO'
80420038 END         FD
8051*010 TRANS-DATE   FD  HR
8051*020 TRANS-IND    HR
8051*030 TRANS-AMT    HR
8051*070 TRANS-DATE   FD  HR
8061*0200 'TOTAL TRANSACTIONS'
8061*030 TRANS-AMT
```

The sequential parameter listing, which is produced in a CA-Culprit run, shows the substitutions made for the symbolic parameters.

Sequential parameter listing:

```

09/27/99      SEQUENTIAL PARAMETER LISTING      CAGJFO V15.0 PAGE    1
00 ** SYSIN **
              USE RPT624
              WITH VALUES (011591,021591,030191,MONTHLY REPORT)
01 RPT624
C200138 INSTALLATION SECURITY OPTION IS NO
              REC ACCOUNT   1      5      'ACCOUNT'
              REC TRANS-IND 6      1      'DEPOSIT/' 'WITHDRAWAL'
              REC TRANS-AMT 7     11 2 DP=2 'AMOUNT OF' 'TRANSACTION'
              REC TRANS-DATE 18     6 2      'DATE'
              REC BRANCH    24     2      'BRANCH'
              REC NAME      26    20      'NAME'
800OUT 80
              8OSORT ACCOUNT 1 BRANCH NAME RPT-DAY START END
              800 START 011591           $First date (011591)
              800 END 021591           $Second date (021591)
              800 RPT-DAY 030191       $Third date (030191)
              803 MONTHLY          $Title (MONTHLY REPORT)
80410010 NAME
80410030 ACCOUNT
80410040 BRANCH
80410050 RPT-DAY      FD
80420010 'PERIOD: FROM'
80420024 START        FD
80420034 'TO'
80420038 END         FD
8051*010 TRANS-DATE   FD  HR
8051*020 TRANS-IND    HR
8051*030 TRANS-AMT    HR
8051*070 TRANS-DATE   FD  HR
8061*0200 'TOTAL TRANSACTIONS'
8061*030 TRANS-AMT

```

The report:

REPORT NO. 80 MONTHLY		\$Title (09/27/91 PAGE 1	
SHARON ARMSTRONG	15060	32	03/01/91
PERIOD: FROM 01/15/91 TO 02/15/91			
DATE	DEPOSIT/ WITHDRAWAL	AMOUNT OF TRANSACTION	DATE
01/15/91	D	10,099.01	01/15/91
01/16/91	W	990.11	01/16/91
01/20/91	D	100,990.11	01/20/91
TOTAL TRANSACTIONS		112,079.23	

REPORT NO. 80 MONTHLY		\$Title (09/27/91 PAGE 2	
ARTHUR LINK	16070	32	03/01/91
PERIOD: FROM 01/15/91 TO 02/15/91			
DATE	DEPOSIT/ WITHDRAWAL	AMOUNT OF TRANSACTION	DATE
01/01/91	D	1,080.04	01/01/91
TOTAL TRANSACTIONS		1,080.04	

12.3 Assigning default values to symbolic parameters (USE parameter)

12.3.1 What you can do

You can assign default values to symbolic parameters in your stored or inline code and use keywords to provide documentation.

12.3.2 How to do it

Code:

- The word **DEFAULT** on the first line of *stored* code.
If used inline, the DEFAULT clause is entered as the last of the USE parameter clauses.
- Symbolic parameter names entered sequentially, starting with &&1.
- A keyword for each symbolic parameter name. (Any keywords in the WITH VALUES clause must match those in the DEFAULT clause.)
- The value for the keyword.

12.3.3 Demonstration

Objective: This example uses a DEFAULT clause to produce the same report as the example using a WITH VALUES clause.

Procedure

- Default values are used for symbolic parameters whose values do not appear on the WITH VALUES clause.
- Keywords **START**, **END**, and **TITLE** provide documentation.
- **RPT-DAY**, the keyword for &&3, is specified on both the **WITH VALUES** clause and the **DEFAULT** clause.

The value given on the WITH VALUES clause (030191) overrides the value on the DEFAULT clause.

Complete code: Below follows the job stream and the stored code:

The job stream

```
USE RPT627
WITH VALUES (RPT-DAY=030191)      $Overriding value
/*
```

The stored code

```

col. 2
↓
DEFAULT &&1=START=011591      $Default values if not overridden
&&2=END=021591
&&3=RPT-DAY=022891
&&4=TITLE='MONTHLY REPORT'

IN 80
REC ACCOUNT    1      5      'ACCOUNT'
REC TRANS-IND   6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT   7      11     2 DP=2  'AMOUNT OF' 'TRANSACTION'
REC TRANS-DATE  18     6      2      'DATE'
REC BRANCH      24     2      'BRANCH'
REC NAME        26     20     'NAME'

800UT 80
80SORT ACCOUNT 1 BRANCH  NAME  RPT-DAY  START  END
800  START &&1
800  END   &&2
800  RPT-DAY &&3
803 &&4
80410010 NAME
80410030 ACCOUNT
80410040 BRANCH
80410050 RPT-DAY      FD
80420010 'PERIOD: FROM'
80420024 START       FD
80420034 'TO'
80420038 END        FD
8051*010 TRANS-DATE  FD  HR
8051*020 TRANS-IND   HR
8051*030 TRANS-AMT   HR
8051*070 TRANS-DATE  FD  HR
8061*0200 'TOTAL TRANSACTIONS'
8061*030 TRANS-AMT

```

Result

REPORT NO. 80	MONTHLY REPORT	09/27/99	PAGE	1
SHARON ARMSTRONG	15060	32	03/01/91	
PERIOD: FROM	01/15/91	TO	02/15/91	
DATE	DEPOSIT/ WITHDRAWAL	AMOUNT OF TRANSACTION	DATE	
01/15/91	D	10,099.01	01/15/91	
01/16/91	W	990.11	01/16/91	
01/20/91	D	100,990.11	01/20/91	
TOTAL TRANSACTIONS		112,079.23		

REPORT NO. 80	MONTHLY REPORT	09/27/99	PAGE	2
ARTHUR LINK	16070	32	03/01/91	
PERIOD: FROM	01/15/91	TO	02/15/91	
DATE	DEPOSIT/ WITHDRAWAL	AMOUNT OF TRANSACTION	DATE	
01/01/91	D	1,080.04	01/01/91	
TOTAL TRANSACTIONS		1,080.04		

12.4 Nesting the USE parameter

12.4.1 What you can do

You can nest the USE parameter. One or more USE parameters and WITH VALUES clauses can be included in stored code.

If one USE parameter appears in the stored code, it is treated as a nested parameter in the job stream at run-time.

12.4.2 How to do it

Code:

1. The **USE** parameter followed by an asterisk (*), and a **WITH VALUES** clause at the beginning of the code affected
2. **END** at the end of the code affected by the value assignments

12.4.3 Demonstration

Objective: This is a partial example that produces the same report as that shown earlier for the reports generated from the WITH VALUES and DEFAULT clauses.

Procedure

- A **USE** parameter in the job stream copies the stored code.
- **USE *** parameter and a **WITH VALUES** clause assign values to the symbolic parameters.
- An **END** clause signals the end of the code affected by the **USE** parameter.

The job stream:

```
USE RPT624
/*
```

The stored code:

```
col. 2
↓
USE *                      $Start of code sequence affected by changes
WITH VALUES (011591,021591,030191,'MONTHLY REPORTS')
IN 80
REC ACCOUNT 1 5  'ACCOUNT'
.
.
.
END                      $End of code incorporating changes
```

12.5 Modifying code (USE parameter)

12.5.1 What you can do

You can change report numbers and character strings, drop or keep parameters, and renumber type 7 and type 8 sequence numbers with the USE parameter.

12.5.2 How to do it

- To modify code, use one or more **CHANGE** clauses for each USE parameter.
Only one clause can specify a report number change.
- To eliminate or retain code, use one or more **DROP** or **KEEP** clauses.
If both **DROP** and **KEEP** are coded, precedence is given to the **DROP** clause.
- To renumber type 7 and type 8 sequence numbers, use one or more **RENUMBER** clauses.
The **RENUMBER** clause must be the last clause coded on the USE parameter.

12.5.3 Demonstration (1): Changing report numbers and character strings

Objective: This report is derived from stored code. The code is changed to provide updated branch office account activity information.

Procedure

- The report number is changed from 80 to 01.
- The title is changed from ACCOUNT ACTIVITY BY BRANCH to BRANCH ACCOUNTS.
- A type 5 parameter is changed to a type 4 to allow the branch number to print as a heading.

Complete code

```

col. 2
↓
IN 80 F
USE *
    CHANGE RPTNO TO 01 AND           $Chained together with AND
        '51*010' 4/9 TO '410010' AND
        'ACCOUNT ACTIVITY BY BRANCH' TO 'BRANCH ACCOUNTS'
    REC T-ACCOUNT      1      5          'ACCOUNT'
    REC T-TRANS-IND   6      1          'DEPOSIT/' 'WITHDRAWAL'
    REC T-TRANS-AMT   7     11  2 DP=2  'AMOUNT OF' 'TRANSACTION'
    REC T-DATE        18     6  2          'DATE'
    REC T-BRANCH      24      2
    REC T-NAME        26     20          'NAME'
    800UT D           $Report number changed
    80SORT T-BRANCH 1
    803 ACCOUNT ACTIVITY BY BRANCH           $Title changed
    8051*010 T-BRANCH                      $Changed to a type 4 parameter
    8051*020 T-ACCOUNT          HR
    8051*030 T-NAME             HR
    8051*050 T-TRANS-AMT        HR
    8051*060 T-TRANS-IND       HR
    8051*070 T-DATE            FD          HR
END

```

Result:**Sequential parameter listing:**

09/27/99	SEQUENTIAL PARAMETER LISTING	CAGJFO V15.0	PAGE	1
IN 80 F				
C200138 INSTALLATION SECURITY OPTION IS NO				
00 ** SYSIN **				
USE *				
CHANGE RPTNO TO 01 AND \$Chained together with AND				
'51*010' 4/9 TO '410010' AND				
'ACCOUNT ACTIVITY BY BRANCH' TO 'BRANCH ACCOUNTS'				
REC T-ACCOUNT 1 5 'ACCOUNT'				
REC T-TRANS-IND 6 1 'DEPOSIT/' 'WITHDRAWAL'				
REC T-TRANS-AMT 7 11 2 DP=2 'AMOUNT OF' 'TRANSACTION'				
REC T-DATE 18 6 2 'DATE'				
REC T-BRANCH 24 2				
REC T-NAME 26 20 'NAME'				
01OUT D \$Report number changed				
01SORT T-BRANCH 1				
013 BRANCH ACCOUNTS \$Title changed				
01410010 T-BRANCH \$Changed to a type 4 parameter				
0151*020 T-ACCOUNT HR				
0151*030 T-NAME HR				
0151*050 T-TRANS-AMT HR				
0151*060 T-TRANS-IND HR				
0151*070 T-DATE FD HR				
END				

Report:

REPORT NO. 01 32		BRANCH ACCOUNTS		09/27/99 PAGE	1
ACCOUNT	NAME	AMOUNT OF TRANSACTION	DEPOSIT/ WITHDRAWAL	DATE	
15060	SHARON ARMSTRONG	10,099.01	D	01/15/91	
15060	SHARON ARMSTRONG	990.11	W	01/16/91	
15060	SHARON ARMSTRONG	100,990.11	D	01/20/91	
16070	ARTHUR LINK	1,080.04	D	01/01/91	
19235	GARY NOBLES	80.04	W	01/02/91	
21056	AMOS JOHNSON	1.02	W	01/20/91	

REPORT NO. 01 35		BRANCH ACCOUNTS		09/27/99 PAGE	2
ACCOUNT	NAME	AMOUNT OF TRANSACTION	DEPOSIT/ WITHDRAWAL	DATE	
23055	JACK JACKSON	500.00	W	01/02/91	
23055	JACK JACKSON	50,000.00	D	01/25/91	
23055	JACK JACKSON	30.00	D	01/26/91	
29557	IRWIN TRIMBLE	500,001.00	D	01/04/91	

12.5.4 Demonstration (2): Using a KEEP clause

Objective: This example changes inline code to produce a report of individual account activity.

Procedure: A **KEEP** clause is used to retain the title and three type 5 fields (ACCOUNT, TRANS-AMT, and TRANS-IND). All other fields are dropped.

Complete code

```

col. 2
↓
IN 80 F
USE *
    CHANGE RPTNO TO 01 AND
        'ACCOUNT ACTIVITY BY BRANCH' TO 'ACCOUNT ACTIVITY'
    KEEP 3 AND 'ACCOUNT' AND 'TRANS-AMT' AND 'TRANS-IND'
    REC ACCOUNT    1      5      'ACCOUNT'
    REC TRANS-IND  6      1      'DEPOSIT/' 'WITHDRAWAL'
    REC TRANS-AMT  7      11     2 DP=2 'AMOUNT OF' 'TRANSACTION'
    REC DATE       18     6      2      'DATE'
    REC BRANCH     24     2
    REC NAME       26     20     'NAME'
    800UT D          $Report number changed
    80SORT NAME 0
    803 ACCOUNT ACTIVITY BY BRANCH      $Title changed
    8051*010 BRANCH
    8051*020 ACCOUNT           HR   $Kept
    8051*030 NAME            HR
    8051*050 TRANS-AMT         HR   $Kept
    8051*060 TRANS-IND         HR   $Kept
    8051*070 DATE FD          HR
END

```

Sequential parameter listing:

```

09/27/99          SEQUENTIAL PARAMETER LISTING      CAGJF0  V15.0 PAGE    1
00 ** SYSIN **           IN 80 F
C200138 INSTALLATION SECURITY OPTION IS NO

USE *
  CHANGE RPTNO TO 01 AND
    'ACCOUNT ACTIVITY BY BRANCH' TO 'ACCOUNT ACTIVITY'
    KEEP 3 AND 'ACCOUNT' AND 'TRANS-AMT' AND 'TRANS-IND'
01 ** SYSIN **
REC ACCOUNT   1      5      'ACCOUNT'
REC TRANS-IND 6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT 7      11     2 DP=2 'AMOUNT OF' 'TRANSACTION'
013 ACCOUNT ACTIVITY           $Title changed
0151*020 ACCOUNT             HR      $Kept
0151*050 TRANS-AMT           HR      $Kept
0151*060 TRANS-IND           HR      $Kept

```

Report:

REPORT NO. 01	ACCOUNT ACTIVITY	
ACCOUNT	AMOUNT OF TRANSACTION	09/27/99 PAGE 1
15060	10,099.01	D
15060	990.11	W
15060	100,990.11	D
16070	1,080.04	D
19235	80.04	W
21056	1.02	W
23055	500.00	W
23055	50,000.00	D
23055	30.00	D
29557	500,001.00	D
30115	500,001.00	D
30115	50,001.00	D
31113	36.00	W
31113	36.00	D
31275	5,000,000.00	D
33470	7,590,001.30	D
33470	1.30	D
34440	195.01	W
34440	2,000.01	D
34440	95.01	D
34440	100.00	D
36682	4,000.78	W
69876	9,009,901.15	D
69876	637.55	D
99083	50,001.30	D
99083	6,468.52	D
		22,877,247.26

12.5.5 Demonstration (3): Using a DROP clause

Objective: This example uses the DROP clause to produce the same report as that shown for the KEEP clause in Demonstration (2).

Procedure: A **DROP** clause is used to leave out DATE, BRANCH, and NAME. ACCOUNT, TRANS-IND, and TRANS-AMT are kept. ADD A LABEL HERE

```

col. 2
↓
IN 80 F
USE *
    CHANGE RPTNO TO 01 AND
        'ACCOUNT ACTIVITY BY BRANCH' TO 'ACCOUNT ACTIVITY'
    DROP 'DATE' AND 'BRANCH' AND 'NAME'
REC ACCOUNT      1      5      'ACCOUNT'
REC TRANS-IND   6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT   7     11 2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC DATE        18      6 2      'DATE'
REC BRANCH       24      2
REC NAME         26     20      'NAME'
800UT D
80SORT NAME 0                      $Dropped
803 ACCOUNT ACTIVITY BY BRANCH      $Title changed
8051*010 BRANCH                   $Dropped
8051*020 ACCOUNT                  HR
8051*030 NAME                     HR      $Dropped
8051*050 TRANS-AMT                HR
8051*060 TRANS-IND                HR
8051*070 DATE FD                 HR      $Dropped
END

```

12.5.6 Demonstration (4): Using the RENUMBER clause

Objective: This example renames type 7 sequence numbers in inline code to produce a current Account Activity Report.

Procedure: The **RENUMBER** clause is used to change type 7 sequence numbers from 010, 020, 040, and 050 to 100, 200, 300, and 400, respectively.

Complete code

```

col. 2
↓
IN 80 F
USE *
    CHANGE 'ACCOUNT ACTIVITY BY BRANCH' TO 'ACCOUNT ACTIVITY'
    DROP 'BRANCH' AND 'NAME'
    RENUMBER 7 1/100 TO 100 BY 100
    REC ACCOUNT      1      5          'ACCOUNT'
    REC TRANS-IND   6      1          'DEPOSIT/' 'WITHDRAWAL'
    REC TRANS-AMT   7     11 2 DP=2  'AMOUNT OF' 'TRANSACTION'
    REC MMDDYY      18      6 2
    REC BRANCH      24      2
    REC NAME        26     20          'NAME'
    01OUT D
    01SORT NAME 0
    010 MMDD
    010 YY
    010 YYMMDD
    013 ACCOUNT ACTIVITY BY BRANCH                      $Changed
    0151*010 BRANCH                                     $Dropped
    0151*020 ACCOUNT          HR
    0151*030 NAME           HR
    0151*050 TRANS-AMT       HR
    0151*060 TRANS-IND       HR
    0151*070 YYMMDD FD       HH 'DATE'
    017010 IF TRANS-AMT LE 100  DROP                     $Renumbered
    017020 COMPUTE TRUNC MMDDYY / 100  MMDD
    017040 COMPUTE MMDDYY - (MMDD X 100)  YY
    017050 COMPUTE (YY X 10000) + MMDD YYMMDD
END

```

Result:**Sequential parameter listing:**

09/27/99	SEQUENTIAL PARAMETER LISTING	CAGJF0 V15.0 PAGE 1
<pre> 00 ** SYSIN ** IN 80 F C200138 INSTALLATION SECURITY OPTION IS NO USE * CHANGE 'ACCOUNT ACTIVITY BY BRANCH' TO 'ACCOUNT ACTIVITY' DROP 'BRANCH' AND 'NAME' RENUMBER 7 1/100 TO 100 BY 100 01 ** SYSIN ** REC ACCOUNT 1 5 'ACCOUNT' REC TRANS-IND 6 1 'DEPOSIT/' 'WITHDRAWAL' REC TRANS-AMT 7 11 2 DP=2 'AMOUNT OF' 'TRANSACTION' REC MMDDYY 18 6 2 REC BRANCH 24 2 REC NAME 26 20 'NAME' 01OUT D 01SORT NAME 0 010 MMDD 010 YY 010 YYMMDD 013 ACCOUNT ACTIVITY \$Changed 0151*020 ACCOUNT HR 0151*050 TRANS-AMT HR 0151*060 TRANS-IND HR 0151*070 YYMMDD FD HH 'DATE' 017100 IF TRANS-AMT LE 100 DROP \$Renumbered 017200 COMPUTE TRUNC MMDDYY / 100 MMDD 017300 COMPUTE MMDDYY - (MMDD X 100) YY 017400 COMPUTE (YY X 10000) + MMDD YYMMDD END </pre>		

Report:

REPORT NO. 01	ACCOUNT ACTIVITY AMOUNT OF TRANSACTION	\$Ch DEPOSIT/ WITHDRAWAL	09/27/99 PAGE 1
ACCOUNT			DATE
15060	10,099.01	D	91/01/15
15060	990.11	W	91/01/16
15060	100,990.11	D	91/01/20
16070	1,080.04	D	91/01/01
23055	500.00	W	91/01/02
23055	50,000.00	D	91/01/25
29557	500,001.00	D	91/01/04
30115	500,001.00	D	91/01/04
30115	50,001.00	D	91/01/05
31275	5,000,000.00	D	91/01/10
33470	7,590,001.30	D	91/01/13
34440	195.01	W	91/01/12
34440	2,000.01	D	91/01/24
36682	4,000.78	W	91/01/12
69876	9,009,901.15	D	91/01/01
69876	637.55	D	91/01/02
99083	50,001.30	D	91/01/20
99083	6,468.52	D	91/01/22

12.6 Copying and modifying code (=COPY parameter)

12.6.1 What you can do

You can copy stored code, change report numbers, and suppress INPUT parameters with the =COPY parameter.

12.6.2 How to do it

1. Code =COPY in the SYSIN file, beginning in column 1.
2. On the same line, include one or all of the following:
 - Source of the code to be used:
 - The name of the file
 - The name of the module
 - An asterisk (*) if inline code follows
 - A RPTNO= keyword followed by the 2-digit report number being assigned to this report

12.6.3 Demonstration

Objective: This example copies stored record descriptions and parameters.

Procedure

- An =COPY parameter is used to copy record descriptions from the file RECS.
- An =COPY parameter is used to copy parameters
- The report number of PARAMS is changed to 02 (RPTNO=02).

Complete code

```
col. 2
↓
=COPY RECS
=COPY PARAMS RPTNO=02
/*
```

Result: A plus sign (+), in the sequential parameter listing below, identifies copied lines:

Sequential parameter listing

```

09/26/99          SEQUENTIAL PARAMETER LISTING      CAGJFO  V15.0  PAGE   1
00 ** SYSIN **          =COPY RECS
C200138 INSTALLATION SECURITY OPTION IS NO

+ IN 200
+ REC EMPLOYEE      5  20
+ REC FIRST-NAME    5  10
+ REC LAST-NAME     15 10
+ REC DEPARTMENT    115 25
+ REC SALARY        160 5  3 DP=2
+ REC TITLE         171 20

=COPY PARAMS RPTNO=02
+ 02$"PARAMS"
+ 023 EMPLOYEE SALARY LISTING BY DEPARTMENT
+ 02SORT DEPARTMENT - SALARY D EMPLOYEE
+ 0251*010 DEPARTMENT      HH 'DEPARTMENT NAME'
+ 0251*020 LAST-NAME      HH '                           EMPLOYEE NAME'
+ 0251*025 FIRST-NAME
+ 0251*030 TITLE          HH 'JOB TITLE'
+ 0251*040 SALARY         HH 'ANNUAL SALARY'

```

Report:

REPORT NO. 02	EMPLOYEE SALARY LISTING BY DEPARTMENT	09/26/99	PAGE	1	JOB TITLE	ANNUAL SALARY
DEPARTMENT NAME	EMPLOYEE NAME					
ACCOUNTING AND PAYROLL	JENSON	RUPERT	MGR ACCTNG/PAYROLL	82,000.00		
ACCOUNTING AND PAYROLL	KIMBALL	MARIANNE	ACCOUNTANT	45,000.00		
ACCOUNTING AND PAYROLL	HUTTON	EDWARD	FINANCIAL ANALYST	44,000.00		
ACCOUNTING AND PAYROLL	BLOOMER	JUNE	PAYROLL CLERK	15,000.00		
ACCOUNTING AND PAYROLL	KING	DORIS	AR CLERK	14,500.00		
ACCOUNTING AND PAYROLL	NICEMAN	BRIAN	AP CLERK	14,000.00		
				214,500.00		
THERMOREGULATION	WILCO	ROGER	MGR THERMOREGULATION	80,000.00		
THERMOREGULATION	FINN	PHINEAS	KEEPER OF BALLOONS	45,000.00		
THERMOREGULATION	CLOTH	TERRY	HUMIDITY CONTROL CLK	38,000.00		
THERMOREGULATION	TIME	MARK	WINTERIZER	33,000.00		
THERMOREGULATION	KASPAR	JOE	WINTERIZER	31,000.00		
				227,000.00		
				2,522,500.00		

12.7 Copying and modifying code (=MACRO parameter)

12.7.1 What you can do

You can copy or modify code for a single CA-Culprit run with the =MACRO parameter. Symbolic parameters can be used in copied or inline code as a substitute for arguments; parameters can be changed or suppressed.

12.7.2 How to do it

- To code changes affecting copied code, enter =MACRO after the INPUT parameter.
- To code changes affecting inline code, enter =MACRO * at the beginning of the code to be changed.
- To modify code, enter the values for symbolic parameters, enclosed in parentheses, on the same line as the =MACRO (*) parameter.
- To eliminate or retain code, enter =DROP and =CHANGE on the line immediately following the =MACRO (*) parameter.
- Enter =MEND immediately after the =MACRO clauses.
- Code a second =MACRO */ =MEND sequence before inline code that is not to be changed.

12.7.3 Demonstration (1): Providing symbolic parameter values

Objective: This report uses copied code that contains symbolic fields to list employees with salaries less than \$60,000 in all departments except Brainstorming.

Procedure: The =MACRO * parameter provides the values for the symbolic parameters.

Complete code

```

col. 2
↓
IN 200 F 400 PS(TAPE)
REC EMP-NAME      5 25          'EMPLOYEE NAME'
REC DEPARTMENT    115 45        'DEPARTMENT'
REC SALARY        160 5 3 DP=2
013 EMPLOYEE SALARY LISTING
010 COUNT 1
=MACRO * (DEPARTMENT 'BRAIN STORMING      ' EMP-NAME SALARY 60000)
=MEND
0151*000 COUNT
0151*010 &&3           $EMP-NAME
0151*020 &&4           SZ=7 F2 HF $SALARY
0161*010 'TOTAL COUNT'
0161*020 COUNT        SZ=8
017010 IF &&1 EQ &&2 DROP      $If DEPARTMENT equal BRAINSTORMING
017     IF &&4 GE &&5 DROP      $If SALARY greater than 60000

```

Result: In the sequential parameter listing, below:

- A plus sign (+) identifies the substitutions made.
- An asterisk (*) indicates a continuation line for parameters having more than one substitution.

Sequential parameter listing:

09/24/99	SEQUENTIAL PARAMETER LISTING	CAGJF0 V15.0 PAGE 1
<pre> 00 ** SYSIN ** IN 200 F 400 PS(TAPE) C200138 INSTALLATION SECURITY OPTION IS NO REC EMP-NAME 5 25 'EMPLOYEE NAME' REC DEPARTMENT 115 45 'DEPARTMENT' REC SALARY 160 5 3 DP=2 013 EMPLOYEE SALARY LISTING 010 COUNT 1 =MACRO * (DEPARTMENT 'BRAIN STORMING ' EMP-NAME SALARY 60000) + =MEND + 0151*000 COUNT + 0151*010 &&3 \$EMP-NAME ++ 0151*010 EMP-NAME \$EMP-NAME + 0151*020 &&4 SZ=7 F2 HF \$SALARY ++ 0151*020 SALARY SZ=7 F2 HF \$SALARY + 0161*010 'TOTAL COUNT' + 0161*020 COUNT SZ=8 + 017010 IF &&1 EQ &&2 DROP \$If DEPARTMENT equal BRAINSTORMING ++ 017010 IF DEPARTMENT +++EQ 'BRAIN STORMING ' +++DROP \$If DEPARTMENT equal BRAINSTORMING + 017 IF &&4 GE &&5 DROP \$If SALARY greater than 60000 ++ 017 IF SALARY +++GE 60000 +++DROP \$If SALARY greater than 60000 </pre>		

Report:

REPORT NO. 01	EMPLOYEE SALARY LISTING	09/24/99 PAGE 1
		SALARY
JUNE	BLOOMER	15,000.00
EDWARD	HUTTON	44,000.00
MARIANNE	KIMBALL	45,000.00
DORIS	KING	14,500.00
BRIAN	NICEMAN	14,000.00
JANE	FERNDALE	22,500.00
GEORGE	FONRAD	14,750.00
ROBIN	GARDNER	14,000.00
DOUGLAS	KAHALLY	20,000.00
TERENCE	KLWELLEN	43,000.00
SANDY	KRAAMER	14,000.00
JAMES	GALLWAY	33,000.00
PERCY	GRANGER	34,500.00
VLADIMIR	HEAROWITZ	33,000.00
JAMES	JACOBI	55,000.00
JULIE	JENSEN	37,000.00
LARRY	LITERATA	37,500.00
KATHERINE	O'HEARN	42,500.00
RALPH	TYRO	20,000.00
TOTAL COUNT		44

12.7.4 Demonstration (2): Modifying parameters and the report number

Objective: This example modifies the report number and parameter of copied code to produce a report on employee salaries.

Procedure

- An =DROP clause is used to omit the INPUT and type 6 parameters.
- An =CHANGE clause is used to change the report number from 01 to 07.

Complete code

```
col. 2
↓
IN 200
=MACRO RPT631A (DEPARTMENT 'BRAIN STORMING      ' EMP-NAME SALARY 60000)
=DROP INPUT TYPE=6 $Drop INPUT and type 6 parameters
=CHANGE RPTNO=07   $Change the report number
=MEND
/*
```

Result: In the sequential parameter listing below:

- A plus sign (+) indicates copied parameters.
- A double plus sign (++) indicates a copied parameter and a value substitution.
- An asterisk (*) indicates a continuation line when more than one substitution occurs.

Sequential parameter listing:

```

09/26/99          SEQUENTIAL PARAMETER LISTING      CAGJF0  V15.0  PAGE   1
00 ** SYSIN **          IN 200
C200138 INSTALLATION SECURITY OPTION IS NO

=MACRO RPT631A (DEPARTMENT 'BRAIN STORMING      ' EMP-NAME SALARY 60000)
=DROP INPUT TYPE=6
=CHANGE RPTNO=07
=MEND

+ IN 200
W C200052 INPUT CARD DROPPED
+ REC EMP-NAME      5 25          'EMPLOYEE NAME'
+ REC DEPARTMENT    115 45        'DEPARTMENT'
+ REC SALARY        160 5 3 DP=2
+ 073 EMPLOYEE SALARY LISTING
+ 070 COUNT 1
+ 0751*010 &&3     HR
++ 0751*010 EMP-NAME HR
+ 0751*020 &&4     SZ=7 F2 HF
++ 0751*020 SALARY  SZ=7 F2 HF
+ 0761*010 'TOTAL COUNT'
+ 0761*020 COUNT    SZ=8
+ 077010 IF &&1 EQ &&2 DROP
++ 077010 IF DEPARTMENT
+++EQ 'BRAIN STORMING '
+++DROP
+ 077    IF &&4 GE &&5 DROP
++ 077    IF SALARY
+++GE 60000
+++DROP

```

Report:

REPORT NO. 07	EMPLOYEE SALARY LISTING EMPLOYEE NAME	09/26/99 PAGE 1 SALARY
JUNE	BLOOMER	15,000.00
EDWARD	HUTTON	44,000.00
MARIANNE	KIMBALL	45,000.00
DORIS	KING	14,500.00
BRIAN	NICEMAN	14,000.00
JANE	FERNDALE	22,500.00
GEORGE	FONRAD	14,750.00
ROBIN	GARDNER	14,000.00
DOUGLAS	KAHALLY	20,000.00
TERENCE	KLWELLEN	43,000.00
SANDY	KRAAMER	14,000.00
HERBERT	LIPSICH	18,500.00
NANCY	TERNER	13,000.00
BETH	CLOUD	52,750.00
TERRY	CLOTH	38,000.00
PHINEAS	FINN	45,000.00
JOE	KASPAR	31,000.00
MARK	TIME	33,000.00
JANE	DOUGH	33,000.00
JAMES	GALLWAY	33,000.00
PERCY	GRANGER	34,500.00
VLADIMIR	HEAROWITZ	33,000.00
JAMES	JACOBI	55,000.00
JULIE	JENSEN	37,000.00
LARRY	LITERATA	37,500.00
KATHERINE	O'HEARN	42,500.00
RALPH	TYRO	20,000.00
		93,500.00

12.8 Listing the contents of a data file

12.8.1 What you can do

You can list the contents of a data file before CA-Culprit reports are run against a file. Three to ten fields for a specified number of records can be printed in the order of appearance in the file.

You can also sort records, select records, and print total lines.

12.8.2 How to do it

Code:

1. An =MACRO or USE parameter with:
 - AMLIST immediately followed by a number from 3 to 10, depending on how many fields are to be listed
 - The WITH VALUES clause if you code the USE parameter
 - An argument list enclosed in parentheses (the number of records to be listed, the subtitle, the field names in the order of listing)
2. The =MEND parameter, if =MACRO is used

12.8.3 Demonstration (1): Printing fields with AMLIST3

Objective: This report prints three fields (EMP-NAME, TITLE, and SALARY) of the first ten records in the input file.

12.8.4 Procedure

- =MACRO is used to copy the input file.
- AMLIST3 calls the AMLIST routine to print three fields (EMP-NAME, TITLE, and SALARY) from the first ten records in the input file.
- The heading of the report is COMPANY EMPLOYEES.
- Column headings are the field names.
- =MEND signals the end of the code affected by the =MACRO parameter.

Complete code

```

col. 2
↓
IN 200
REC EMP-NAME 5 25
REC SALARY 160 5 3 DP=2
REC TITLE 171 20
=MACRO AMLIST3 (10 'COMPANY EMPLOYEES' EMP-NAME TITLE SALARY)
=MEND
/*

```

Sequential parameter listing:

09/26/99	SEQUENTIAL PARAMETER LISTING	CAGJFO V15.0 PAGE 1
<pre> 00 ** SYSIN ** IN 200 C200138 INSTALLATION SECURITY OPTION IS NO REC EMP-NAME 5 25 REC SALARY 160 5 3 DP=2 REC TITLE 171 20 =MACRO AMLIST3 (10 'COMPANY EMPLOYEES' EMP-NAME TITLE SALARY) =MEND + 03\$00****CULPRIT ROUTINE-AMLIST3 + 030 SEQUENCE ALL + 033DETAIL LIST + 034100010&&2 ++ 034100010'COMPANY EMPLOYEES' + 03420001 '' + 0351*010 &&3 HF ++ 0351*010 EMP-NAME HF + 0351*020 &&4 HF ++ 0351*020 TITLE HF + 0351*030 &&5 HF ++ 0351*030 SALARY HF + 0368*001 '' + 037110SEQUENCE A 1 SEQUENCE \$ COUNT RECORDS READ + 037 M &&1 ALL \$ GET MAXIMUM ++ 037 M 10 ALL \$ GET MAXIMUM + 037 ALL EQ 0 TAKE \$ TAKE ALL RECORDS + 037 SEQUENCE GT &&1 DROP \$ DROP IF OVER MAXIMUM ++ 037 SEQUENCE GT 10 DROP \$ DROP IF OVER MAXIMUM </pre>		

Report:

REPORT NO. 03	DETAIL LIST	09/26/99 PAGE 1	
COMPANY EMPLOYEES			
	EMP-NAME	TITLE	SALARY
JUNE	BLOOMER	PAYROLL CLERK	15,000.00
EDWARD	HUTTON	FINANCIAL ANALYST	44,000.00
RUPERT	JENSON	MGR ACCTNG/PAYROLL	82,000.00
MARIANNE	KIMBALL	ACCOUNTANT	45,000.00
DORIS	KING	AR CLERK	14,500.00
BRIAN	NICEMAN	AP CLERK	14,000.00
HERBERT	CRANE	MGR COMPUTER OPS	75,000.00
JANE	FERNDALE	COMPUTER OPERATOR	22,500.00
GEORGE	FONRAD	DATA ENTRY CLERK	14,750.00
ROBIN	GARDNER	DATA ENTRY CLERK	14,000.00

12.8.5 Demonstration (2): Modifying code using WITH VALUES and CHANGE

Objective: This report is generated from the source code used for Demonstration (1). Three fields from the first ten records of the input file are listed.

Procedure

- The **USE** parameter is coded with the **AMLIST** routine.
- The **WITH VALUES** clause specifies a subtitle for the report by enclosing the literal that is to appear on the generated type 4 parameter ('COMPANY EMPLOYEES') in double quotation marks.
- The **CHANGE** clause changes the report number from 03 to 01.
- The **SORT** parameter sorts the data by employee title.
- A subtotal prints for every title.
- **Type 7** process code selects the first ten records in the file in which the salary amount is less than or equal to \$25,000.

Note: CA-Culprit sorts process parameters in ascending order by sequence number. Therefore, be sure to code a sequence number that reflects the sequence in which record selection is to occur.

Complete code

```

col. 2
↓
IN 200
REC EMP-NAME   5   25
REC SALARY     160   5 3 DP=2
REC TITLE      171   20
USE AMLIST3
WITH VALUES (10 "'COMPANY EMPLOYEES'" EMP-NAME TITLE SALARY)
CHANGE RPTNO TO 01           $Change the report number
01SORT TITLE -             $Sort the data
017010 SALARY GT 25000 DROP    $Select the salary amounts
0161*030 SALARY             $Total the salary amounts

```

Input parameter listing:

09/26/99			INPUT PARAMETER LISTING					CAGJF0 V15.0 PAGE 3	

PROCESS	USER	INTERNAL	PROCESS STATEMENT						

01	7	010	1	SALARY	GT	25000	DROP	\$Select the salary amounts	
01	7	110	2	SEQUENCE	+	1	SEQUENCE	\$ COUNT RECORDS READ	
01	7		3	M	10	ALL	\$ GET MAXIMUM		
01	7		4	ALL	EQ	0	TAKE	\$ TAKE ALL RECORDS	
01	7		5	SEQUENCE	GT	10	DROP	\$ DROP IF OVER MAXIMUM	

Sequential parameter listing:

```

09/26/99          SEQUENTIAL PARAMETER LISTING          CAGJF0 V15.0 PAGE 1
00 ** SYSIN **          IN 200
C200138 INSTALLATION SECURITY OPTION IS NO

      REC EMP-NAME 5 25
      REC SALARY 160 5 3 DP=2
      REC TITLE 171 20
      USE AMLIST3
      WITH VALUES (10 "'COMPANY EMPLOYEES'" EMP-NAME TITLE SALARY)
      CHANGE RPTNO TO 01 $Change the report number
01 AMLIST3
      01$00****CULPRIT ROUTINE-AMLIST3
      010 SEQUENCE ALL
      013DETAIL LIST
      014100010'COMPANY EMPLOYEES'
      01420001 ''
      0151*010 EMP-NAME HF
      0151*020 TITLE HF
      0151*030 SALARY HF
      0168*001 ''
      017110SEQUENCE A 1 SEQUENCE $ COUNT RECORDS READ
      017 M 10 ALL $ GET MAXIMUM
      017 ALL EQ 0 TAKE $ TAKE ALL RECORDS
      017 SEQUENCE GT 10 DROP $ DROP IF OVER MAXIMUM
      01SORT TITLE - $Sort the data
      017010 SALARY GT 25000 DROP $Select the salary amounts
      0161*030 SALARY $Total the salary amounts
00 ** SYSIN **

```

Report:

REPORT NO. 01	DETAIL LIST	09/26/99 PAGE 1
COMPANY EMPLOYEES		
	EMP-NAME	TITLE
BRIAN	NICEMAN	AP CLERK
		14,000.00
		14,000.00
DORIS	KING	AR CLERK
		14,500.00
		14,500.00
JANE	FERNDALE	COMPUTER OPERATOR
DOUGLAS	KAHALLY	COMPUTER OPERATOR
HERBERT	LIPSICH	COMPUTER OPERATOR
		22,500.00
		20,000.00
		18,500.00
		61,000.00
GEORGE	FONRAD	DATA ENTRY CLERK
ROBIN	GARDNER	DATA ENTRY CLERK
SANDY	KRAAMER	DATA ENTRY CLERK
NANCY	TERNER	DATA ENTRY CLERK
		14,750.00
		14,000.00
		14,000.00
		13,000.00
		55,750.00
JUNE	BLOOMER	PAYROLL CLERK
		15,000.00
		15,000.00
		160,250.00

Chapter 13. Additional Standard File Facilities

13.1	Creating nonprint report output	13-4
13.1.1	What you can do	13-4
13.1.2	How to do it	13-4
13.1.3	Demonstration (1): Writing to a sequential file	13-5
13.1.4	Demonstration (2): Converting to packed decimal format	13-6
13.1.5	Demonstration (3): Writing complete records	13-8
13.1.6	Demonstration (4): Writing totals-only	13-8
13.1.7	Demonstration (5): Writing variable-length records	13-9
13.1.8	Demonstration (6): Writing from the database	13-10
13.2	Creating variable headings	13-11
13.2.1	What you can do	13-11
13.2.2	How to do it	13-11
13.2.3	Demonstration (1): Using the SORT parameter	13-11
13.2.4	Demonstration (2): Using the SORT/NOSORT parameters	13-12
13.3	Accessing non-database files defined in the data dictionary	13-14
13.3.1	What you can do	13-14
13.3.2	How to do it	13-14

CA-Culprit allows you to create and store nonprint reports and vary headings on printed reports. You can also access non-database files that are defined in the data dictionary.

13.1 Creating nonprint report output

13.1.1 What you can do

You can write and store CA-Culprit reports on conventional files on tape or disk.

These nonprint reports can be created from:

- Selected report information, including final report summary information
- Variable-length records
- Sequential files created from database information

The output from a CA-Culprit run can be stored on tape or disk as:

- A sequential (PS) file
- An indexed sequential (IS) file
- A card (CARD) file
- A VSAM file
- A data table

13.1.2 How to do it

Use:

- OUTPUT parameter options to set the output record size, block size, type of file, and the device to be used.
If the device is TAPE, the label type is also specified on the OUT parameter.
- Special formatting codes on type 5 and 6 parameters:
 - **FB** to convert an integer field value to binary output.
 - **FU** to convert a numeric field value to a single precision floating point value. (SZ= and DP= cannot be used with the FU format code.)
 - **FW** to convert a numeric field value to a double precision floating point value. (SZ= and DP= cannot be used with the FW format code.)
 - **FP** to convert a numeric output field value into a signed packed decimal number.
 - **FZ** to convert a numeric output field value into a signed zoned decimal number.
- No titles or headings.
- A plus sign (+) for a break code if you want total records for the output.
- Absolute column placement.

- The JCL label SYS020 to define the output file.

13.1.3 Demonstration (1): Writing to a sequential file

Objective: This example writes out selected data items listed on REC parameters to a 40-byte sequential file. A hexadecimal dump shows how the data is stored on the file.

Procedure

- An **OUTPUT** parameter defines a sequential file that will store detail information extracted from the type 5 detail lines:
 - Records are 40 bytes, fixed length.
 - DD=SYS020 specifies the name of the output file in execution JCL.
 In a VSE/ESA environment, the logical unit number of the device receiving the output file (DD=SYS020,20) must also be specified if the file type is not CARD.
- Type 5 lines specify **absolute column placement** for the data items written out. The column specifications become the byte positions of the items on the record.
- **Format codes** are used on numeric fields.

Complete code

```

col. 2
↓
IN 150 F 1500 PS(TAPE)
REC CUST-NO      1      5  3
REC GROSS-AMT   10     9  2  DP=2
REC NET          19     9  2  DP=2
REC ITEM-NO     28     4  1
REC TYPE         38     3
01OUT 40 4000 D PS DD=SYS020
01SORT CUST-NO
01510001 CUST-NO  FN           $Outputs leading zeros
01510010 GROSS-AMT FZ           $Formats as zoned decimals
01510019 NET        FZ
01510028 ITEM-NO   FM '999999999' $Requires ten digits
01510038 TYPE       IF TYPE = ('E00' TO 'T99') DROP

```

The hex dump output:

```
RECORD BUFFER DUMP
CHAR 00107661500024690 000222200 144523008A27
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 00107661500024690000222200144523008127
01....5...10....5...20....5...30....5...40

CHAR 0010766150005250 000047420 283356401B44
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 00107661500052500000474200283356401244
01....5...10....5...20....5...30....5...40

CHAR 00107661500015590 000132750 324836977B49
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 001076615000155900001327500324836977249
01....5...10....5...20....5...30....5...40

CHAR 00132696800239500 002195000 144523008A27
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 00132696800239500002195000144523008127
01....5...10....5...20....5...30....5...40

CHAR 00132696800009095 000081150 262973310B52
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 0013269680000909500000811500262973310252
01....5...10....5...20....5...30....5...40

CHAR 00588568300000000 000002200 424304486X12
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 005885683000000000000000022000424304486712
01....5...10....5...20....5...30....5...40

CHAR 00588568300010000 000085000 262973311B52
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 00588568300010000000085000262973311252
01....5...10....5...20....5...30....5...40

CHAR 00588568300044500 000400500 669011172C06
ZONE FFFFFFFFFFFFFFCFFFFFCFFFFFCFFFFFCFF
DIGIT 005885683000445000004005000669011172306
01....5...10....5...20....5...30....5...40
```

13.1.4 Demonstration (2): Converting to packed decimal format

Objective: This example writes the report in Demonstration (1) to a sequential file.

Procedure

- The record length is specified as 29 bytes to accommodate the signed packed decimal numeric fields.
- **FP** converts the numeric output into signed packed decimal format.

Complete code

```

col. 2
↓
IN 80 F 400
REC CUST-NO      1    9 2
REC GROSS-AMT   10   9 2 DP=2
REC NET          19   9 2 DP=2
REC ITEM-NO     28   10 2
REC TYPE         38    3
010 DIFF DP=2
OUT 29 2900 D PS DD=SYS020 $Less space needed for packed decimal
01SORT CUST-NO
01510001 CUST-NO  FP      $Stored as packed decimal
01510006 GROSS-AMT FP SZ=5
01510011 NET      FP SZ=5
01510016 ITEM-NO FP SZ=6
01510022 TYPE
01510025 DIFF      FP SZ=5
017010  IF TYPE = ('E00' TO 'T99') DROP
017015  GROSS-AMT - NET  DIFF

```

The hex dump output

```

RECORD BUFFER DUMP
CHAR      /*           A27
ZONE     017650049000220014208cff00270
DIGIT    0061c0260c0220c04530c1270040c
01...5...10....5...20....5.....

CHAR      /* & *       B44 *
ZONE     017650055000445023541cff00005
DIGIT    0061c0020c0072c08360c2440057c
01...5...10....5...20....5.....

CHAR      /*           @B49
ZONE     017650059000370034397cff00210
DIGIT    0061c0150c0125c02867c2490035c
01...5...10....5...20....5.....

CHAR      & & & B27
ZONE     012980035000350042851cff00000
DIGIT    0366c0080c0050c02230c2270030c
01...5...10....5...20....5.....

CHAR      A27
ZONE     012980290002900014208cff00000
DIGIT    0366c0350c0150c04530c1270200c
01...5...10....5...20....5.....

CHAR      B52
ZONE     012980099000810022730cff00080
DIGIT    0366c0005c0015c06931c2520090c
01...5...10....5...20....5.....

CHAR      %X12
ZONE     058630000000020044046eff00020
DIGIT    0858c0000c0020c02348c7120020d
01...5...10....5...20....5.....

CHAR      B52
ZONE     0586300000000800022731cff00100
DIGIT    0858c0100c0050c06931c2520050c
01...5...10....5...20....5.....

CHAR      &      C06 &
ZONE     058630000000050069112cff00450
DIGIT    0858c0450c0400c06017c3060040c
01...5...10....5...20....5.....

```

13.1.5 Demonstration (3): Writing complete records

Objective: This example writes input records to a file by defining the entire input record on one REC parameter.

Procedure

- **ENTIRE-RECORD** defines the whole input record.
- CUST-NO is defined separately for use on the SORT parameter.
- TYPE is defined separately for use in type 7 logic.

Complete code

```
col. 2
↓
IN 80 F 400
REC CUST-NO      1      9  2
REC TYPE        38      3
REC ENTIRE-RECORD 1    40 $Defines all input fields
01OUT 80 D
01SORT CUST-NO
01510001 ENTIRE-RECORD
017010 IF TYPE = ('E00' TO 'T99') DROP
```

Result

```
001076615000246900002222000144523008A27
0010766150000525000000474250283356401B44
001076615000155900001327500324836977849
001326968000038500000355000422283501B27
0013269680023950000021950000144523008A27
0013269680000909500000811500262973310B52
00588568300000000000000000022000424304486X12
00588568300010000000000850000262973311B52
0058856830004450000004005000669011172C06
```

13.1.6 Demonstration (4): Writing totals-only

Objective: This example writes a sorted totals-only report to a tape sequential file.

Procedure: A plus sign (+) inserted after CUST-NO indicates a control break, which prevents a blank record from being written each time the break occurs.

```

col. 2
↓
IN 80 F 400
REC CUST-NO      1     9 2
REC GROSS-AMT   10    9 2
REC NET          19    9 2
REC ITEM-NO     28   10 2
REC TYPE         38     3
010 DIFF
01OUT 40 4000 T PS(TAPE)
01SORT CUST-NO + $Control break on CUST-NO
01510000 GROSS-AMT
01510000 NET
01510000 DIFF
017010 IF TYPE = ('E00' TO 'T99') DROP
017015 GROSS-AMT - NET DIFF
01610001 CUST-NO FN
01610010 GROSS-AMT FZ
01610019 NET FZ
01610028 DIFF FZ SZ=11
018 IF LEVL EQ 2 DROP

```

13.1.7 Demonstration (5): Writing variable-length records

Objective: The following example reads in variable-length records and writes out variable-length records to a sequential file.

Procedure

- Input records are sorted by customer number and selectively written, depending to the value of TYPE.
- REC parameters define the complete input file in three segments, none of which exceeds 256 bytes:
 - The first four bytes of the record contain the RDW.
A new RDW is not required in this example. User module CULLUS33 can be used if a new record descriptor word (RDW) must be generated.
 - The sort key CUST-NO and the type 7 field TYPE are the only individual fields defined.
- Variable length output files are created by coding:
 - A sequential output file type **OUT PS**
 - JCL output specifications of:
 - RECFM=V or VB (SYS020)
 - An increased LRECL size (a multiple of 4 but not larger than 4096 bytes or the block size) in SYS006 of the JCL to accommodate the SORT parameters

Complete code

```
col. 2
↓
IN 640 V           $Variable-length file
REC CUST-NO      5      5   $Used as a sort key
REC TYPE        70      3   $Used in type 7 logic
REC PART-ONE    1     250
REC PART-TWO   251    250
REC PART-THREE 501    140
01SORT CUST-NO
01OUT 640 D PS
01510001 PART-ONE
01510251 PART-TWO
01510501 PART-THREE
017010 IF TYPE EQ ('E00' TO 'T99')  DROP
```

13.1.8 Demonstration (6): Writing from the database

Objective: This example creates a sequential data set, consisting of employee ID, employee name, and department name from data stored on the employee database.

Procedure: A conventional file is created from data that resides on the database by coding:

- The **DATABASE** parameter with the name of an alternate dictionary in a multiple dictionary environment.
- The **INPUT (IN) DB** parameter with the name of the subschema to be accessed.
- SYS020 in the JCL.
- Regular CA-Culprit parameters.

See Chapter 7, “Generating Reports From Database Files” on page 7-1 for information on coding the database parameters and options.

Complete code

```
col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01,EMPSCHM,100
PATHAA DEPARTMENT EMPLOYEE
01OUT 80 D PS DD=SYS020
01SORT DEPT-NAME-0410  EMP-NAME-0415
01510001 EMP-ID-0415  FZ
01510005 EMP-NAME-0415
01510030 DEPT-NAME-0410
```

13.2 Creating variable headings

13.2.1 What you can do

You can change the values that appear in report headings.

13.2.2 How to do it

Code:

- The field to be used as a heading on a **SORT** parameter
- **Type 4** parameters to reference the sort key
 - The current value will be obtained at each control break.

When a field to be used as a variable heading cannot be included on the **SORT** parameter, a **BRANCH** to **HEAD** instruction can be used (see the *CA-Culprit Reference*).

13.2.3 Demonstration (1): Using the **SORT** parameter

Objective: This report lists monthly account activity by branch. Each branch is listed on a separate page with the branch number as a subtitle.

Procedure

- A **SORT** parameter specifies:
 - **BRANCH** as a sort key
 - A control break (1), which starts a new report page
- A **type 4** parameter specifies
 - Absolute column placement
 - The word **BRANCH** to be printed
 - The value of **BRANCH** to be printed on the same line

Complete code

```

col. 2
↓
IN 80 F 400
REC ACCOUNT    1      5      'ACCOUNT'
REC TRANS-IND   6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT   7      11     2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC DATE        18     6      2      'DATE'
REC BRANCH      24     2
REC NAME        26     20      'NAME'
800UT
80SORT BRANCH 1 ACCOUNT           $Control break on BRANCH
803 ACCOUNT ACTIVITY BY BRANCH
80410001 'BRANCH'                $Print BRANCH
80410009 BRANCH                  $Prints the branch number
8051*020 ACCOUNT                 HR
8051*030 NAME                   HR
8051*050 TRANS-AMT              HR
8051*060 TRANS-IND              HR
8051*070 TRANS-DATE             FD

```

Results:

REPORT NO. 80		ACCOUNT ACTIVITY BY BRANCH	80	09/29/99	PAGE	1
ACCOUNT	NAME	AMOUNT OF TRANSACTION		DEPOSIT/ WITHDRAWAL	DATE	
15060	SHARON ARMSTRONG	10,099.01		D	01/15/91	
15060	SHARON ARMSTRONG	990.11		W	01/16/91	
15060	SHARON ARMSTRONG	100,990.11		D	01/20/91	
16070	ARTHUR LINK	1,080.04		D	01/01/91	
19235	GARY NOBLES	80.04		W	01/02/91	
21056	AMOS JOHNSON	1.02		W	01/20/91	
		113,240.33				

REPORT NO. 80		ACCOUNT ACTIVITY BY BRANCH	80	09/29/99	PAGE	2
ACCOUNT	NAME	AMOUNT OF TRANSACTION		DEPOSIT/ WITHDRAWAL	DATE	
23055	JACK JACKSON	500.00		W	01/02/91	
23055	JACK JACKSON	50,000.00		D	01/25/91	
23055	JACK JACKSON	30.00		D	01/26/91	
29557	IRWIN TRIMBLE	500,001.00		D	01/04/91	
		550,531.00				

13.2.4 Demonstration (2): Using the SORT/NOSORT parameters

Objective: This is the same report as that shown in Demonstration (1). In this example, the input file is presorted and an actual sort is not needed.

Procedure

- A sort with the NOSORT option occurs on BRANCH:
 - **SORT BRANCH 1** creates the control break, which makes the current value of BRANCH available to the type 4 reference.
 - **NOSORT** prevents the resorting of the already sorted file.

■ **Type 4** parameters:

- Place the word BRANCH on the subtitle line.
- Obtain and print the current value of BRANCH.

Complete code

```

col. 2
↓
IN 80 F 400
REC ACCOUNT    1      5      'ACCOUNT'
REC TRANS-IND   6      1      'DEPOSIT/' 'WITHDRAWAL'
REC TRANS-AMT   7      11     2 DP=2 'AMOUNT OF' 'TRANSACTION'
REC DATE        18     6      2      'DATE'
REC BRANCH      24     2      ''
REC NAME         26     20     'NAME'
800UT
80SORT BRANCH 1 NOSORT           $Control break without a sort
803 ACCOUNT ACTIVITY BY BRANCH
80410001 'BRANCH'               $Prints BRANCH
80410009 BRANCH                $Prints the branch number
8051*020 ACCOUNT              HR
8051*030 NAME                 HR
8051*050 TRANS-AMT             HR
8051*060 TRANS-IND             HR
8051*070 TRANS-DATE            FD   HR

```

Results:

REPORT NO. 80 BRANCH 32		ACCOUNT ACTIVITY BY BRANCH	80	09/29/99	PAGE	1
ACCOUNT	NAME	AMOUNT OF TRANSACTION		DEPOSIT/ WITHDRAWAL	DATE	
15060	SHARON ARMSTRONG	10,099.01		D	01/15/91	
15060	SHARON ARMSTRONG	990.11		W	01/16/91	
15060	SHARON ARMSTRONG	100,990.11		D	01/20/91	
16070	ARTHUR LINK	1,080.04		D	01/01/91	
19235	GARY NOBLES	80.04		W	01/02/91	
21056	AMOS JOHNSON	1.02		W	01/20/91	
		113,240.33				

REPORT NO. 80 BRANCH 35		ACCOUNT ACTIVITY BY BRANCH	80	09/29/99	PAGE	2
ACCOUNT	NAME	AMOUNT OF TRANSACTION		DEPOSIT/ WITHDRAWAL	DATE	
23055	JACK JACKSON	500.00		W	01/02/91	
23055	JACK JACKSON	50,000.00		D	01/25/91	
23055	JACK JACKSON	30.00		D	01/26/91	
29557	IRWIN TRIMBLE	500,001.00		D	01/04/91	
		550,531.00				

13.3 Accessing non-database files defined in the data dictionary

13.3.1 What you can do

When a file is defined in the data dictionary, the file attributes are automatically supplied to any CA-Culprit program referencing the file. The file attributes include record size, record type, block size, file type, device type, user module name, and label type.

You can use standard files that are defined in the data dictionary and revise already existing data dictionary definitions.

13.3.2 How to do it

To use the data dictionary definitions of a standard file, code:

- The **DATABASE** parameter with the DICTNAME= option if an alternate dictionary is used
- The **FN=** option on the INPUT parameter
- **REC** parameters if the fields in the file are to be renamed or redefined
- **HR** on edit parameters to print column headings stored in the data dictionary

To revise data dictionary definitions, code:

- A redefined **file definition** on the INPUT parameter:
Data dictionary file definition: record size is 100.
CA-Culprit code redefinition: IN 80
- A redefined **field definition** on a REC parameter with a field name identical to that used in the data dictionary:
Dictionary field definition: T-ACCOUNT with PIC X(5).
CA-Culprit code redefinition: REC T-ACCOUNT 1 10 'ACCOUNT'

Chapter 14. Additional CA-IDMS/DB Facilities

14.1	How to prepare for record retrieval	14-4
14.2	Retrieving partial paths	14-5
14.2.1	What you can do	14-5
14.2.2	How to do it	14-5
14.2.3	Demonstration	14-5
14.3	Retrieving record types by name	14-7
14.3.1	What you can do	14-7
14.3.2	How to do it	14-7
14.3.3	Demonstration	14-7
14.4	Retrieving record types by key	14-10
14.4.1	What you can do	14-10
14.4.2	How to do it	14-10
14.4.3	Demonstration (1): Accessing records by key	14-11
14.4.4	Demonstration (2): Accessing records with a key file	14-12
14.5	Retrieving stand-alone records	14-14
14.5.1	What you can do	14-14
14.5.2	How to do it	14-14
14.5.3	Demonstration	14-14
14.6	Retrieving all record occurrences	14-16
14.6.1	What you can do	14-16
14.6.2	How to do it	14-16
14.6.3	Demonstration	14-16
14.7	Testing for record occurrences	14-19
14.7.1	What you can do	14-19
14.7.2	How to do it	14-19
14.7.3	Demonstration	14-19
14.8	Accessing bill-of-materials structures	14-21
14.8.1	What you can do	14-21
14.8.2	How to do it	14-21
14.8.3	Demonstration	14-21
14.9	Accessing multiple-member sets	14-24
14.9.1	What you can do	14-24
14.9.2	How to do it	14-24
14.9.3	Demonstration	14-24

CA-Culprit allows you to access the database, retrieve all record types, and write reports using standard CA-Culprit code.

14.1 How to prepare for record retrieval

Code:

1. The data dictionary name if using an alternate dictionary
2. The **INPUT (IN) DB** parameter specifying:
 - The subschema name
 - The size of the input buffer if more than 1000 bytes is needed

The buffer must be large enough to hold:

 - Thirty-eight bytes of overhead
 - The sum of the length of each record type specified on the PATH parameter with the most records
 - Four additional bytes per record for the database key
3. The **path** or paths to navigate for retrieval of data

14.2 Retrieving partial paths

14.2.1 What you can do

You can retrieve a segment of a path (partial string) when the entire path cannot be completed.

14.2.2 How to do it

Specify a shorter (alternate) path.

A special type of alternate path ID, the null path identifier, can be used to indicate that an occurrence of the entry record was either not found or not selected.

►► The null path identifier is discussed in 14.4, “Retrieving record types by key” on page 14-10 later in this chapter.

Code:

1. A **PATH** parameter with the path identifier and the records to be accessed
2. An **alternate path identifier** placed one record after the last record of the shorter path

PATH01 EMPLOYEE EXPERTISE (02) SKILL \$Shorter path is EMPLOYEE

14.2.3 Demonstration

Objective: This report:

- Lists all employees, their insurance coverage, and hospital claims each time the complete path is navigated
- Lists all employees and their insurance coverage each time the alternate path is navigated
- Prints the path navigated for each report line

Procedure

- PATHAA EMPLOYEE COVERAGE HOSPITAL-CLAIM is the **primary path** for retrieval of employees having hospital claims.
- PATHAA EMPLOYEE COVERAGE HOSPITAL-CLAIM (BB) is an **alternate path** (PATHBB) for retrieval of insurance coverage codes.
- The value of **PATH-ID** is tested in the type 7 procedure logic for selecting one or both detail lines for printing.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
PATHAA EMPLOYEE COVERAGE HOSPITAL-CLAIM (BB)
013 EMPLOYEE LISTING
010 EMP-NAME '
010 INS-CODE '
0151*005 PATH-ID          HH 'PATH' 'USED'
0151*020 EMP-NAME-0415    HH 'EMPLOYEE' 'NAME'
0151*030 INS-PLAN-CODE-0400 HH 'INSURANCE' 'PLAN CODE'
0152*040+HOSPITAL-NAME-0430 HH 'HOSPITAL'
017010 IF EMP-NAME-0415 EQ EMP-NAME AND INS-PLAN-CODE-0400 EQ
* INS-CODE DROP                                     $Eliminate duplicate entries
017020 MOVE EMP-NAME-0415 TO EMP-NAME
017030 MOVE INS-PLAN-CODE-0400 TO INS-CODE
017100  IF PATH-ID EQ 'AA' 300
017200  TAKE 1
017300  TAKE (1 2)

```

Result:

REPORT NO. 01	EMPLOYEE LISTING		09/25/99 PAGE	1	
	PATH USED	EMPLOYEE NAME		INSURANCE PLAN CODE	
					HOSPITAL
	BB	KATHERINE O'HEARN		001	
	BB	KATHERINE O'HEARN		002	
	BB	ROBBY WILDER		001	
	BB	BURT LANCHESTER		004	
	BB	BURT LANCHESTER		003	
	AA	BURT LANCHESTER		002	
+					GENERAL HOSPITAL
	BB	BURT LANCHESTER		001	
	BB	VLADIMIR HEAROWITZ		001	
	BB	THEMIS PAPAZEUS		003	
	BB	HERBERT LIPSICH		001	
	BB	HENRIETTA HENDON		004	
	AA	HENRIETTA HENDON		002	
+					ANGELS OF MERCY HOSPITAL
	BB	RUPERT JENSON		001	
	BB	EDWARD HUTTON		003	
	BB	RICHARD MUNYON		002	
	BB	HARRY ARM		001	
	BB	HARRY ARM		002	
	BB	MICHAEL ANGELO		001	
	BB	LAURA PENMAN		003	
	BB	JAMES JACOBI		004	
	BB	ROBIN GARDNER		002	
	BB	HERBERT CRANE		004	
	BB	HERBERT CRANE		003	
	BB	HERBERT CRANE		001	
	BB	PERCY GRANGER		003	
	BB	JANE DOUGH		003	
	BB	LARRY LITERATA		003	
	BB	GEORGE FONRAD		004	
	BB	DANIEL MOON		002	

14.3 Retrieving record types by name

14.3.1 What you can do

You can specify selection criteria for database records named on one or more PATH parameters.

14.3.2 How to do it

Code **SELECT** or **BYPASS** parameters after the PATH parameters:

- The **IN PATH** clause specifies selection criteria for a specific path.
- The **BUFFER** specifies selection criteria for every path. (**BUFFER** is discussed further under 14.9, “Accessing multiple-member sets” on page 14-24 later in this chapter.)

14.3.3 Demonstration

Objective: This example creates two reports:

- Report 01 lists all employees for each department.
- Report 02 lists the skills of employees who work in departments 4000 and 3200.

Procedure

- Two paths are used to access data:
 - for departments 4000 and 3200.
- A **SELECT** statement retrieves path BB when the department id is 4000 or 3200.
- **Type 7** procedure logic selects path AA for Report 01 and path BB for Report 02.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
PATHAA DEPARTMENT EMPLOYEE
PATHBB DEPARTMENT EMPLOYEE EXPERTISE SKILL
SELECT DEPARTMENT IN PATH BB WHEN DEPT-ID-0410 EQ (4000 3200)
013 EMPLOYEES BY DEPARTMENT
0151*020 DEPT-NAME-0410    HH 'DEPARTMENT'
0151*030 EMP-NAME-0415    HH 'EMPLOYEE'
017010 IF PATH-ID NE 'AA' DROP
023 EMPLOYEE SKILL LEVELS FOR DEPTS 4000 AND 3200
0251*010 DEPT-ID-0410   FN HH 'DEPT ID'
0251*020 DEPT-NAME-0410   HH 'DEPARTMENT'
0251*030 EMP-NAME-0415   HH 'EMPLOYEE'
0251*040 START-YEAR-0415  HH 'START YEAR'
0251*050 SKILL-NAME-0455  HH 'SKILL'
027010 IF PATH-ID NE 'BB' DROP

```

Result:

REPORT NO. 01	EMPLOYEES BY DEPARTMENT	09/24/99 PAGE	1	EMPLOYEE
	DEPARTMENT			
	ACCOUNTING AND PAYROLL		JUNE	BLOOMER
	ACCOUNTING AND PAYROLL		EDWARD	HUTTON
	ACCOUNTING AND PAYROLL		RUPERT	JENSON
	ACCOUNTING AND PAYROLL		MARIANNE	KIMBALL
	ACCOUNTING AND PAYROLL		DORIS	KING
	ACCOUNTING AND PAYROLL		BRIAN	NICEMAN
	COMPUTER OPERATIONS		HERBERT	CRANE
	COMPUTER OPERATIONS		JANE	FERNDALE
	COMPUTER OPERATIONS		GEORGE	FONRAD
	COMPUTER OPERATIONS		ROBIN	GARDNER
	COMPUTER OPERATIONS		DOUGLAS	KAHALLY
	COMPUTER OPERATIONS		TERENCE	KLWELLEN
	COMPUTER OPERATIONS		SANDY	KRAAMER
	COMPUTER OPERATIONS		HERBERT	LIPSICH
	COMPUTER OPERATIONS		NANCY	TERNER
	BLUE SKIES		BETH	CLOUD
	BLUE SKIES		ALAN	DONOVAN
	BLUE SKIES		DANIEL	MOON
	BRAINSTORMING		ROY	ANDALE
	BRAINSTORMING		HARRY	ARM

REPORT NO. 02 DEPT ID	EMPLOYEE SKILL LEVELS FOR DEPTS 4000 AND 3200 DEPARTMENT	09/24/99 PAGE EMPLOYEE	1	START YEAR	SKILL
3200	COMPUTER OPERATIONS	HERBERT CRANE	77	IDMSDC	
3200	COMPUTER OPERATIONS	HERBERT CRANE	77	IBM	
3200	COMPUTER OPERATIONS	HERBERT CRANE	77	ASSEMBLER	
3200	COMPUTER OPERATIONS	HERBERT CRANE	77	DEC	
3200	COMPUTER OPERATIONS	HERBERT CRANE	77	RPGII	
3200	COMPUTER OPERATIONS	JANE FERNDALE	79	IDMSDC	
3200	COMPUTER OPERATIONS	JANE FERNDALE	79	IBM	
3200	COMPUTER OPERATIONS	GEORGE FONRAD	80	DATA ENTRY	
3200	COMPUTER OPERATIONS	GEORGE FONRAD	80	FILING	
3200	COMPUTER OPERATIONS	ROBIN GARDNER	81	DATA ENTRY	
3200	COMPUTER OPERATIONS	DOUGLAS KAHALLY	79	KEYPUNCH	
3200	COMPUTER OPERATIONS	DOUGLAS KAHALLY	79	FILING	
3200	COMPUTER OPERATIONS	DOUGLAS KAHALLY	79	TYPING	
3200	COMPUTER OPERATIONS	DOUGLAS KAHALLY	79	PHONE	
3200	COMPUTER OPERATIONS	TERENCE KLWELLEN	78	COBOL	
3200	COMPUTER OPERATIONS	TERENCE KLWELLEN	78	ASSEMBLER	
3200	COMPUTER OPERATIONS	SANDY KRAAMER	81	KEYPUNCH	
3200	COMPUTER OPERATIONS	SANDY KRAAMER	81	TYPING	
3200	COMPUTER OPERATIONS	HERBERT LIPSICH	81	IBM	
3200	COMPUTER OPERATIONS	NANCY TERNER	82	FILING	
3200	COMPUTER OPERATIONS	NANCY TERNER	82	PHONE	
3200	COMPUTER OPERATIONS	NANCY TERNER	82	TYPING	
4000	PUBLIC RELATIONS	MICHAEL ANGELO	79	RAINBOW	
4000	PUBLIC RELATIONS	MICHAEL ANGELO	79	ART DESIGN	
4000	PUBLIC RELATIONS	MICHAEL ANGELO	79	PASTEUP	
4000	PUBLIC RELATIONS	MONTE BANK	78	PUB REL MGT	
4000	PUBLIC RELATIONS	CHARLES BOWER	77	PHOTO COLOR	
4000	PUBLIC RELATIONS	CHARLES BOWER	77	PHOTO B/W	
4000	PUBLIC RELATIONS	CAROL McDougall	80	CUT/PASTE	
4000	PUBLIC RELATIONS	CAROL McDougall	80	LAYOUT	

14.4 Retrieving record types by key

14.4.1 What you can do

You can retrieve the entry record on a PATH parameter directly by using a specific key value or a range of key values. Key values allow you to avoid sweeping a database area for a particular record.

The following types of key values can be specified:

- **CALC-key** values when entry records are stored in the CALC location mode
- **Index-key** values when entry records are indexed
- **Db-key** values for any location mode (VIA, CALC, or DIRECT) unless the entry records are indexed or are logical records
- Any **logical record fields** when entry records are logical records

14.4.2 How to do it

Retrieve specific occurrences of the PATH entry record by using:

- **KEY** parameters to name key fields and key values

To code a KEY parameter, supply the following information:

1. The KEY keyword (beginning in column 2)
2. The key field name
3. A key value, a list of values, or a range of values

- **A KEYFILE** parameter to access a sequential file containing the key values

To code a KEYFILE parameter, supply the following information:

1. The KEYFILE keyword (beginning in column 2)
2. The size, in bytes, of the key file record
3. The KF= keyword expression, followed by the starting position of the key field in the file and its length in bytes
4. Other options:

- The **FN=** keyword expression to name a file defined to the data dictionary:

```
PATH01 DEPT-EMP-LR WHERE EMP-NAME EQ EMP-KEY  
KEYFILE FN=EMPLOYEE-KEYS
```

- The **LRFNAME=** keyword expression to name a logical record when the entry record is a logical record and you are supplying key information to the WHERE clause on the PATH parameter:

```
PATH01 DEPT-EMP-LR WHERE EMP-LAST-NAME EQ KEY-VALUE  
KEYFILE CARD KF=10 15 LRFNAME=EMP-LAST-NAME
```

You can also use the KEYFILE parameter to:

- Define fields in a sequential key file that are not defined to the data dictionary:

```
REC EMP-ID (KEYFILE) 20 4
```

- Define selection criteria applicable to key-file records:

```
KEYFILE 80 KF=1 4  
SELECT DEPT-LOCATION EQ 'DEDHAMPLAC'
```

14.4.3 Demonstration (1): Accessing records by key

Objective: This example uses a KEY parameter to list the job ids, job titles, and job openings for three selected departments.

Procedure

- A KEY parameter is used to directly access DEPARTMENT records with ids of 0100, 3200, and 5100.
- The **key field** on the KEY parameter is DEPT-ID-0410 (the CALC-key value for the DEPARTMENT record).

Complete code

```
col. 2  
↓  
DATABASE DICTNAME=DOCUDICT  
IN DB SS=EMPSS01  
PATH01 DEPARTMENT EMPLOYEE EMPOSITION JOB  
KEY DEPT-ID-0410 (0100 3200 5100)  
01OUT D  
013 JOB OPENINGS  
0151*010 DEPT-ID-0410 FN HH 'DEPARTMENT'  
0151*020 JOB-ID-0440 FN HH 'JOB ID'  
0151*030 TITLE-0440 HH 'JOB TITLE'  
0151*040 NUMBER-OPEN-0440 HH 'JOB' 'OPENINGS'
```

Result:

REPORT NO. 01	JOB OPENINGS	09/24/99 PAGE	1	
DEPARTMENT	JOB ID	JOB TITLE		JOB OPENINGS
0100	9001	PRESIDENT		
0100	9007	DIR WEATHER		
0100	5001	MGR BRAINSTORMING		
0100	9003	DIR OPERATIONS		
0100	3001	MGR INTERNL SOFTWARE		
0100	9005	DIR CORP CONFUSION		
3200	3003	MGR COMPUTER OPS		
3200	3003	MGR COMPUTER OPS		
3200	3003	MGR COMPUTER OPS		
3200	3029	COMPUTER OPERATOR		
3200	3051	DATA ENTRY CLERK	1	
3200	3051	DATA ENTRY CLERK	1	
3200	3029	COMPUTER OPERATOR		
3200	3023	SYSTEMS PROGRAMMER		
3200	3051	DATA ENTRY CLERK	1	
3200	3029	COMPUTER OPERATOR		
3200	3051	DATA ENTRY CLERK	1	
5100	5025	SNOWBLOWER		
5100	5029	STURM/DRANG ADMIN		
5100	5027	KEEPER OF THE WINDS		
5100	5023	RAINDANCE CONSULTANT		
5100	5021	RAINMAKER		
5100	5021	RAINMAKER		
5100	5021	RAINMAKER		
5100	5001	MGR BRAINSTORMING		
5100	5001	MGR BRAINSTORMING		
5100	5025	SNOWBLOWER		
5100	5029	STURM/DRANG ADMIN		

14.4.4 Demonstration (2): Accessing records with a key file

Objective: This report lists all employees associated with specific departments located at Dedham Place and University Avenue. Department IDs and building locations are stored in a key file.

Procedure

- A **KEYFILE** parameter defines a key file with records 80 bytes long. The key field is a CALC-key value that begins in position 1 and is four bytes long.
- A user-supplied **REC** parameter identifies DEPT-LOCATION as an alphanumeric field. DEPT-LOCATION starts in position 5 of the key file record for a length of ten bytes.
- A **SELECT** parameter uses DEPT-LOCATION to select only those records in the key file that specify Lost Brook or University Avenue.
- A **SORT** parameter sorts the output by building location. Building location appears as a heading on each page.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
PATHAA DEPARTMENT EMPLOYEE
KEYFILE 80 KF=1 4
REC DEPT-LOCATION(KEYFILE) 5 10
SELECT DEPT-LOCATION EQ ('DEDHAMPLAC' 'UNIVERSITY')
01SORT DEPT-LOCATION,1
01410005 'BUILDING LOCATION'
01410025 DEPT-LOCATION
0151*001 DEPT-ID-0410    FM '9999' HH 'DEPT ID'
0151*003 EMP-ID-0415    FM '9999' HH 'EMPLOYEE' 'ID'
0151*005 EMP-NAME-0415           HH 'EMPLOYEE' 'NAME'

```

The standard file used to store the keys is specified in the JCL:

```
//SYS002 DD DSN=user.keyfile,DISP=SHR
```

Result:

BUILDING LOCATION	DEDHAMPLAC	EMPLOYEE	EMPLOYEE
		ID	NAME
5200		0479	TERRY CLOTH
5200		0329	PHINEAS FINN
5200		0469	JOE KASPAR
5200		0355	MARK TIME
5200		0349	ROGER WILCO
5300		0371	BETH CLOUD
5300		0366	ALAN DONOVAN
5300		0321	DANIEL MOON
3100		0024	JANE DOUGH
3100		0029	JAMES GALLWAY
3100		0003	JENNIFER GARFIELD
3100		0028	PERCY GRANGER
3100		0027	VLADIMIR HEAROWITZ
3100		0020	JAMES JACOBI
3100		0019	JULIE JENSEN
3100		0035	LARRY LITERATA
3100		0023	KATHERINE O'HEARN
3100		0021	RALPH TYRO

BUILDING LOCATION	UNIVERSITY	EMPLOYEE	EMPLOYEE
		ID	NAME
5100		0466	ROY ANDALE
5100		0457	HARRY ARM
5100		0467	C. BREEZE
5100		0334	CAROLYN CROW
5100		0301	BURT LANCHESTER
5100		0015	RENE MAKER
5100		0341	RICHARD MUNYON
5100		0458	RICHARD WAGNER
3200		0004	HERBERT CRANE
3200		0032	JANE FERNDALE
3200		0045	GEORGE FONRAD
3200		0053	ROBIN GARDNER
3200		0049	DOUGLAS KAHALLY
3200		0016	TERENCE KLWELLEN
3200		0074	SANDY KRAMMER
3200		0031	HERBERT LIPSICH
3200		0048	NANCY TERNER

14.5 Retrieving stand-alone records

14.5.1 What you can do

You can retrieve record occurrences that do not participate in defined paths, as well as path-related records, by calling or branching to the DB-EXIT facility in type 7 logic.

14.5.2 How to do it

To use the DB-EXIT facility, code:

1. A **dummy PATH parameter (PATH--)** with:
 - One or more database records
 - One or more logical records
2. A **STOP-RUN** command on a type 7 parameter if the only paths coded for the run are dummy paths.
3. A **CALL** or **BRANCH** to the DB-EXIT facility on a type 7 parameter
A branch to DB-EXIT is preceded by a series of logic statements that move values to the DB-EXIT arguments.
4. **IDMS-STATUS** and **LR-STATUS** value checking in procedure logic statements after each DB-EXIT call.

14.5.3 Demonstration

Objective: This report lists the start date of five employees selected from the database EMPLOYEE record type. The selection occurs by means of the DB-exit facility.

Procedure: The code for this report includes:

- A dummy path, **PATH--**, allocates space in the input buffer for the retrieved records.
- The work field **IDS.5** contains employee ID values.
- A call to the **DB-EXIT** facility has the following selection criteria arguments:
 - **Argument 1 ('CALC')** generates an OBTAIN CALC EMPLOYEE retrieval command.
 - **Argument 2 ('EMPLOYEE ')** specifies the EMPLOYEE record. The trailing blank is required (the record name is less than 16 characters).
 - **Argument 3 (EMP-ID-0415)** specifies the name of the CALC-key field for the EMPLOYEE record.
 - **Argument 4 ('IDS.INDEX')** specifies an occurrence of the work field IDS.5 that has the value of the EMP-ID-0415 field being selected.

- **Argument 5** specifies the length of the EMP-ID-0415 field.
- **IDMS-STATUS NE '0000' 400** checks the IDMS Status for values that indicate error conditions or records not retrieved. If the condition is true, CA-Culprit generates a buffer dump and stops the run.
- **STOP-RUN** terminates the run.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01 $RPT707
PATH--EMPLOYEE
01OUT 80 D
010 IDS.5 '0302' '0048' '0054' '0301' '0001' $Employee ID values
010 INDEX 1
010 DUMP
010 MESSAGE 'EMPLOYEE NOT FOUND'
0151*005 IDMS-STATUS          HH 'IDMS-STATUS' 'FIELD'
0151*010 EMP-ID-0415        FM '9999' HH 'EMPLOYEE' 'ID'
0151*020 EMP-NAME-0415       HH 'EMPLOYEE' 'NAME'
0151*030 START-YEAR-0415     HH 'START' 'YEAR'
0152*005 IDMS-STATUS
0152*010 IDS.INDEX
0152*020 MESSAGE
017100 CALL DB-EXIT ('CALC' 'EMPLOYEE ' EMP-ID-0415 IDS.INDEX 4)
017 IF IDMS-STATUS EQ '0326' 200           $Employee not found
017 IDMS-STATUS NE '0000' 400
017 RELS 1
017 B 300
017200 MOVE 'EMPLOYEE NOT FOUND' TO MESSAGE
017 RELS 2
017300 INDEX + 1 INDEX
017 IF INDEX LE 5 100
017 STOP-RUN
017400 DUMP / DUMP DUMP
017 STOP-RUN                                $Forces a buffer dump

```

Result

IDMS-STATUS FIELD	EMPLOYEE ID	EMPLOYEE NAME	START YEAR
0326	0302	EMPLOYEE NOT FOUND	
0000	0048	NANCY TERNER	82
0326	0054	EMPLOYEE NOT FOUND	
0000	0301	BURT LANCHESTER	75
0000	0001	JOHN RUPEE	75

14.6 Retrieving all record occurrences

14.6.1 What you can do

You can retrieve all the occurrences of a database record type by navigating (walking) through each set of occurrences with the DB-EXIT facility.

14.6.2 How to do it

Code the arguments of the DB-EXIT facility with:

- **ARG1** — A DB-EXIT argument that generates an CA-IDMS/DB retrieval command
- **ARG2** — The field to be obtained
- **ARG3** — The area or set to be accessed

14.6.3 Demonstration

Objective: This example uses the DB-EXIT facility to retrieve all employees working in each office.

Procedure: The DB-EXIT facility is used to walk the OFFICE-EMPLOYEE set until the end of the set is reached:

- **CALL DB-EXIT ('FIRST-AREA' 'OFFICE ' 'ORG-DEMO-REGION ')** retrieves the first OFFICE record in the ORG-DEMO-REGION area.
- **CALL DB-EXIT ('FIRST' 'EMPLOYEE ' 'OFFICE-EMPLOYEE ')** retrieves the first EMPLOYEE record in the OFFICE-EMPLOYEE set if the first office retrieval was successful.
- **CALL DB-EXIT ('NEXT' 'EMPLOYEE ' 'OFFICE-EMPLOYEE ')** retrieves the next EMPLOYEE record until the end of the set is reached.
- **CALL DB-EXIT ('NEXT-AREA' 'OFFICE ' 'ORG-DEMO-REGION ')** retrieves the next OFFICE record and, if successful, returns to retrieve the first EMPLOYEE record for that office.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
PATH-- OFFICE EMPLOYEE
010 MSG 'ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQ'
010 JUNK
0151*005 IDMS-STATUS           HH 'IDMS-STATUS' 'FIELD'
0151*010 OFFICE-CODE-0450     HH 'OFFICE CODE'
0151*020 EMP-NAME-0415        HH 'EMPLOYEE' 'NAME'
0152*005 IDMS-STATUS
0152*010 MSG
010UT D
01SORT OFFICE-CODE-0450 0 EMP-NAME-0415
017 CALL DB-EXIT ('FIRST-AREA' 'OFFICE ' 'ORG-DEMO-REGION ')
017 IF IDMS-STATUS EQ '0000' 050
017 MOVE 'ERROR IN RETRIEVAL OF FIRST OFFICE' MSG
017 RELS 2
017 PERFORM 800
017050 CALL DB-EXIT ('FIRST' 'EMPLOYEE ' 'OFFICE-EMPLOYEE ')
017 B 150
017100 CALL DB-EXIT ('NEXT' 'EMPLOYEE ' 'OFFICE-EMPLOYEE ')
017150 IF IDMS-STATUS EQ '0000' 200          $Test for record found
017 IF IDMS-STATUS EQ '0307' 300          $Test for end of set
017 MOVE 'ERROR IN RETRIEVAL OF EMPLOYEE' MSG
017 RELS 2
017 PERFORM 800
017200 RELS 1
017 B 100
017300 CALL DB-EXIT ('NEXT-AREA' 'OFFICE ' 'ORG-DEMO-REGION ')
017 IF IDMS-STATUS EQ '0000' 050
017 IF IDMS-STATUS EQ '0307' STOP
017 MOVE 'ERROR IN RETRIEVAL OF NEXT OFFICE' MSG
017 RELS 2
017 PERFORM 800
017800 JUNK / 0 JUNK                 $Forces a buffer dump
017 STOP

```

Result:

14.6 Retrieving all record occurrences

IDMS-STATUS FIELD	OFFICE CODE	EMPLOYEE NAME	
0000	001	BETSY	ZEDI
0000	001	HERBERT	CRANE
0000	001	HERBERT	LIPSICH
0000	001	JAMES	GALLWAY
0000	001	JAMES	JACOBI
0000	002	DOUGLAS	KAHALLY
0000	002	SANDY	KRAAMER
0000	002	TOM	FITZHUGH
0000	005	ALAN	DONOVAN
0000	005	BETH	CLOUD
0000	005	BURT	LANCHESTER
0000	005	CAROLYN	CROW
0000	005	DANIEL	MOON
0000	005	RENE	MAKER
0000	005	TERRY	CLOTH
0000	008	C.	BREEZE
0000	008	HARRY	ARM
0000	008	JOE	KASPAR
0000	008	MARK	TIME
0000	008	RICHARD	MUNYON
0000	008	RICHARD	WAGNER
0000	008	ROGER	WILCO
0000	008	ROY	ANDALE
0000	008	THEMIS	PAPAZEUS

14.7 Testing for record occurrences

14.7.1 What you can do

You can identify record occurrences that are not found in the database.

14.7.2 How to do it

1. Create a **null path** by placing the path identifier immediately after the entry record.
2. Test the **current value of PATH-ID** in type 7 logic.

14.7.3 Demonstration

Objective: This report lists selected employees and their start year. If an employee record is not found, the message EMPLOYEE NOT FOUND is returned.

Procedure

- A primary path, **PATHOK**, returns the employee name, ID, and date of hire.
- A null path, **NO**, places NO in the input buffer if the employee ID is not found. selects one of two possible print lines.

Complete code

```

col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01
REC JOB-START(EMPOSITION) 1 6 2 'START' 'DATE'
PATHOK EMPLOYEE(NO) EMPOSITION JOB
KEY EMP-ID-0415 (0003 0023 0100 0301 3200 5100)
01OUT 80 D
013 SELECTED EMPLOYEE DETAIL LIST
010 WK-ID
0151*0100EMP-NAME-0415           HH 'EMPLOYEE NAME'
0151*020 EMP-ID-0415 FM '9999'   HH 'EMPLOYEE' 'ID'
0151*030 JOB-START    FD        HR
0151*040 TITLE-0440             HH 'JOB' 'TITLE'
0152*010 'EMPLOYEE NOT FOUND'   '
0152*020 EMP-ID-0415 FM '9999'
017005 IF PATH-ID EQ 'OK'  100
017010 IF PATH-ID NE 'NO'  DROP
017020 TAKE 2
017100 IF EMP-ID-0415 EQ WK-ID  DROP
017110 MOVE EMP-ID-0415 TO WK-ID
017120 TAKE 1

```

Result

14.7 Testing for record occurrences

REPORT NO. 01	SELECTED EMPLOYEE	DETAIL LIST	09/24/99	PAGE	1
EMPLOYEE NAME	EMPLOYEE ID	START DATE	JOB	TITLE	
JENNIFER GARFIELD	0003	82/01/01	MGR INTERNL SOFTWARE		
KATHERINE O'HEARN	0023	79/05/05	PROGRAMMER/ANALYST		
EDWARD HUTTON	0100	77/09/07	FINANCIAL ANALYST		
BURT LANCHESTER	0301	80/02/03	RAINMAKER		
EMPLOYEE NOT FOUND	3200				
EMPLOYEE NOT FOUND	5100				

14.8 Accessing bill-of-materials structures

14.8.1 What you can do

You can access database information stored as a bill-of-materials data structure.

The bill-of-materials structure, which can be thought of as a series of nested set relationships, allows a path to be established between one record type and another of the same type. Different set relationships that exist between the record types are used.

14.8.2 How to do it

Code:

1. **PATH** parameters using set names that define the level of information to be retrieved
2. **Field names** using level numbers to specify the level of information to be acted upon

14.8.3 Demonstration

Objective: This report lists:

- The managers of the company (first level)
- The supervisors reporting to each manager (second level)
- The workers reporting to each supervisor (third level)

Procedure

- **PATHAA** defines a three-level bill-of-materials relationship:
 1. EMPLOYEE STRUCTURE(MANAGES)
 2. EMPLOYEE(REPORTS-TO) STRUCTURE(MANAGES)
 3. EMPLOYEE(REPORTS-TO)
- **Type 5** and **type 7** parameters specify the level from which the names are extracted:
 - EMP-NAME-0415 (1) prints manager's names.
 - EMP-NAME-0415 (2) prints supervisor's names.
 - EMP-NAME-0415 (3) prints worker's names.

Complete code

```
col. 2
↓
DATABASE DICTNAME=DOCUDICT
IN DB SS=EMPSS01 $RPT708
PATHAA EMPLOYEE STRUCTURE(MANAGES) EMPLOYEE(REPORTS-TO)
*STRUCTURE(MANAGES) EMPLOYEE(REPORTS-TO)
01OUT 120
010 MGR '
010 SUP '
010 EMP '
0141*010 'MANAGER'
0141*020 'SUPERVISOR'
0141*030 'WORKER'
0142*001 '
0151*010 EMP-NAME-0415 (1)
0152*020 EMP-NAME-0415 (2)
0153*030 EMP-NAME-0415 (3)
017010 IF EMP-NAME-0415 (1) NE MGR 100
017020 IF EMP-NAME-0415 (2) NE SUP 200
017030 IF EMP-NAME-0415 (3) NE EMP 300
017    DROP
017100 RELS 1
017200 RELS 2
017300 RELS 3
017    MOVE EMP-NAME-0415 (1) TO MGR
017    MOVE EMP-NAME-0415 (2) TO SUP
017    MOVE EMP-NAME-0415 (3) TO EMP
```

Result:

MANAGER		SUPERVISOR		WORKER	
ROBBY	WILDER	THEMIS	PAPAZEUS	ROGER DANIEL RENE	WILCO MOON MAKER
		MONTE	BANK	BETSY JOCK LAURA CAROL MICHAEL CHARLES	ZEDI JACKSON PENMAN MCDougall ANGELO BOWER
THEMIS	PAPAZEUS	ROGER	WILCO	TERRY JOE MARK PHINEAS	CLOTH KASPAR TIME FINN
		DANIEL	MOON	BETH ALAN	CLOUD DONOVAN
		RENE	MAKER	C. ROY RICHARD HARRY RICHARD CAROLYN BURT	BREEZE ANDALE WAGNER ARM MUNYON CROW LANCHESTER
HENRIETTA HENDON		ROGER	WILCO	TERRY JOE MARK PHINEAS	CLOTH KASPAR TIME FINN
		DANIEL	MOON	BETH ALAN	CLOUD DONOVAN
		ELEANOR	PEOPLES	MADELINE TOM CYNTHIA	ORGRATZI FITZHUGH JOHNSON
	JENNIFER GARFIELD			LARRY JAMES PERCY VLADIMIR JANE KATHERINE RALPH JAMES JULIE	LITERATA GALLWAY GRANGER HEAROWITZ DOUGH O'HEARN TYRO JACOBI JENSEN

14.9 Accessing multiple-member sets

14.9.1 What you can do

You can access records in a set that has one owner record and two or more member records.

14.9.2 How to do it

Code:

1. A PATH parameter placing the **ALL-MEMBERS** option after the name of the owner record
2. An optional SELECT/BYPASS BUFFER parameter *after the INPUT, KEYFILE, and PATH parameters* that includes the REC-NAME keyword

14.9.3 Demonstration

Objective: This report lists employee hospital and dental claims. The HOSPITAL-CLAIM and DENTAL-CLAIM records are owned by the COVERAGE record.

Procedure

- The **INPUT** parameter specifies an input buffer size of 4000 bytes to accommodate the five record types (EMPLOYEE, COVERAGE, HOSPITAL-CLAIM, NON-HOSP-CLAIM, and DENTAL-CLAIM), overhead, and DB-keys that are read into the buffer.

The NON-HOSPITAL-CLAIM record, while not used for this report, must be included in the buffer size.

- The **ALL-MEMBERS** option on the PATH parameter specifies access to each occurrence of the records in the multiple-member set COVERAGE-CLAIMS.
- The **SELECT BUFFER** parameter with **REC-NAME** specifies access to two of the three records of the COVERAGE-CLAIMS set.
- **REC** parameters redefine the claim date fields as numeric to allow for automatic date formatting.

Field redefinition is based upon the starting point in the record, as opposed to the position in the input buffer, by including the record type name immediately after the field name.

Complete code

```

col. 2
↓
IN 4000 DB SS=EMPSS01
PATH01 EMPLOYEE COVERAGE ALL-MEMBERS (COVERAGE-CLAIMS)
SELECT BUFFER WHEN REC-NAME EQ ('HOSPITAL-CLAIM' 'DENTAL-CLAIM')
REC CLAIM-DATE-0430(HOSPITAL-CLAIM) 1 6 2
REC CLAIM-DATE-0405(DENTAL-CLAIM) 1 6 2
01OUT D
0141*010 'PATIENT'
0141*020 'HOSPITAL CLAIMS'
0141*030 'DENTAL CLAIMS'
0141*040 'EMPLOYEE'
0142*001 ' '
0151*010 PATIENT-NAME-0430
0151*020 CLAIM-DATE-0430 FD
0151*040 EMP-NAME-0415
0152*010 PATIENT-NAME-0405
0152*030 CLAIM-DATE-0405 FD
0152*040 EMP-NAME-0415
017010 IF REC-NAME EQ 'HOSPITAL-CLAIM' 300
017      TAKE 2
017300 TAKE 1

```

Result:

	PATIENT	HOSPITAL CLAIMS	DENTAL CLAIMS	EMPLOYEE
KATE	HEPBURN	79/09/18		BURT LANCHESTER
HELOISE	HENDON		77/05/23	HENRIETTA HENDON
HATFIELD	HENDON	82/11/07		HENRIETTA HENDON
JESSICA	CRANE		80/10/04	HERBERT CRANE

Using SQL Tables

Chapter 15. Retrieving Data With SQL

15.1	Getting the exact column names for an SQL table	15-4
15.2	Using the AS clause to rename SQL columns	15-5
15.3	Retrieving floating point data from SQL columns	15-6
15.4	Displaying an SQL column with data type BINARY	15-7
15.5	How to test and display null values	15-9
15.6	Handling null values at total time	15-10
15.7	Understanding CA-Culprit decimal point handling	15-12
15.8	Interpreting common error messages in SQL retrieval	15-13

This chapter provides many useful procedures and CA-Culprit coding techniques to help you write complex reports using SQL and CA-Culprit.

15.1 Getting the exact column names for an SQL table

If you do not know the exact column names in the INV STOCKS table, try running the following simple CA-Culprit report.

```
col. 2
↓
PRO EX=NO
IN DB(Q) DICTIONARY=TSTDICT SCHEMA=INV
SQL SELECT * FROM STOCKS;
0151*001 ''
```

The EX=NO tells CA-Culprit to simply compile the report. No data extraction will be performed.

SQL SELECT * FROM STOCKS tells CA-Culprit to describe all of the columns and null indicators for the STOCKS table. The generated REC cards will appear on the INPUT PARAMETER LISTING.

Generated REC cards

REC	START	SIZE	TYPE	DP	FIELD-NAME
REC	0100	00005	3	3	CLOSE_PRICE
REC	0105	00004	1		CLOSE_PRICE_NULL_IND
REC	0069	00005			COMPANY_ID
REC	0074	00004	1		COMPANY_ID_NULL_IND
REC	0049	00020			COMPANY_NAME
REC	0078	00010			TRADING_DATE
REC	0088	00004	1		TRADING_DATE_NULL_IND
REC	0092	00004	1		VOLUME
REC	0096	00004	1		VOLUME_NULL_IND

From this report, you can deduce that the STOCKS table was created using the following SQL command:

SQL command

```
CREATE TABLE INV STOCKS
(COMPANY_NAME CHAR(20) NOT NULL,
 COMPANY_ID CHAR(5),
 TRADING_DATE DATE,
 VOLUME INTEGER,
 CLOSE_PRICE DECIMAL(9,3));
```

15.2 Using the AS clause to rename SQL columns

Long SQL column names can present a problem to CA-Culprit, since you can create a null indicator name by appending **_NULL_IND** to the end of the previously generated REC card name. You can avoid potential errors by using an AS clause to assign an alias to each SQL column.

AS clause example

```
col. 2
↓
INPUT DB(Q)    DICTIONARY=TSTDICT    SCHEMA=INV
SQL SELECT "EMP-ID-0415"      AS "ID",
*          "EMP-FIRST-NAME-0415"   AS "FNAME",
*          "EMP-LAST-NAME-0415"   AS "LNAME"
*     FROM EMPLOYEE
*    WHERE "EMP-ID-0415" = 2000
*    ORDER BY ID;
0151*001  ID
0151*002  FNAME
0151*003  LNAME
```

Please note that you may not use the alias names ID, FNAME, and LNAME on most clauses in the SQL statement (Aliases are allowed on the order by clause). However, the aliases will be used to generate REC parameters, and must be used on all subsequent CA-Culprit statements.

Generated REC cards

*****	REC	START	SIZE	TYPE	DP	FIELD-NAME

	REC	0057	00015			FNAME
	REC	0072	00004	1		FNAME_NULL_IND
	REC	0049	00004	2		ID
	REC	0053	00004	1		ID_NULL_IND
	REC	0076	00020			LNAME
	REC	0096	00004	1		LNAME_NULL_IND

15.3 Retrieving floating point data from SQL columns

Although CA-Culprit 15.0 does not support **real** and **double precision** as REC parameter data types, there is a simple coding technique to work around the deficiency.

CA-Culprit will treat real SQL columns as four byte alphanumeric fields. Double precision columns will be treated as eight byte alphanumeric fields. 16 byte packed decimal work fields can be defined, and CULLUS36 can be used to convert the floating point values to packed decimal. The CA-Culprit program below illustrates the technique.

Converting to packed decimal

```
col. 2
↓
IN DB(Q) DICTIONARY=TSTDICT SCHEMA=INV
SQL SELECT * FROM FLOAT_TABLE;
$ CA-Culprit will generate the REC cards shown below.
$REC REAL_DATA    49 4           $ GENERATED
$REC DOUBLE_DATA   53 8           $ GENERATED
$
010 WK_REAL      0.000000
010 WK_DOUBLE     0.000000000000
010 RETCODE '
0151*001 WK_REAL HF
0151*002 WK_DOUBLE HF
017   CALL US36(REAL_DATA, 'S', WK_REAL, 6, RETCODE)
017   IF RETCODE NE 200
017   CALL US36(DOUBLE_DATA, 'D', WK_DOUBLE, 12, RETCODE)
017   IF RETCODE NE 200
017   TAKE
017200 CALL US48(' ERROR IN FLOATING POINT CONVERSION%')
017   DROP
```

The arguments for CULLUS36 are described in the table below:

US36 arguments

Argument	Description
ARG1 field-name	Name of input field containing the floating point value
ARG2 indicator	'S' = Single Precision: 'D' = Double Precision
ARG3 result-field	Name of 16 byte packed decimal work field
ARG4 decimal-pts	Integer indicating number of decimal places in result field
ARG5 return-code	Name of four byte alpha work field

15.4 Displaying an SQL column with data type BINARY

The BINARY data type is a CA-IDMS/DB extension to ANSI SQL. It is used to store hexadecimal literals up to 32,760 bytes in length.

During retrieval, CA-Culprit treats columns with SQL data type BINARY as an alphanumeric field. To display the binary data on a report, use CULLUS31 to obtain a hexadecimal representation of the actual data.

In the example below, the SYSTEM.CONSTRAINT table has a column named REFCOLUMNS defined as BINARY(64). To print the data in hex on our report, displaying 10 bytes on each detail line.

Report output

REPORT 01	CONSTRAINT TABLE REPORT	08/21/99 PAGE 1
CONSTRAINT NAME		REFCOLUMNS 01.....05.....10
REFERENCED-TABLE	BYTES 1 - 10	00010002000000000000
	BYTES 11 - 20	00000000000000000000
	BYTES 21 - 30	00000000000000000000
	BYTES 31 - 40	00000000000000000000
	BYTES 41 - 50	00000000000000000000
	BYTES 51 - 60	00000000000000000000
	BYTES 61 - 64	00000000

Report syntax

```
col. 2
↓
PRO  USER=MZC  PW=MZC
IN  300 DB(Q)   SCHEMA=SYSTEM  DICTIONARY=TSTDICT
REC HEX_REFCOL 150 128
REC HEX_REFCOL1 150 20
REC HEX_REFCOL2 170 20
REC HEX_REFCOL3 190 20
REC HEX_REFCOL4 210 20
REC HEX_REFCOL5 230 20
REC HEX_REFCOL6 250 20
REC HEX_REFCOL7 270 20
SQL  SELECT NAME, REFCOLUMNS FROM CONSTRAINT;
01OUT 80 D
013  CONSTRAINT TABLE REPORT
0151*001 NAME HH 'CONSTRAINT' 'NAME'
0151*002 'BYTES 1 - 10'
0151*003 HEX_REFCOL1 HH 'REFCOLUMNS' '01.....05.....10'
0152*002 'BYTES 11 - 20'
0152*003 HEX_REFCOL2
0153*002 'BYTES 21 - 30'
0153*003 HEX_REFCOL3
0154*002 'BYTES 31 - 40'
0154*003 HEX_REFCOL4
0155*002 'BYTES 41 - 50'
0155*003 HEX_REFCOL5
0156*002 'BYTES 51 - 60'
0156*003 HEX_REFCOL6
0157*002 'BYTES 61 - 64'
0157*003 HEX_REFCOL7
017      CALL US31(REFCOLUMNS, 64, HEX_REFCOL)
```

The arguments for CULLUS31 are described in the table below:

US31 arguments

Argument	Description
ARG1 field-name	Name of input field containing the value to be converted
ARG2 length	Numeric literal or 8 byte work field containing the length of the input field. For release 12.0, this value must be in the range 1 through 127.
ARG3 result-field	Name of the output field. Result-field must be twice as long as the length of the input field.

15.5 How to test and display null values

The sales table below contains daily sales data for specific products.

Table definition

```
CREATE TABLE INV.SALES
(PROMPT      CHAR(20) NOT NULL,
SALES_DATE   DATE    NOT NULL,
UNITS_SOLD   INTEGER);
```

Notice that the UNITS_SOLD field may contain null values. A null value means "We don't know how many units we sold". That is much different than the value zero which means "We didn't sell any product on this date".

Here's how to create a detail report that displays null values on the detail line.

Detail report

```
col. 2
↓
IN DB(Q) DICTIONARY=TSTDICT SCHEMA=INV
SQL SELECT * FROM SALES;
01OUT 80 D
0151*001 PRODUCT      HF
0151*002 SALES_DATE   HF
0151*003 UNITS_SOLD   SZ=6  F-  HF
0152*001 PRODUCT
0152*002 SALES_DATE
0152*003 '---null---'
017     IF UNITS_SOLD_NULL_IND = -1      200
017100  TAKE 1
017200  TAKE 2
```

The UNITS_SOLD_NULL_IND will be set to minus one to indicate a null value. A zero value indicates that UNITS_SOLD is not null.

Report output

PRODUCT	SALES_DATE	UNITS_SOLD
WASHING MACHINES	1991-08-05	14
Washing Machines	1991-08-06	---null---
WASHING MACHINES	1991-08-07	0
WASHING MACHINES	1991-08-08	3
WASHING MACHINES	1991-08-09	12

15.6 Handling null values at total time

You can enhance the previous example to produce weekly summary information for each product. You also want to print the average number of units sold per day. The final report should look like this:

Report output

REPORT 01		WEEKLY SALES BY PRODUCT		PAGE 1
PRODUCT	SALES_DATE	UNITS SOLD		
Clothes Driers	1991-08-05	3		
Clothes Driers	1991-08-06	—null---		
Clothes Driers	1991-08-07	—null---		
Clothes Driers	1991-08-08	6		
Clothes Driers	1991-08-09	3		
	Total Weekly Sales	12		
	Average Daily Sales	4.00		
Stoves	1991-08-05	—null---		
Stoves	1991-08-06	—null---		
Stoves	1991-08-07	—null---		
Stoves	1991-08-08	—null---		
Stoves	1991-08-09	—null---		
	Total Weekly Sales	—null---		
	Average Daily Sales	—null---		
Washing Machines	1991-08-05	14		
Washing Machines	1991-08-06	—null---		
Washing Machines	1991-08-07	0		
Washing Machines	1991-08-08	3		
Washing Machines	1991-08-09	12		
	Total Weekly Sales	29		
	Average Daily Sales	7.25		

WHERE clause: First, add a WHERE clause to select data for one week:

```
col. 2
↓
SQL SELECT * FROM SALES
*   WHERE SALES_DATE >= '1991-08-04' AND
*         SALES_DATE <= '1991-08-10';
```

Subtotals: Next, add a title parameter and a SORT parameter to request subtotals by product. A COUNT field has been initialized to one. It will be used to determine how many detail records have been processed for each group. You can automatically accumulate subtotals for COUNT and UNITS_SOLD_NULL_IND if they appear on detail lines. However, since you don't want the detail values to print, use print position zero:

```
col. 2
↓
01SORT  PRODUCT 0, SALES_DATE
010 COUNT 1
013 WEEKLY SALES BY PRODUCT
0151*000  COUNT
0151*000  UNITS_SOLD_NULL_IND
```

Based on the information in the report shown above, the total-time values for PRODUCT, COUNT, and UNITS_SOLD_NULL_IND are:

Subtotal values

PRODUCT	COUNT	UNITS_SOLD_NULL_IND
Washing Machines	5	-1
Clothes Driers	5	-2
Stoves	5	-5

These values let you determine how many UNITS_SOLD for each product are not null. This in turn provides you with the divisor you need to compute the average daily sales for each product. The total-time logic is shown below:

Total-time logic

```

col. 2
↓
010 NOT_NULL_COUNTER 0
010 AVERAGE      0.00
018   IF LEVL = 2 DROP
018   COMPUTE COUNT + UNITS_SOLD_NULL_IND  NOT_NULL_COUNTER
018   IF NOT_NULL_COUNTER = 0  400
018   COMPUTE UNITS_SOLD / NOT_NULL_COUNTER  AVERAGE
018   TAKE (1,2,3)
018400  TAKE (1,4,5)
0161*003  '-----'
0162*002  'Total Weekly Sales'
0162*003  UNITS_SOLD  SZ=6  F-
0163*002  'Average Daily Sales'
0163*003  AVERAGE    SZ=6  F-
0164*002  'Total Weekly Sales'
0164*003  '---null---'
0165*002  'Average Daily Sales'
0165*003  '---null---'
```

15.7 Understanding CA-Culprit decimal point handling

In our STOCKS table example, the CLOSE_PRICE field is defined as having three implied decimal digits.

Table definition

```
CREATE TABLE INV.STOCKS
(COMPANY_NAME CHAR(20) NOT NULL,
 COMPANY_ID CHAR(5),
 TRADING_DATE DATE,
 VOLUME INTEGER,
 CLOSE_PRICE DECIMAL(9,3));
```

This allows you to represent 1/8 point fractions as a decimal value. However when formatting reports, you often want to print the value as a money field, dollars and cents. This means converting the number from DP=3 to DP=2.

To obtain the correctly rounded value, move the CLOSE_PRICE field to a work field, WK_CLOSE defined with two decimal points. Then, reference WK_CLOSE on the detail line. The report below shows why:

Report syntax

```
col. 2
↓
IN DB(Q) DICTIONARY=TSTDICT SCHEMA=INV
SQL SELECT * FROM STOCKS;
01OUT 80 D
010 WK_CLOSE 0.00
0151*001 COMPANY_NAME HF
0151*002 CLOSE_PRICE HF
0151*003 CLOSE_PRICE DP=2 HH 'DP=2 ON' '5 LINE'
0151*004 WK_CLOSE HF
017 MOVE CLOSE_PRICE TO WK_CLOSE
```

Report output

COMPANY_NAME	CLOSE_PRICE	DP=2 ON 5 LINE	WK_CLOSE
Computer Associates	9.125	91.25	9.13
Boston Edison	20.375	203.75	20.38

As you can see, coding **CLOSE_PRICE DP=2** on the edit parameter changes the location of the implied decimal point, and produces erroneous results. To insure correct rounding, move the value to a work variable that was defined with the correct number of decimal points.

15.8 Interpreting common error messages in SQL retrieval

For retrieval processing, SQL Prepare and Describe commands are issued during the pre-compile phase of CA-Culprit. If any errors are detected, you will receive the following messages on your Sequential Parameter Listing.

```
08/14/99          SEQUENTIAL PARAMETER LISTING      V15.0  PAGE 1
00 ** SYSIN **    PRO  USER=MZC  PW=MZC
C200138 INSTALLATION SECURITY OPTION IS NO
           IN DB(Q) SCHEMA=INV  DICTIONARY=TSTDICT
           SQL SELECT * FROM XYZ;
E C700054 ERROR FROM PREPARE SQL SELECT
C700055  SQLCODE: -0004      SQLCERC: 5138
C700056  SQLCESER: 0000     SQLCNRP: 0000
C700057  Table INV.XYZ not found in catalog
E C700052 SQL ERROR ENCOUNTERED
```

Messages 55, 56, and 57 print the error codes and text contained in the SQL Communications Area (SQLCA).

►► Refer to *CA-IDMS SQL Reference Guide*, for more information about the SQLCA error code fields.

Any **E** or **F** level errors associated with the SQL statement will prevent CA-Culprit from generating REC parameters to describe the SQL columns and null indicators. That means you will probably receive messages like "Field Not Defined" on the Input Parameter Listing. Ignore these extraneous errors for now. They will probably go away as soon as you correct the SQL statement.

During the extract phase of CA-Culprit, you issue SQL commands to perform the bulk fetch operation. Any errors at this time appear under Run Time Messages. The SQL error codes and text are be formatted exactly as shown above.

Chapter 16. Creating New SQL Data Tables

16.1 Create table syntax generated by CA-Culprit	16-4
16.2 Converting ASF tables to SQL tables	16-5
16.3 Assigning Null values to an SQL column	16-6
16.4 How CA-Culprit handles data insertion errors	16-7

This chapter illustrates many CA-Culprit coding techniques used to create new CA-IDMS/DB SQL tables.

16.1 Create table syntax generated by CA-Culprit

Information from the OUTPUT parameter and the EDIT parameters is used to generate SQL Create Table syntax during the compile phase of CA-Culprit. To help illustrate this correspondence, a CA-Culprit program fragment is included below along with the SQL commands that are generated:

CA-Culprit syntax

```
col. 2
↓
010 ZERO  0
01OUT      SQLTABLE=INVENTORY  TYPE=CREATE
*          DICTIONARY=TSTDICT  SCHEMA=INV
0151*010  WAREHOUSE    SZ=10  NOT NULL
0151*020  PRODUCT_CODE  SZ=10  NOT NULL
0151*030  UNITS_ON_HAND FP     SZ=4
0151*031  ZERO         FB     SZ=4
```

Compile-phase SQL syntax

```
CONNECT TO TSTDICT;
SET SESSION CURRENT SCHEMA "INV";
CREATE TABLE "INVENTORY"
    ("WAREHOUSE"    CHAR(0010)  NOT NULL,
     "PRODUCT_CODE" CHAR(0010)  NOT NULL,
     "UNITS_ON_HAND" DECIMAL(0007,0000));
COMMIT RELEASE;
```

These SQL commands are generated and issued directly by CA-Culprit; nothing is required of the user.

16.2 Converting ASF tables to SQL tables

The job of converting a ASF table to an SQL table is really quite simple using CA-Culprit. Simply read the ASF table as input, and create an OUTPUT SQL table with TYPE=CREATE.

CA-Culprit syntax

```
col. 2
↓
PRO    USER=MZC  PW=
INPUT   TABLE=INVENTORY  TYPE=COPY    CATALOG=ASFDIR
010UT  SQLTABLE=INVENTORY  TYPE=CREATE
*      DICTIONARY=TSTDICT  SCHEMA=INV
*      BULKROWS=500
010    UNITS_ON_HAND    0
010    ZERO              0
0151*010  WAREHOUSE      SZ=10  NOT NULL
0151*020  PRODUCT_CODE    SZ=10  NOT NULL
0151*030  UNITS_ON_HAND  FP      SZ=4
0151*031  ZERO            FB      SZ=4
017    MOVE ITEMS_IN_STOCK TO UNITS_ON_HAND
017    TAKE
```

The technique of moving ITEMS_IN_STOCK to UNITS_ON_HAND lets the user create new SQL column names for the fields. Data values can be converted from one format within ASF to another SQL data type by simply coding a format code and SZ= on the edit card.

Parameter **0151*031** is the null indicator value for the UNITS_ON_HAND field. It must be present, and must have a data specification of **FB SZ=4**. Often a work field initialized to zero is used to represent a null indicator that may not have been present on the input data file.

16.3 Assigning Null values to an SQL column

The technique for assigning null values to an SQL column is quite simple: set the column null indicator to minus one (-1).

Consider the example of a phonelist. For the WORK_PHONE field, you could use blanks to mean "This person has no work phone" and null to mean "He may have a work phone, but we don't know the number".

On the sequential input file, null values are represented by a question mark (?) in the WORK_PHONE field. The example below reads the input file, converts question marks to null values, and creates a new SQL table containing the phonelist:

Code example

```
col. 2
↓
IN 80
REC FIRST_NAME 1 12
REC LAST_NAME 13 15
REC IN_HOMEPH 28 12
REC IN_WORKPH 40 12
010 HOME_PHONE '
010 WORK_PHONE '
010 SPACES '
010 HOME_PHONE_NULL_IND 0
010 WORK_PHONE_NULL_IND 0
010UT SQLTABLE=PHONELIST TYPE=CREATE
*      DICTIONARY=TSTDICTION SCHEMA=DEMO
0151*010 FIRST_NAME      NOT NULL  WITH DEFAULT
0151*020 LAST_NAME       NOT NULL  WITH DEFAULT
0151*030 HOME_PHONE
0151*031 HOME_PHONE_NULL_IND FB SZ=4
0151*040 WORK_PHONE
0151*041 WORK_PHONE_NULL_IND FB SZ=4
017   IF IN_HOMEPH = '?' 100
017   MOVE IN_HOMEPH TO HOME_PHONE      $ Not Null - Valid Number
017   MOVE 0          TO HOME_PHONE_NULL_IND
017   B 200
017100 MOVE SPACES      TO HOME_PHONE      $ Assign Null Value
017   MOVE -1          TO HOME_PHONE_NULL_IND
017$
017200 IF IN_WORKPH = '?' 300
017   MOVE IN_WORKPH TO WORK_PHONE      $ Not Null - Valid Number
017   MOVE 0          TO WORK_PHONE_NULL_IND
017   B 400
017300 MOVE SPACES      TO WORK_PHONE      $ Assign Null Value
017   MOVE -1          TO WORK_PHONE_NULL_IND
017400 TAKE
```

16.4 How CA-Culprit handles data insertion errors

During the output phase of CA-Culprit, it is possible to encounter errors during the SQL bulk insert. If this occurs, CA-Culprit produces a diagnostic dump of the output data record and continue processing, as shown below:

Output phase diagnostic error

```
E C760001 THE FOLLOWING ROWS WERE NOT STORED DUE TO INSERT ERRORS:
```

OUTPUT RECORD NUMBER	CHAR ZONE DIGIT	RECORD LAYOUT
0005	1023ABCDE.... 444444444FFFF000130000CCCC0000 0000000000FFFF0002C0000123450000 1...5...10....5...20....5...30...

In the above example, the **Output Record Number** indicates that the fifth row being written contained an error.

The **Record Layout** provides a complete vertical hex dump of the CA-Culprit output buffer to assist in debugging. The **Error Code** 1023 indicates a constraint violation. This value was obtained from the SQLCERC field in the SQL Communications Area. A list of common error codes is given below:

SQLCERC codes

Error code	Description
1008	No storage to load data
1010	Load of Table Deadlocked
1023	Constraint Violation
1025	Data Exception
1026	Data Conversion Error
1051	Cardinality Violation
1058	Duplicate Key Violation
1067	I/O Error While Loading Table

Chapter 17. Adding, Replacing, and Dropping Data on SQL Tables

17.1 Updating SQL columns of type DATE	17-3
17.2 Getting the exact syntax for updating an SQL table	17-4
17.3 Drop table example	17-5

17.1 Updating SQL columns of type DATE

SQL Tables that contain columns of type DATE may be updated using the ADD or REPLACE functions. Data to be loaded into a DATE column should be stored in a 10 byte alphanumeric variable or literal. The format of the date value should be '**YYYY-MM-DD**'. CA-IDMS/DB is responsible for converting this date value into its own internal DATE format.

Existing table definition

```
CREATE TABLE STOCKS
((COMPANY_NAME      CHAR(20)  NOT NULL,
  COMPANY_ID       CHAR(5),
  TRADING_DATE    DATE,
  VOLUME          INTEGER,
  CLOSE_PRICE     DECIMAL(9,3));
```

Report syntax

```
col. 2
↓
IN 80
REC COMPANY_NAME 1 20
REC COMPANY_ID   21 5
REC VOLUME        26 8 2
REC CLOSE_PRICE   34 9 2 DP=3
010 TRADING_DATE '1991-08-14'
010UT  SQLTABLE=STOCKS  TYPE=ADD
*      DICTIONARY=TSTDICT SCHEMA=INV
010  ZERO 0
0151*010 COMPANY_NAME
0151*020 COMPANY_ID
0151*021 ZERO      FB  SZ=4
0151*030 TRADING_DATE
0151*031 ZERO      FB  SZ=4
0151*040 VOLUME    FB  SZ=4
0151*041 ZERO      FB  SZ=4
0151*050 CLOSE_PRICE FP  SZ=5 DP=3
0151*051 ZERO      FB  SZ=4
```

17.2 Getting the exact syntax for updating an SQL table

In order to update an existing SQL table, it is necessary to know the exact order of the columns, the data types, and the existence of null indicators. Fortunately, there is an easy technique to get the exact CA-Culprit syntax needed to update an SQL table. Here's how:

Code a small CA-Culprit program using TYPE=ADD on the OUTPUT parameter. Deliberately make a mistake on the data type or size of an EDIT parameter. This produces E level errors during the compile phase of CA-Culprit.

Report syntax

```
col. 2
↓
IN 80
REC ID 1 4 2
01OUT SQLTABLE=STOCKS      TYPE=ADD
*      DICTIONARY=TSTDICT   SCHEMA=INV
0151*001 ID FW      $ Deliberate Error!
017    DROP
```

At the end of the compile phase, error messages are produced on the Input Parameter Listing. Message C700071 provides you with sample EDIT parameters to describe the output SQL table exactly. Use this as a template for coding your CA-Culprit program.

Table diagnostics

```
*****
EDIT LINE CC COLUMN  VALUE OR FIELD-NAME AND EDIT OPTIONS
*****
01 5 1 *001 ID FW
E C300064 FORMAT CODE IS INVALID
E C300023 FIELD SIZE IS INVALID

I C700071 USE THE SAMPLE EDIT PARAMETERS BELOW TO CORRECT YOUR PROGRAM
0151*010 COMPANY_NAME          SZ=0020
0151*020 COMPANY_ID           SZ=0005
0151*021 COMPANY_ID_NULL_IND   FB  SZ=0004
0151*030 TRADING_DATE         SZ=0010
0151*031 TRADING_DATE_NULL_IND FB  SZ=0004
0151*040 VOLUME               FB  SZ=0004
0151*041 VOLUME_NULL_IND      FB  SZ=0004
0151*050 CLOSE_PRICE           FP  SZ=0005  DP=0003
0151*061 CLOSE_PRICE_NULL_IND  FB  SZ=0004
```

17.3 Drop table example

The code needed to delete an SQL table definition using CA-Culprit is given below:

CA-Culprit syntax

```
col. 2
↓
IN 80
REC FIELDA 1 4
01OUTPUT SQLTABLE=INVENTORY TYPE=DROP
*      DICTIONARY=TSTDICT SCHEMA=INV
0151*001  '
017    DROP
```

Report output

SQL TABLE UPDATE STATISTICS					
REPORT	FUNCTION	SCHEMA	TABLE NAME	TABLE DEFINITION	ROWS STORED
01	DEL	INV	INVENTORY	DELETED	0

17.3 Drop table example

Appendices

Appendix A. The Personnel File Description

The Commonwealth Corporation personnel file, which is used in examples in this manual, is described fully in the figure below:

The Commonwealth Corporation personnel file description

RECORD SIZE 200 bytes
BLOCK SIZE 400 bytes
RECORD FORMAT FIXED (F)
FILE TYPE SEQUENTIAL (PS)

Data Field	Start Position	Length (bytes)	Data Type
EMPLOYEE-ID	1	4	Zoned decimal
EMPLOYEE-NAME	5	25	Alphanumeric
FIRST-NAME	5	10	Alphanumeric
LAST-NAME	15	15	Alphanumeric
STREET	30	20	Alphanumeric
CITY	50	15	Alphanumeric
STATE	65	2	Alphanumeric
ZIP-CODE	67	5	Alphanumeric
PHONE	72	10	Zoned decimal
STATUS	82	2	Alphanumeric
SS-NUMBER	84	9	Zoned decimal
START-MONTH	93	2	Zoned decimal
START-DAY	95	2	Zoned decimal
START YEAR	97	2	Zoned decimal
TERMINATION-MONTH	99	2	Zoned decimal
TERMINATION-DAY	101	2	Zoned decimal
TERMINATION-YEAR	103	2	Zoned decimal
BIRTH-MONTH	105	2	Zoned decimal
BIRTH-DAY	107	2	Zoned decimal
BIRTH-YEAR	109	2	Zoned decimal
DEPT-ID	111	4	Zoned decimal
DEPT-NAME	115	45	Alphanumeric
SALARY-AMOUNT	160	5	Packed Decimal
JOB-ID	167	4	Zoned decimal
TITLE	171	20	Alphanumeric

Appendix B. Debugging a CA-Culprit Report

B.1	Reviewing CA-Culprit listings and messages	B-4
B.1.1	Sequential parameter listing	B-4
B.1.2	Input parameter listing	B-4
B.1.3	Runtime messages	B-5
B.1.4	The report	B-6
B.2	Using CULLUS48	B-8
B.3	Printing additional report information	B-9
B.3.1	Demonstration	B-9

A CA-Culprit report is debugged by using a combination of standard debugging techniques and the informational, warning, error, and statistical messages provided by CA-Culprit. The following procedural sequence, recommended for finding errors in CA-Culprit coding, uses standard and CA-Culprit techniques:

1. Desk check the code.
2. Review CA-Culprit listings, messages, and error codes, including buffer dumps.
►► Refer to the *CA-Culprit Messages and Codes*, for detailed information.
3. Use procedure module CULLUS48.
4. Print unsorted listings of data and walk through program logic with the data (play computer).
5. Print information beyond that required for the report.
6. Force buffer dumps.

This appendix presents a summary of CA-Culprit capabilities that can be useful in debugging.

B.1 Reviewing CA-Culprit listings and messages

CA-Culprit generates a sequential parameter listing, an input parameter listing, and run-time messages in addition to printed reports. Through these listings, errors can be flagged from all of CA-Culprit's processing phases.

B.1.1 Sequential parameter listing

Preliminary parameter syntax diagnostics (precompile phase) are printed in the sequential parameter listing. Diagnostics are printed immediately below an invalid parameter or at the end of the listing. Asterisks (*) may underline the field in error:

REC SALARY 160 ***** E INCOMPLETE PARAMETER CARD

B.1.2 Input parameter listing

Warning and error messages produced during parameter validation (compile phase) are printed on the input parameter listing. Error messages relating to a particular input parameter appear below the invalid parameter. Errors relating to the entire report are listed at the end. The final message indicates if the extract phase was performed:

09/19/99 ***** INPUT RECORD TYPE BLOCK FILE DESCRIPTION... ***** INPUT 00200 F 00400 PS(TAPE) LT=S ***** REC START SIZE TYPE DP FIELD-NAME RECORD-NAME, LEVEL ***** REC 00160 00007 3 2 SALARY 'SALARY' 'TOTAL'	CAGJF0 V15.0 PAGE 1
--	---------------------

09/19/99 ***** INPUT PARAMETER LISTING ***** CAGJF0 V15.0 PAGE 2 ***** OUTPUT RECORD BLOCK T/D FILE DESCRIPTION... ***** 01 OUT T ***** EDIT LINE CC COLUMN VALUE OR FIELD-NAME AND EDIT OPTIONS... ***** 01 5 1 *020 SALARY ***** EDIT LINE CC COLUMN VALUE OR FIELD-NAME AND EDIT OPTIONS... ***** 01 6 1 *010 'ALL DEPARTMENTS' HH 'DEPARTMENT' 01 6 1 *020 SALARY SZ=0010 HH 'TOTAL SALARIES' C300106 EXTRACT WILL BE PERFORMED C300119 PROFILE OPTION IN EFFECT: RELEASE = 6

B.1.3 Runtime messages

Processing errors resulting from input-file processing, buffer dumps, and end-of-file statistics are listed in run-time messages (extract phase):

- Processing errors include database access and file-matching operations.
- Buffer dumps show the contents of the input buffer in character and hexadecimal representation when an error occurs in input processing.

If an error condition does not create a buffer dump, the dump can be forced by creating a zero-divide condition in the CA-Culprit program:

```
010 ZERO
.
.
.
017    ZERO / ZERO  ZERO
```

The dump looks like this:

09/19/99	RUN TIME MESSAGES	CAGJF0 V15.0 PAGE 1
<pre>C350023 INVALID NUMERIC DATA ENCOUNTERED IN REPORT 01 (CONT) ERROR OCCURRED DURING PROCESSING DETAIL LINE 1 FIELD 1 (CONT) MOST RECENT SUBSCRIPT VALUE 0 (CONT) INPUT BUFFER NUMBER 1 (CONT) RECORD BUFFER DUMP CHAR 0069JUNE BLOOMER 14 ZITHER TERR LEXINGTON MA01675 0103955781880050500 ZONE FFFFDEDC444444CDDDDCD4444444FF4ECECCD4ECDD444444DCECDCEDD444444DCFFFFF4444444444FFFFFFFFF DIGIT 0069145500000023664590000000014099385903599000003579573650000004101675000000000103955781880050500 01...5..10....5..20....5..30....5..40....5..50....5..60....5..70....5..80....5..90....5..00 CHAR 0000 600425ACCOUNTING AND PAYROLL PAYROLL CLERK ZONE FFFF4444FFFFFCCCCDEEDCDC4CDC4DCEDDDD4444444444444444440100444444DCEDDDD4CDCDD4444444444444444 DIGIT 00000006004251336453957015407189633000 101...5..10....5..20....5..30....5..40....5..50....5..60....5..70....5..80....5..90....5..00</pre>		

09/19/99	RUN TIME MESSAGES	CAGJF0 V15.0 PAGE 25
<pre>C350023 INVALID NUMERIC DATA ENCOUNTERED IN REPORT 01 (CONT) ERROR OCCURRED DURING PROCESSING DETAIL LINE 1 FIELD 1 (CONT) MOST RECENT SUBSCRIPT VALUE 0 (CONT) INPUT BUFFER NUMBER 25 (CONT) RECORD BUFFER DUMP CHAR 0341RICHARD MUNYON 17 BLACKHILL DR WESTWOOD MA02090 011100208180111400 ZONE FFFFDCCCCD444DEDED44444444FF4CDCCDCDD4CD444444ECEEEDDC4444444DCFFFFF4444444444FFFFFFFFF DIGIT 0341993819400045865000000000170231328933049000065236664000000410209000000000000111100208180111400 01...5..10....5..20....5..30....5..40....5..50....5..60....5..70....5..80....5..90....5..00 CHAR 0000 501214BRAINSTORMING SNOWBLOWER ZONE FFFF4444FFFFFCDDEEDDCDC444444444444444444444444440300444444EDDECDDDECD4444444444444444 DIGIT 00000005012142919523694957000 101...5..10....5..20....5..30....5..40....5..50....5..60....5..70....5..80....5..90....5..00 ***** C350033 NUMBER OF EXTRACT DATA ERRORS HAS EXCEEDED 25 C350037 MESSAGES SUPPRESSED UNTIL END OF FILE ***** ***** END OF FILE ***** 56 INPUT RECORDS READ 56 DATA ERRORS OCCURRED IN EXTRACT LOGIC 56 ERRORS IN REPORT 01 DEPARTMENT TOTAL SALARIES ALL DEPARTMENTS C750009 RECORDS WRITTEN FOR REPORT 01 -- 3</pre>		

- End-of-file statistics are generated at the conclusion of extract file processing.
The amount of information depends upon the type of run:

B.1 Reviewing CA-Culprit listings and messages

```

09/19/99          RUN TIME MESSAGES          CAGJF0 V15.0 PAGE 1
C350023 INVALID NUMERIC DATA ENCOUNTERED IN REPORT 01 <----- CAUSE OF DUMP
(CONT) ERROR OCCURRED DURING PROCESSING DETAIL LINE 1 FIELD 1 <----- WHERE DUMP OCCURRED
(Cont) MOST RECENT SUBSCRIPT VALUE      0
(Cont) INPUT BUFFER NUMBER      1
(Cont) RECORD BUFFER DUMP
CHAR 0069JUNE     BLOOMER      14 ZITHER TERR    LEXINGTON    MA01675    0103955781880050500
ZONE FFFFDEDC444444CDDDCD444444FF4ECECCD4ECDD444444DCECDCEDD444444DCFFFFF4444444444FFFFFFFFFFFFFFF
DIGIT 0069145500000023664590000000001409938590359900000035795736500000041016750000000000103955781880050500
01...5....10....5...20....5...30....5...40....5...50....5...60....5...70....5...80....5...90....5...00

CHARACTER
REPRESENTATION
OF INPUT BUFFER
HEX REPRESENTATION
OF INPUT BUFFER

```

B.1.4 The report

Logic and format errors, generated in the output phase, can frequently be detected by reviewing the printed report. Error messages are printed where the processing problem occurs to the limit set in the OE= specification on the PROFILE parameter:

REPORT NO. 01	AVERAGE START AGE BY DEPARTMENT	AND STATUS		09/19/99 PAGE	1	BIRTH	YEAR	AGE
DEPARTMENT	EMPLOYEE NAME	STATUS CODE	START YEAR					
ACCOUNTING AND PAYROLL	BLOOMER	01	80			60	20	
ACCOUNTING AND PAYROLL	HUTTON	01	77			41	36	
ACCOUNTING AND PAYROLL	JENSON	01	80			48	32	
ACCOUNTING AND PAYROLL	KIMBALL	01	78			49	29	
ACCOUNTING AND PAYROLL	KING	01	80			60	20	
ACCOUNTING AND PAYROLL	NICEMAN	01	80			55	25	
				NUMBER OF EMPLOYEES BY STATUS CODE:		6		
				AVERAGE AGE			27	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		6		
				AVERAGE AGE			27	
BLUE SKIES	CLOUD	01	77			45	32	
BLUE SKIES	DONOVAN	01	81			51	30	
BLUE SKIES	MOON	01	78			44	34	
				NUMBER OF EMPLOYEES BY STATUS CODE:		3		
				AVERAGE AGE			32	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		3		
				AVERAGE AGE			32	
BRAINSTORMING	BREEZE	01	79			34	45	
BRAINSTORMING	CROW	01	79			44	35	
BRAINSTORMING	LANCHESTER	01	75			32	43	
BRAINSTORMING	MAKER	01	78			45	33	
BRAINSTORMING	MUNYON	01	80			50	30	
BRAINSTORMING	WAGNER	01	78			34	44	
				NUMBER OF EMPLOYEES BY STATUS CODE:		6		
				AVERAGE AGE			38	
BRAINSTORMING	ANDALE	03	78			60	18	
				NUMBER OF EMPLOYEES BY STATUS CODE:		1		
				AVERAGE AGE			18	

B.1 Reviewing CA-Culprit listings and messages

REPORT NO. 01	AVERAGE START AGE BY DEPARTMENT AND STATUS		09/19/99 PAGE	4	BIRTH	AGE
DEPARTMENT	EMPLOYEE NAME	STATUS CODE	START YEAR		YEAR	
PUBLIC RELATIONS	MCDOUGALL	01	80		59	21
PUBLIC RELATIONS	PENMAN	01	77		44	33
				NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE	6	28
PUBLIC RELATIONS	ZEDI	05	76	NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE	40	36
				NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE	1	36
					7	29

THERMOREGULATION	CLOTH	01	79		45	34
THERMOREGULATION	FINN	01	79		38	41
THERMOREGULATION	TIME	01	81		58	23
THERMOREGULATION	WILCO	01	79		50	29
				NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE	4	32
THERMOREGULATION	KASPAR	04	80	NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE	40	40
				NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE	1	40
				NUMBER OF EMPLOYEES FOR THE COMPANY: AVERAGE AGE	5	33
					56	31

C750009 RECORDS WRITTEN FOR REPORT 01 -- 139

B.2 Using CULLUS48

Status or diagnostic messages can be added to the run-time message chapter of a CA-Culprit job by using the CULLUS48 user module. A message containing up to 132 characters can be executed in type 7 logic, as follows:

```
010 MESSAGE '0 THIS IS A MESSAGE %%'  
.  
.  
.  
927 CALL US48 (MESSAGE)
```

Note: The message, an alphanumeric string placed within single quotation marks, must start with a carriage control character and end with two percent signs.

►► More information about CULLUS48 can be found in *CA-Culprit User Modules*.

B.3 Printing additional report information

Debugging complex reports can be made easier by printing additional information in your report. You can:

- Print out sort key values in order to understand CA-Culprit's sort sequence
- Print out detail lines for totals-only reports in order to determine the source of totals generated for the report
- Release values between complex calculations in order to track the processing sequence

B.3.1 Demonstration

Objective: This example lists values obtained while computing the volume of a sphere, in cubic centimeters, from diameters measured in inches. The program releases a detail line after every second arithmetic operation.

Procedure: The program issues several arithmetic instructions to compute the volume of the sphere. CA-Culprit automatically *rounds* the result of a calculation. As the results below indicate, the code should specify truncation for the arithmetic operations in this example.

Code

```
col. 2
↓
IN 80
REC TANK-DIA-IN 1 3
010 RADIUS-IN
010 RADIUS-CM
010 WORK-FLD
010 CUBIC-CM
013 CALCULATING THE CC VOLUME OF A SPHERE FROM A DIAMETER IN INCHES
0152*010 TANK-DIA-IN      HF
0152*020 RADIUS-IN       HF
0152*030 RADIUS-CM       HF
0152*040 WORK-FLD        HF
0152*050 CUBIC-CM        HF
.
.
.
017    TANK-DIA / 2 RADIUS-IN
017    RADIUS-IN X 2.54 RADIUS-CM
017    RELS 2
017    COMPUTE (RADIUS-CM X RADIUS-CM X RADIUS-CM) WORK-FLD
017    WORK-FLD X 3.1416 WORK-FLD
017    RELS 2
017    WORK-FLD X 4 WORK-FLD
017    WORK-FLD / 3 CUBIC-CM
017    RELS 2
.
.
.
```

Result: The values released between calculations are:

B.3 Printing additional report information

TANK-DIA-IN	RADIUS-IN	RADIUS-CM	WORK-FLD	CUBIC-CM
15	8	20		
15	8	20	25,133	
15	8	20	100,532	33,511
45	24	60	125,665	33,511

The result should have been:

TANK-DIA-IN	RADIUS-IN	RADIUS-CM	WORK-FLD	CUBIC-CM
15	7.5	19.05		
15	7.5	19.05	21,718.8	
15	7.5	19.05	86,875.2	28,958.4
45	22.5	57.15	108,594.0	28,958.4

The released values show that rounding occurs in all calculations. Truncation should be indicated in the code for correct results.

Appendix C. Coding for Efficiency

C.1	Eliminating unnecessary records	C-4
C.2	SELECT vs. type 7 logic	C-5
C.3	Avoiding unnecessary logic	C-6
C.4	Combining tasks for multiple reports	C-7
C.5	Avoiding unwanted buffer dumps	C-8
C.6	Creating subtotal work fields for presorted input files	C-9
C.7	JCL considerations	C-11
C.8	Miscellaneous hints	C-12

Efficient coding prevents excessive use of system resources by minimizing overhead and input/output (I/O) processing wherever possible. Factors influencing coding choices include:

- The number of records processed
- The number of reports in a single run
- The number of times an instruction is executed

This chapter presents some of the most common coding alternatives that affect the efficiency of a CA-Culprit program.

C.1 Eliminating unnecessary records

Eliminating the most frequently occurring record first is most efficient. For example:

Elimination of the most frequent record

```
A occurs 2,000 times  
B occurs 1,000 times  
C occurs 50 times  
  
017 COMPANY EQ 'A' DROP      (Tests 3,050 records, drops 2,000)  
017 COMPANY EQ 'B' DROP      (Tests 1,050 records, drops 1,000)  
017 COMPANY EQ 'C' DROP      (Tests 50 records, drops 50)  
4,150 tests required to drop all records.
```

or

Eliminating records randomly is least efficient. For example:

Elimination of random records

```
A occurs 50 times  
B occurs 1,000 times  
C occurs 2,000 times  
  
017 COMPANY EQ 'A' DROP      (Tests 3,050 records, drops 50)  
017 COMPANY EQ 'B' DROP      (Tests 3,000 records, drops 1,000)  
017 COMPANY EQ 'C' DROP      (Tests 2,000 records, drops 2,000)  
8,050 tests required to drop all records.
```

C.2 SELECT vs. type 7 logic

Selection criteria can be applied to the entire run with the SELECT or BYPASS parameter or stated in the procedure logic for each report:

```
SELECT PRODUCT EQ 'CA-CULPRIT'  
      vs.  
017    PRODUCT EQ 'CA-CULPRIT'  
027    PRODUCT EQ 'CA-CULPRIT'  
037    PRODUCT EQ 'CA-CULPRIT'
```

Although it is quicker to write, the SELECT parameter uses more processing overhead than type 7 logic. SELECT only becomes efficient when six or more reports are batched together. Type 7 logic is more efficient than SELECT for a smaller number of reports.

C.3 Avoiding unnecessary logic

Obviously, stream lined code is more efficient than code that generates unnecessary processing.

In the example below, a call for the current date is executed for each record in the input file by processing user module US10:

```
017 CALL US10 ( 2 CURRENT-DATE )
```

The code below shows one method of calling the user module one time for the report, thereby avoiding unnecessary processing. The call executes only when the switch value is Y:

```
010 FIRST-TIME-SW 'Y'  
.  
. .  
017      FIRST-TIME-SW EQ 'N' 010  
017      MOVE 'N' FIRST-TIME-SW  
017      CALL US10 ( 2 CURRENT-DATE) $Executes only on first record  
017010      ...
```

C.4 Combining tasks for multiple reports

Reports created from the same data source can often be grouped together to improve processing efficiency. The most common coding strategies are to:

- **Batch related reports together** to allow CA-Culprit to read the data file once for all the reports. Code reports in ascending numerical sequence.
- **Use global work fields** to allow all the reports in the run access to work field contents. Calculation results can be shared between reports in this manner.
- **Use exact positioning** for edit parameters wherever possible; for example 01510045.
- **Keep relative column numbers low** (*0002, *0004, *0006) to allow for later insertions and still minimize the number of work areas (buckets) CA-Culprit must set up. CA-Culprit uses the highest relative column number to allocate storage for some internal tables:

```
*0002  
*0004      Allocates a table with 6 entries  
*0006  
  
*0100  
*0200      Allocates a table with 300 entries  
*0300
```

C.5 Avoiding unwanted buffer dumps

Buffer dumps should be used for debugging purposes only. Avoid unnecessary dumps by observing the following precautions:

- Test the value of any input field that could be zero (0) and used as a divisor. Code should include a branch around the division when the divisor equals zero (0). The result should be set to zero.
- Use a work field in place of an input field that could be nonnumeric and used in calculations. Move the work field as the program logic dictates.
- Sort match files in the same sequence. Match keys must be in ascending sequence.
- Use the BRANCH to HEAD instruction only when absolutely necessary to avoid the creation of additional records in the extract file.

C.6 Creating subtotal work fields for presorted input files

If your input file is presorted, it is more efficient to use type 7 logic to accumulate totals in a work field rather than allow CA-Culprit to automatically total the fields each time a control break occurs on a SORT parameter.

Two different techniques for reporting total amounts by city are shown below:

1. This example uses type 7 logic to accumulate the total amount in a work field until the value of CONTROL-FLD changes. This method is extremely efficient, since NOSORT is required and only 7 extract records will be written to the extract file.

```
IN 200
REC CONTROL-FLD 1 4
REC INPUT-FLD 5 9 2
0151*002 SAVE-FLD HH 'CONTROL'
0151*004 BUCKET HH 'AMOUNT'
010 SAVE-FLD '
010 BUCKET 0
017 EOF = TAKE $Print last work field
017 CONTROL-FLD NE SAVE-FLD 100 $Control change?
017 INPUT-FLD + BUCKET BUCKET $Accumulate
017 DROP $Get next record
017 100 RELS $Previous control-fld
017 MOVE CONTROL-FLD TO SAVE-FLD $Save current information
017 MOVE INPUT-FLD TO BUCKET $Save first amount
017 DROP $Get next
```

2. This example takes full advantage of CA-Culprit's automatic subtotaling capabilities and does not require a presorted input file. However, it is less efficient because every record on the input file will be written to the extract file, which must then be sorted.

```
IN 200
REC CONTROL-FLD 1 4
REC INPUT-FLD 5 9 2
0261*002 CONTROL-FLD HH 'CONTROL'
0261*004 INPUT-FLD HH 'AMOUNT'
02SORT CONTROL-FLD 0
02OUT T
027 TAKE
```

Output

	AMOUNT
BOSTON	136,409
CLEV	17,500
DENV	140,000
NYC	17,170
PHIL	267,000
SF	20,090
TAMP	2,000
	600,169

When you are coding a CA-Culprit report, keep in mind these efficiency guidelines:

1. Always seek to limit I/O activity by reducing the size of the extract file. DROP records in type 7 logic whenever possible.
2. Avoid coding unnecessary SORT parameters. If your input file is presorted and you want control breaks, use the NOSORT option.
3. CPU activity is less expensive than I/O activity.

C.7 JCL considerations

Defined record lengths, the amount of space allocated, and blocking factors affect CA-Culprit's use of system resources. The following specifications are recommended:

- On **SYS005** (parameter file), the record length *must* be 320 bytes. A quarter or half track is recommended for the block size.
- On **SYS006** (extract file) and **SYS008** (NOSORT file), record length should be at least 256 bytes (larger for nonprint records). Block size should be equal to a half or full track, large enough to avoid overrides. Contiguous space is the most efficient, but not always possible.
- On **SYS007** (SORT parameters), the record length should be 80 bytes and block size should be a quarter track.

C.8 Miscellaneous hints

The following suggestions may prove helpful:

- Document everything the code does. Use a dollar sign (\$) for comments placed in the code.
- Use test files when developing routines that will run against large files.
- Use copied code syntax (USE, =COPY, =MACRO) to read INPUT and REC parameters into code.

Appendix D. Precoded CA-Culprit User Modules

D.1 Input modules	D-4
D.2 Procedure modules	D-5
D.3 Output modules	D-7

The following precoded CA-Culprit user modules are provided on the CA-Culprit installation tape. These modules are compiled and link edited at installation time:

D.1 Input modules

Input modules

Module name	Processing task
CULSPAN	Reads an input file containing spanned records in a VSE/ESA environment
CULLVSAM	Reads key or entry-sequenced VSAM files sequentially or directly

D.2 Procedure modules

Procedure modules

Module name	Processing task
CULLUS00	Provides an interface between CA-Culprit and user-written modules
CULLUS01	Retrieves records from a sequential file in an OS/390 environment
CULLUS10	Returns the system time and date to a user-defined field
CULLUS11	Converts a Julian date (YYDDD) to a Gregorian date (MMDDYY)
CULLUS12	Converts any century date to a user-specified format.
CULLUS14	Converts a Gregorian date (MMDDYY) to a Julian date (YYDDD)
CULLUS15	Converts a date in any format to a user-specified format
CULLUS22	Retrieves records from ISAM files and delivers them to a specified location
CULLUS25	Retrieves records from VSAM files and delivers them to a specified location
CULLUS29	Formats a vertical hexadecimal dump of all or part of a record, input field, or workfield
CULLUS31	Displays an input or work field in hexadecimal representation
CULLUS33	Converts a numeric field in packed signed decimal format to binary format
CULLUS34	Converts a packed signed numeric field to zoned decimal format
CULLUS35	Represents bit settings in display format
CULLUS36	Converts floating point values to decimal integers
CULLUS37	Converts a doubleword binary number to an eight-byte packed decimal number
CULLUS40	Enables VSE/ESA users to send messages to the console operator
CULLUS43	Moves data from a field known to CA-Culprit to a position in another field
CULLUS45	Performs multiple MOVE operations on data
CULLUS46	Searches a string to locate a position of a specified character
CULLUS48	Adds a user-written message to the runtime messages section of a CA-Culprit job

Module name	Processing task
CULLUS50	Converts a binary string to a string of characters or workfields
CULLUS53	Concatenates a maximum of three fields to a single field
CULLUS62	Searches a CA-Culprit table for specified fields
CULLUS64	Maintains a table of user-defined attributes for DDR (Data Dictionary Reporter) reports external to CA-Culprit
CULLUS99	Causes a memory dump

D.3 Output modules

Output modules

Module name	Processing task
CULEDUMP	Print an output line in vertical or horizontal dump format
CULELBL	Creates labels for output
CULEMLIN	Prints multiple output lines and multiple logical footer lines
CULEVSAM	Writes records to a user-defined VSAM file
CULEPOWR	Segments reports in a CA-Culprit job through VSE/POWER

Appendix E. Employee Database Subschema

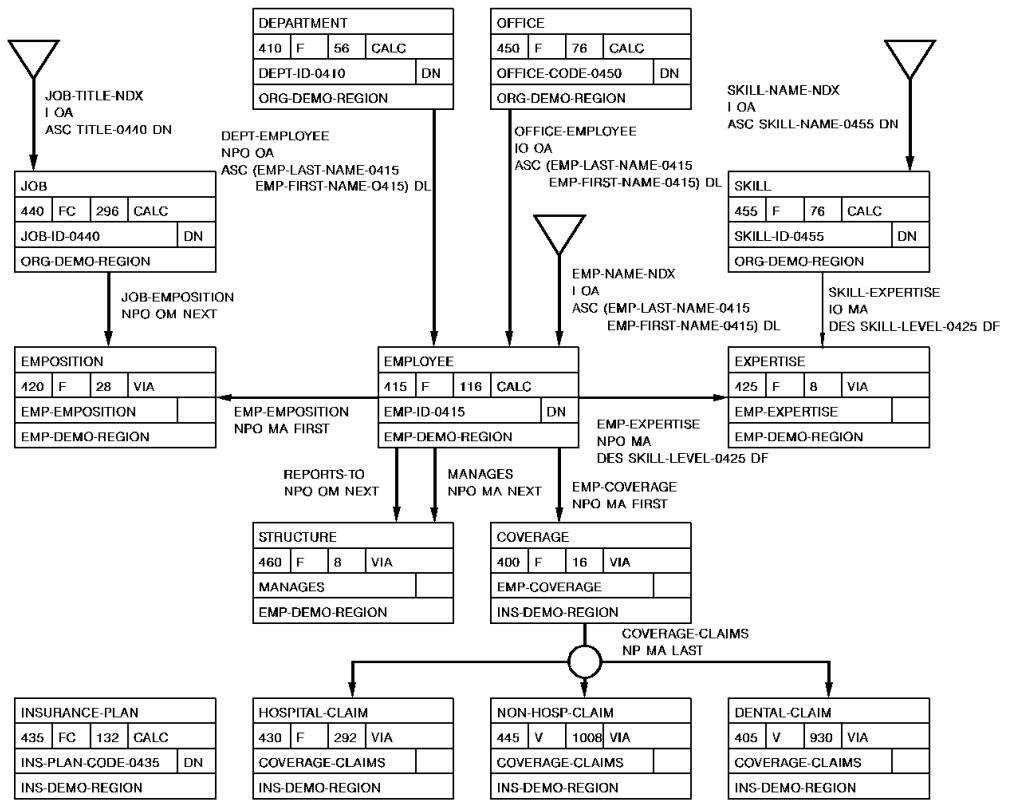
E.1 Data structure diagram	E-4
E.2 Subschema listing	E-5

The employee database is the input for the examples and programs in this manual that reference database records or logical records.

The remainder of this appendix lists the fields defined for each record in the database.

E.1 Data structure diagram

The figure below shows the structure of the database.



E.2 Subschema listing

Below is a listing of the Employee database:

```

SCHEMA=EMPSCHM VERSION=(100)
SUBSCHEMA=(EMPSS01)

RECORD NAME..... COVERAGE RLGTH= 36
RECORD VERSION.... 0100 DLGTH= 20
RECORD ID..... 0400 KLGTH= 16
RECORD LENGTH..... FIXED DSTRT= 16
LOCATION MODE..... VIA SET EMP-COVERAGE DISPLACEMENT 0000 PAGES
WITHIN..... INS-DEMO-REGION OFFSET 5 PGS FOR 20 PGS
DBKEY POSITIONS.... SET..... TYPE..... NEXT PRIOR OWNER
EMP-COVERAGE MEMBER 1 2 3
COVERAGE-CLAIMS OWNER 4
DATA ITEM..... REDEFINES... USAGE..... VALUE..... PICTURE. STRT LGTH
02 SELECTION-DATE-0400 DISPLAY 1 8
03 SELECTION-YEAR-0400 DISPLAY 9(4) 1 4
03 SELECTION-MONTH-0400 DISPLAY 9(2) 5 2
03 SELECTION-DAY-0400 DISPLAY 9(2) 7 2
02 TERMINATION-DATE-0400 DISPLAY 9 8
03 TERMINATION-YEAR-0400 DISPLAY 9(4) 9 4
03 TERMINATION-MONTH-0400 DISPLAY 9(2) 13 2
03 TERMINATION-DAY-0400 DISPLAY 9(2) 15 2
02 TYPE-0400 DISPLAY X 17 1
88 MASTER-0400 COND 'M' 17
88 FAMILY-0400 COND 'F' 17
88 DEPENDENT-0400 COND 'D' 17
02 INS-PLAN-CODE-0400 DISPLAY X(3) 18 3
88 GROUP-LIFE-0400 COND '001' 18
88 HMO-0400 COND '002' 18
88 GROUP-HEALTH-0400 COND '003' 18
88 GROUP-DENTAL-0400 COND '004' 18

RECORD NAME..... DENTAL-CLAIM RLGTH= 944
RECORD VERSION.... 0100 DLGTH= 936
RECORD ID..... 0405 KLGTH= 8
RECORD LENGTH..... VARIABLE DSTRT= 12
MINIMUM ROOT..... 132 CHARACTERS
MINIMUM FRAGMENT... 932 CHARACTERS
LOCATION MODE..... VIA SET COVERAGE-CLAIMS DISPLACEMENT 0000 PAGES
WITHIN..... INS-DEMO-REGION OFFSE 5 PGS FOR 20 PGS
DBKEY POSITIONS.... SET..... TYPE..... NEXT PRIOR OWNER
COVERAGE-CLAIMS MEMBER 1
(FRAGMENT CHAIN) INTRNL 2

```

E.2 Subschema listing

DATA ITEM.....	REDEFINES...	USAGE.....	VALUE.....	PICTURE.	STRT	LGTH
02 CLAIM-DATE-0405		DISPLAY			1	8
03 CLAIM-YEAR-0405		DISPLAY		9(4)	1	4
03 CLAIM-MONTH-0405		DISPLAY		9(2)	5	2
03 CLAIM-DAY-0405		DISPLAY		9(2)	7	2
02 PATIENT-NAME-0405		DISPLAY			9	25
03 PATIENT-FIRST-NAME-0405		DISPLAY		X(10)	9	10
03 PATIENT-LAST-NAME-0405		DISPLAY		X(15)	19	15
02 PATIENT-BIRTH-DATE-0405		DISPLAY			34	8
03 PATIENT-BIRTH-YEAR-0405		DISPLAY		9(4)	34	4
03 PATIENT-BIRTH-MONTH-0405		DISPLAY		9(2)	38	2
03 PATIENT-BIRTH-DAY-0405		DISPLAY		9(2)	40	2
02 PATIENT-SEX-0405		DISPLAY		X	42	1
02 RELATION-TO-EMPLOYEE-0405		DISPLAY		X(10)	43	10
02 DENTIST-NAME-0405		DISPLAY			53	25
03 DENTIST-FIRST-NAME-0405		DISPLAY		X(10)	53	10
03 DENTIST-LAST-NAME-0405		DISPLAY		X(15)	63	15
02 DENTIST-ADDRESS-0405		DISPLAY			78	46
03 DENTIST-STREET-0405		DISPLAY		X(20)	78	20
03 DENTIST-CITY-0405		DISPLAY		X(15)	98	15
03 DENTIST-STATE-0405		DISPLAY		X(2)	113	2
03 DENTIST-ZIP-0405		DISPLAY			115	9
04 DENTIST-ZIP-FIRST-FIVE-0405		DISPLAY		X(5)	115	5
04 DENTIST-ZIP-LAST-FOUR-0405		DISPLAY		X(4)	120	4
02 DENTIST-LICENSE-NUMBER-0405		DISPLAY		9(6)	124	6
02 NUMBER-OF-PROCEDURES-0405		COMP		9(2)	130	2
02 FILLER		DISPLAY		X	132	1
02 DENTIST-CHARGES-0405		DISPLAY	OCCURS 0 TO 10		133	800
DEPENDING ON --- NUMBER-OF-PROCEDURES-0405						
03 TOOTH-NUMBER-0405		DISPLAY		9(2)	1	2
03 SERVICE-DATE-0405		DISPLAY			3	8
04 SERVICE-YEAR-0405		DISPLAY		9(4)	3	4
04 SERVICE-MONTH-0405		DISPLAY		9(2)	7	2
04 SERVICE-DAY-0405		DISPLAY		9(2)	9	2
03 PROCEDURE-CODE-0405		DISPLAY		9(4)	11	4
03 DESCRIPTION-OF-SERVICE-0405		DISPLAY		X(60)	15	60
03 FEE-0405		COMP-3		S9(7)V99	75	5
03 FILLER		DISPLAY		X	80	1

RECORD NAME.....	DEPARTMENT	RLGTH=	72
RECORD VERSION....	0100	DLGTH=	56
RECORD ID.....	0410	KLGTH=	16
RECORD LENGTH.....	FIXED	DSTRT=	16
LOCATION MODE.....	CALC USING DEPT-ID-0410	DUPLICATES NOT ALLOWED	
WITHIN.....	ORG-DEMO-REGION	OFFSET	5 PGS FOR 20 PGS
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT PRIOR OWNER
	CALC	MEMBER	1 2
	DEPT-EMPLOYEE	INDEX OWNER	3 4

DATA ITEM.....	REDEFINES...	USAGE.....	VALUE.....	PICTURE.	STRT	LGTH
02 DEPT-ID-0410		DISPLAY		9(4)	1	4
02 DEPT-NAME-0410		DISPLAY		X(45)	5	45
02 DEPT-HEAD-ID-0410		DISPLAY		9(4)	50	4
02 FILLER		DISPLAY		XXX	54	3
 RECORD NAME.....	EMPLOYEE			RLGTH=	192	
RECORD VERSION....	0100			DLGTH=	120	
RECORD ID.....	0415			KLGTH=	72	
RECORD LENGTH.....	FIXED			DSTRT=	72	
LOCATION MODE.....	CALC USING EMP-ID-0415			DUPLICATES NOT ALLOWED		
WITHIN.....	EMP-DEMO-REGION	OFFSET		5 PGS FOR	45 PGS	
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT	PRIOR OWNER		
	CALC	MEMBER	1	2		
	DEPT-EMPLOYEE	INDEX MEMBER	3		4	
	EMP-NAME-NDX	INDEX MEMBER	5			
	EMP-SSN-NDX	INDEX MEMBER	6			
	OFFICE-EMPLOYEE	INDEX MEMBER	7		8	
	EMP-COVERAGE	OWNER	9	10		
	EMP-EMPOSITION	OWNER	11	12		
	EMP-EXPERTISE	OWNER	13	14		
	MANAGES	OWNER	15	16		
	REPORTS-TO	OWNER	17	18		
 DATA ITEM.....	REDEFINES...	USAGE.....	VALUE.....	PICTURE.	STRT	LGTH
02 EMP-ID-0415		DISPLAY		9(4)	1	4
02 EMP-NAME-0415		DISPLAY			5	25
03 EMP-FIRST-NAME-0415		DISPLAY		X(10)	5	10
03 EMP-LAST-NAME-0415		DISPLAY		X(15)	15	15
02 EMP-ADDRESS-0415		DISPLAY			30	46
03 EMP-STREET-0415		DISPLAY		X(20)	30	20
03 EMP-CITY-0415		DISPLAY		X(15)	50	15
03 EMP-STATE-0415		DISPLAY		X(2)	65	2
03 EMP-ZIP-0415		DISPLAY			67	9
04 EMP-ZIP-FIRST-FIVE-0415		DISPLAY		X(5)	67	5
04 EMP-ZIP-LAST-FOUR-0415		DISPLAY		X(4)	72	4
02 EMP-PHONE-0415		DISPLAY		9(10)	76	10
02 STATUS-0415		DISPLAY		X(2)	86	2
88 ACTIVE-0415	COND	'01'			86	
88 ST-DISABIL-0415	COND	'02'			86	
88 LT-DISABIL-0415	COND	'03'			86	
88 LEAVE-OF-ABSENCE-0415	COND	'04'			86	
88 TERMINATED-0415	COND	'05'			86	

02 SS-NUMBER-0415	DISPLAY	9(9)	88	9
02 START-DATE-0415	DISPLAY		97	8
03 START-YEAR-0415	DISPLAY	9(4)	97	4
03 START-MONTH-0415	DISPLAY	9(2)	101	2
03 START-DAY-0415	DISPLAY	9(2)	103	2
02 TERMINATION-DATE-0415	DISPLAY		105	8
03 TERMINATION-YEAR-0415	DISPLAY	9(4)	105	4
03 TERMINATION-MONTH-0415	DISPLAY	9(2)	109	2
03 TERMINATION-DAY-0415	DISPLAY	9(2)	111	2
02 BIRTH-DATE-0415	DISPLAY		113	8
03 BIRTH-YEAR-0415	DISPLAY	9(4)	113	4
03 BIRTH-MONTH-0415	DISPLAY	9(2)	117	2
03 BIRTH-DAY-0415	DISPLAY	9(2)	119	2
RECORD NAME..... EMPOSITION			RLGTH=	56
RECORD VERSION.... 0100			DLGTH=	32
RECORD ID..... 0420			KLGTH=	24
RECORD LENGTH..... FIXED			DSTRT=	24
LOCATION MODE..... VIA SET	EMP-EMPOSITION	DISPLACEMENT	0000 PAGES	
WITHIN..... EMP-DEMO-REGION	OFFSET	5 PGS FOR	45 PGS	
DBKEY POSITIONS.... SET..... TYPE..... NEXT PRIOR OWNER				
	EMP-EMPOSITION MEMBER	1 2 3		
	JOB-EMPOSITION MEMBER	4 5 6		
DATA ITEM..... REDEFINES... USAGE..... VALUE..... PICTURE. STRT LGTH				
02 START-DATE-0420	DISPLAY		1	8
03 START-YEAR-0420	DISPLAY	9(4)	1	4
03 START-MONTH-0420	DISPLAY	9(2)	5	2
03 START-DAY-0420	DISPLAY	9(2)	7	2
02 FINISH-DATE-0420	DISPLAY		9	8
03 FINISH-YEAR-0420	DISPLAY	9(4)	9	4
03 FINISH-MONTH-0420	DISPLAY	9(2)	13	2
03 FINISH-DAY-0420	DISPLAY	9(2)	15	2
02 SALARY-GRADE-0420	DISPLAY	9(2)	17	2
02 SALARY-AMOUNT-0420	COMP-3	S9(7)V99	19	5
02 BONUS-PERCENT-0420	COMP-3	SV999	24	2
02 COMMISSION-PERCENT-0420	COMP-3	SV999	26	2
02 OVERTIME-RATE-0420	COMP-3	S9V99	28	2
02 FILLER	DISPLAY	XXX	30	3
RECORD NAME..... EXPERTISE			RLGTH=	32
RECORD VERSION.... 0100			DLGTH=	12
RECORD ID..... 0425			KLGTH=	20
RECORD LENGTH..... FIXED			DSTRT=	20
LOCATION MODE..... VIA SET	EMP-EXPERTIS	DISPLACEMENT	0000 PAGES	
WITHIN..... EMP-DEMO-REGION	OFFSET	5 PGS FOR	45 PGS	
DBKEY POSITIONS.... SET..... TYPE..... NEXT PRIOR OWNER				
	EMP-EXPERTISE MEMBER	1 2 3		
	SKILL-EXPERTISE INDEX MEMBER	4 5		
DATA ITEM..... REDEFINES... USAGE..... VALUE..... PICTURE. STRT LGTH				

02 SKILL-LEVEL-0425	DISPLAY	XX	1	2
88 EXPERT-0425	COND	'04'	1	
88 PROFICIENT-0425	COND	'03'	1	
88 COMPETENT-0425	COND	'02'	1	
88 ELEMENTARY-0425	COND	'01'	1	
02 EXPERTISE-DATE-0425	DISPLAY		3	8
03 EXPERTISE-YEAR-0425	DISPLAY	9(4)	3	4
03 EXPERTISE-MONTH-0425	DISPLAY	9(2)	7	2
03 EXPERTISE-DAY-0425	DISPLAY	9(2)	9	2
02 FILLER	DISPLAY	XX	11	2

RECORD NAME.....	HOSPITAL-CLAIM	RLGTH=	304
RECORD VERSION....	0100	DLGTH=	300
RECORD ID.....	0430	KLGTH=	4
RECORD LENGTH.....	FIXED	DSTRT=	4
LOCATION MODE.....	VIA SET	COVERAGE-CLAIMS	DISPLACEMENT 0000 PAGES
WITHIN.....	INS-DEMO-REGION	OFFSET	5 PGS FOR 20 PGS
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT PRIOR OWNER
	COVERAGE-CLAIMS MEMBER		1
DATA ITEM.....	REDEFINES...	USAGE.....	PICTURE. STRT LGTH
02 CLAIM-DATE-0430	DISPLAY		1 8
03 CLAIM-YEAR-0430	DISPLAY	9(4)	1 4
03 CLAIM-MONTH-0430	DISPLAY	9(2)	5 2
03 CLAIM-DAY-0430	DISPLAY	9(2)	7 2
02 PATIENT-NAME-0430	DISPLAY		9 25
03 PATIENT-FIRST-NAME-0430	DISPLAY	X(10)	9 10
03 PATIENT-LAST-NAME-0430	DISPLAY	X(15)	19 15
02 PATIENT-BIRTH-DATE-0430	DISPLAY		34 8
03 PATIENT-BIRTH-YEAR-0430	DISPLAY	9(4)	34 4
03 PATIENT-BIRTH-MONTH-0430	DISPLAY	9(2)	38 2
03 PATIENT-BIRTH-DAY-0430	DISPLAY	9(2)	40 2
02 PATIENT-SEX-0430	DISPLAY	X	42 1
02 RELATION-TO-EMPLOYEE-0430	DISPLAY	X(10)	43 10
02 HOSPITAL-NAME-0430	DISPLAY	X(25)	53 25
02 HOSP-ADDRESS-0430	DISPLAY		78 46
03 HOSP-STREET-0430	DISPLAY	X(20)	78 20
03 HOSP-CITY-0430	DISPLAY	X(15)	98 15
03 HOSP-STATE-0430	DISPLAY	X(2)	113 2
03 HOSP-ZIP-0430	DISPLAY		115 9
04 HOSP-ZIP-FIRST-FIVE-0430	DISPLAY	X(5)	115 5

04 HOSP-ZIP-LAST-FOUR-0430	DISPLAY	X(4)	120	4
02 ADMIT-DATE-0430	DISPLAY		124	8
03 ADMIT-YEAR-0430	DISPLAY	9(4)	124	4
03 ADMIT-MONTH-0430	DISPLAY	9(2)	128	2
03 ADMIT-DAY-0430	DISPLAY	9(2)	130	2
02 DISCHARGE-DATE-0430	DISPLAY		132	8
03 DISCHARGE-YEAR-0430	DISPLAY	9(4)	132	4
03 DISCHARGE-MONTH-0430	DISPLAY	9(2)	136	2
03 DISCHARGE-DAY-0430	DISPLAY	9(2)	138	2
02 DIAGNOSIS-0430	DISPLAY OCCURS 2	X(60)	140	120
02 HOSPITAL-CHARGES-0430	DISPLAY		260	41
03 ROOM-AND-BOARD-0430	DISPLAY		260	26
04 WARD-0430	DISPLAY		260	13
05 WARD-DAYS-0430	COMP-3	S9(5)	260	3
05 WARD-RATE-0430	COMP-3	S9(7)V99	263	5
05 WARD-TOTAL-0430	COMP-3	S9(7)V99	268	5
04 SEMI-PRIVATE-0430	DISPLAY		273	13
05 SEMI-DAYS-0430	COMP-3	S9(5)	273	3
05 SEMI-RATE-0430	COMP-3	S9(7)V99	276	5
05 SEMI-TOTAL-0430	COMP-3	S9(7)V99	281	5
03 OTHER-CHARGES-0430	DISPLAY		286	15
04 DELIVERY-COST-0430	COMP-3	S9(7)V99	286	5
04 ANESTHESIA-COST-0430	COMP-3	S9(7)V99	291	5
04 LAB-COST-0430	COMP-3	S9(7)V99	296	5

RECORD NAME.....	INSURANCE-PLAN	RLGTH=	140
RECORD VERSION....	0100	DLGTH=	132
RECORD ID.....	0435	KLGTH=	8
RECORD LENGTH.....	FIXED	DSTRT=	8
LOCATION MODE.....	CALC USING INS-PLAN-CODE-0435	DUPLICATES NOT ALLOWED	
WITHIN.....	INS-DEMO-REGION	OFFSET	1 PGS FOR 4 PGS
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT PRIOR OWNER
	CALC	MEMBER	1 2
DATA ITEM.....	REDEFINES...	USAGE.....	VALUE..... PICTURE. STRT LGTH
02 INS-PLAN-CODE-0435	DISPLAY		X(3) 1 3

88 GROUP-LIFE-0435	COND	'001'	1
88 HMO-0435	COND	'002'	1
88 GROUP-HEALTH-0435	COND	'003'	1
88 GROUP-DENTAL-0435	COND	'004'	1
02 INS-CO-NAME-0435	DISPLAY	X(45)	4 45
02 INS-CO-ADDRESS-0435	DISPLAY		49 46
03 INS-CO-STREET-0435	DISPLAY	X(20)	49 20
03 INS-CO-CITY-0435	DISPLAY	X(15)	69 15
03 INS-CO-STATE-0435	DISPLAY	X(2)	84 2
03 INS-CO-ZIP-0435	DISPLAY		86 9
04 INS-CO-ZIP-FIRST-FIVE-0435	DISPLAY	X(5)	86 5
04 INS-CO-ZIP-LAST-FOUR-0435	DISPLAY	X(4)	91 4
02 INS-CO-PHONE-0435	DISPLAY	9(10)	95 10
02 GROUP-NUMBER-0435	DISPLAY	9(6)	105 6
02 PLAN-DESCRIPTION-0435	DISPLAY		111 20
03 DEDUCT-0435	COMP-3	S9(7)V99	111 5
03 MAXIMUM-LIFE-COST-0435	COMP-3	S9(7)V99	116 5
03 FAMILY-COST-0435	COMP-3	S9(7)V99	121 5
03 DEP-COST-0435	COMP-3	S9(7)V99	126 5
02 FILLER	DISPLAY	XX	131 2

RECORD NAME.....	JOB	RLGTH=	324
RECORD VERSION....	0100	DLGTH=	300
RECORD ID.....	0440	KLGTH=	24
RECORD LENGTH.....	FIXED (INTERNALLY VARIABLE)	DSTRT=	28
MINIMUM ROOT.....	24 CHARACTERS		
MINIMUM FRAGMENT...	296 CHARACTERS		
LOCATION MODE.....	CALC USING JOB-ID-0440	DUPLICATES NOT ALLOWED	
WITHIN.....	ORG-DEMO-REGION OFFSET	5 PGS FOR	20 PGS
CALL PROCEDURES....	NAME.... WHEN.. FUNCTION IDMSCOMP BEFORE STORE IDMSCOMP BEFORE MODIFY IDMSDCOM AFTER GET		
DBKEY POSITIONS....	SET..... TYPE..... NEXT PRIOR OWNER CALC MEMBER 1 2 JOB-TITLE-NDX INDEX MEMBER 3 JOB-EMPOSITION OWNER 4 5 (FRAGMENT CHAIN) INTRNL 6		
DATA ITEM.....	REDEFINES... USAGE..... VALUE..... PICTURE. STRT LGTH		
02 JOB-ID-0440	DISPLAY 9(4) 1 4		
02 TITLE-0440	DISPLAY X(20) 5 20		

E.2 Subschema listing

02 DESCRIPTION-0440	DISPLAY		25	120
03 DESCRIPTION-LINE-0440	DISPLAY OCCURS 2	X(60)	25	120
02 REQUIREMENTS-0440	DISPLAY		145	120
03 REQUIREMENT-LINE-0440	DISPLAY OCCURS 2	X(60)	145	120
02 MINIMUM-SALARY-0440	DISPLAY	S9(6)V99	265	8
02 MAXIMUM-SALARY-0440	DISPLAY	S9(6)V99	273	8
02 SALARY-GRADES-0440	DISPLAY OCCURS 4	9(2)	281	8
02 NUMBER-OF-POSITIONS-0440	DISPLAY	9(3)	289	3
02 NUMBER-OPEN-0440	DISPLAY	9(3)	292	3
02 FILLER	DISPLAY	XX	295	2

RECORD NAME.....	NON-HOSP-CLAIM	RLGTH=	1064	
RECORD VERSION....	0100	DLGTH=	1056	
RECORD ID.....	0445	KLGTH=	8	
RECORD LENGTH.....	VARIABLE	DSTRT=	12	
MINIMUM ROOT.....	248 CHARACTERS			
MINIMUM FRAGMENT...	1052 CHARACTERS			
LOCATION MODE.....	VIA SET COVERAGE-CLAIMS	DISPLACEMENT	0000 PAGES	
WITHIN.....	INS-DEMO-REGION OFFSET	5 PGS FOR	20 PGS	
DBKEY POSITIONS....	SET..... TYPE..... NEXT PRIOR OWNER			
	COVERAGE-CLAIMS MEMBER	1		
	(FRAGMENT CHAIN) INTRNL	2		
DATA ITEM.....	REDEFINES... USAGE..... VALUE..... PICTURE.	STRT	LGTH	
02 CLAIM-DATE-0445	DISPLAY	1	8	
03 CLAIM-YEAR-0445	DISPLAY	9(4)	1	4
03 CLAIM-MONTH-0445	DISPLAY	9(2)	5	2
03 CLAIM-DAY-0445	DISPLAY	9(2)	7	2
02 PATIENT-NAME-0445	DISPLAY		9	25
03 PATIENT-FIRST-NAME-0445	DISPLAY	X(10)	9	10
03 PATIENT-LAST-NAME-0445	DISPLAY	X(15)	19	15
02 PATIENT-BIRTH-DATE-0445	DISPLAY		34	8
03 PATIENT-BIRTH-YEAR-0445	DISPLAY	9(4)	34	4
03 PATIENT-BIRTH-MONTH-0445	DISPLAY	9(2)	38	2
03 PATIENT-BIRTH-DAY-0445	DISPLAY	9(2)	40	2
02 PATIENT-SEX-0445	DISPLAY	X	42	1
02 RELATION-TO-EMPLOYEE-0445	DISPLAY	X(10)	43	10
02 PHYSICIAN-NAME-0445	DISPLAY		53	25

03 PHYSICIAN-FIRST-NAME-0445	DISPLAY	X(10)	53	10
03 PHYSICIAN-LAST-NAME-0445	DISPLAY	X(15)	63	15
02 PHYSICIAN-ADDRESS-0445	DISPLAY		78	46
03 PHYSICIAN-STREET-0445	DISPLAY	X(20)	78	20
03 PHYSICIAN-CITY-0445	DISPLAY	X(15)	98	15
03 PHYSICIAN-STATE-0445	DISPLAY	X(2)	113	2
03 PHYSICIAN-ZIP-0445	DISPLAY		115	9
04 PHYSICIAN-ZIP-FIRST-FIVE-0445	DISPLAY	X(5)	115	5
04 PHYSICIAN-ZIP-LAST-FOUR-0445	DISPLAY	X(4)	120	4
02 PHYSICIAN-ID-0445	DISPLAY	9(6)	124	6
02 DIAGNOSIS-0445	DISPLAY OCCURS 2	X(60)	130	120
02 NUMBER-OF-PROCEDURES-0445	COMP	9(2)	250	2
02 FILLER	DISPLAY	X	252	1
02 PHYSICIAN-CHARGES-0445	DISPLAY OCCURS 0 TO 10 DEPENDING ON -- NUMBER-OF-PROCEDURES-0445		253	800
03 SERVICE-DATE-0445	DISPLAY		1	8
04 SERVICE-YEAR-0445	DISPLAY	9(4)	1	4
04 SERVICE-MONTH-0445	DISPLAY	9(2)	5	2
04 SERVICE-DAY-0445	DISPLAY	9(2)	7	2
03 PROCEDURE-CODE-0445	DISPLAY	9(4)	9	4
03 DESCRIPTION-OF-SERVICE-0445	DISPLAY	X(60)	13	60
03 FEE-0445	COMP-3	S9(7)V99	73	5
03 FILLER	DISPLAY	XXX	78	3

RECORD NAME..... OFFICE		RLGTH=	92
RECORD VERSION.... 0100		DLGTH=	76
RECORD ID..... 0450		KLGTH=	16
RECORD LENGTH..... FIXED		DSTRT=	16
LOCATION MODE..... CALC USING OFFICE-CODE-0450	DUPLICATES NOT ALLOWED		
WITHIN..... ORG-DEMO-REGION	OFFSET	5 PGS FOR	20 PGS
DBKEY POSITIONS.... SET..... TYPE..... NEXT PRIOR OWNER			
CALC MEMBER	1 2		
OFFICE-EMPLOYEE INDEX OWNER	3 4		
DATA ITEM..... REDEFINES... USAGE..... VALUE..... PICTURE.	STRT	LGTH	
02 OFFICE-CODE-0450	DISPLAY	X(3)	1 3
02 OFFICE-ADDRESS-0450	DISPLAY		4 46
03 OFFICE-STREET-0450	DISPLAY	X(20)	4 20
03 OFFICE-CITY-0450	DISPLAY	X(15)	24 15
03 OFFICE-STATE-0450	DISPLAY	X(2)	39 2
03 OFFICE-ZIP-0450	DISPLAY		41 9
04 OFFICE-ZIP-FIRST-FIVE-0450	DISPLAY	X(5)	41 5
04 OFFICE-ZIP-LAST-FOUR-0450	DISPLAY	X(4)	46 4
02 OFFICE-PHONE-0450	DISPLAY OCCURS 3	9(7)	50 21
02 OFFICE-AREA-CODE-0450	DISPLAY	X(3)	71 3
02 SPEED-DIAL-0450	DISPLAY	X(3)	74 3

RECORD NAME.....	SKILL	RLGTH=	96			
RECORD VERSION....	0100	DLGTH=	76			
RECORD ID.....	0455	KLGTH=	20			
RECORD LENGTH.....	FIXED	DSTRT=	20			
LOCATION MODE.....	CALC USING SKILL-ID-0455	DUPLICATES NOT ALLOWED				
WITHIN.....	ORG-DEMO-REGION	OFFSET	5 PGS FOR 20 PGS			
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT PRIOR OWNER			
	CALC	MEMBER	1 2			
	SKILL-NAME-NDX	INDEX MEMBER	3			
	SKILL-EXPERTISE	INDEX OWNER	4 5			
DATA ITEM.....	REDEFINES...	USAGE.....	VALUE.....	PICTURE.	STRT	LGTH
02 SKILL-ID-0455		DISPLAY		9(4)	1	4
02 SKILL-NAME-0455		DISPLAY		X(12)	5	12
02 SKILL-DESCRIPTION-0455		DISPLAY		X(60)	17	60
RECORD NAME.....	STRUCTURE	RLGTH=	36			
RECORD VERSION....	0100	DLGTH=	12			
RECORD ID.....	0460	KLGTH=	24			
RECORD LENGTH.....	FIXED	DSTRT=	24			
LOCATION MODE.....	VIA SET	MANAGES	DISPLACEMENT 0000 PAGES			
WITHIN.....	EMP-DEMO-REGION	OFFSET	5 PGS FOR 45 PGS			
DBKEY POSITIONS....	SET.....	TYPE.....	NEXT PRIOR OWNER			
	MANAGES	MEMBER	1 2 3			
	REPORTS-TO	MEMBER	4 5 6			
DATA ITEM.....	REDEFINES...	USAGE.....	VALUE.....	PICTURE.	STRT	LGTH
02 STRUCTURE-CODE-0460		DISPLAY		X(2)	1	2
88 ADMIN-0460		COND 'A'			1	
88 PROJECT-0460		COND 'P1' THRU 'P9'			1	
02 STRUCTURE-DATE-0460		DISPLAY			3	8
03 STRUCTURE-YEAR-0460		DISPLAY		9(4)	3	4
03 STRUCTURE-MONTH-0460		DISPLAY		9(2)	7	2
03 STRUCTURE-DAY-0460		DISPLAY		9(2)	9	2
02 FILLER		DISPLAY		XX	11	2

Appendix F. Summary of CA-Culprit parameters

F.1	The basic CA-Culprit functions	F-4
F.2	Processing operations	F-5
F.3	Summary of advanced capabilities for standard files	F-6
F.4	Summary of parameters for using database data	F-8
F.5	Summary of advanced capabilities for accessing the database	F-9
F.6	Minimum coding requirements for using data tables	F-10
F.7	Minimum coding requirements for using SQL tables	F-12

This appendix presents tables that summarize the CA-Culprit parameters presented in this manual.

F.1 The basic CA-Culprit functions

The basic CA-Culprit functions

Function	What CA-Culprit needs	What to use
Create the basic report.	Record size and type	IN
	Fields being used	REC
	Size of the report	OUT
	Output organization	Type 5
Add a title.	Title details	Type 3
Generate subtitles.	Subtitle details	Type 4
Generate column headings.	Where to find the header information	HH HR HF
Adjust column widths.	Column size	SZ=
Format numeric output.	A template	F\$ FS FD Edit masks
Create multiple reports.	Report identifier	Report numbers
Sequence data.	The field(s) and their order	SORT
Process selected data.	Selection criteria	SELECT/BYPASS
Reference multiple detail lines.	Line identifier	Type 5 line numbers
Generate subtotals and page breaks.	Break codes	1 0 - +
Control the output of totals.	Identification of the field to be totaled	Type 6

F.2 Processing operations

Processing operations

Operation	Command	Action
Input processing	Type 7 parameter	Acts upon input data and writes it to the extract file
Output processing	Type 8 parameter	Acts upon data stored in the extract file for report output
Arithmetic	ADD (+) MINUS (-) TIMES (X) DIVIDE (/) COMPUTE	Add Subtract Multiply Divide Performs compound arithmetic operations
Conditional	IF	Performs a comparison or a test
Assignment	MOVE	Transfers a value to a field
Processing control	TAKE DROP RELS	Prints selected edit lines and stops processing Stops processing current record Prints selected edit lines without interrupting processing
	PERFORM	Sends the processing to a specified line or chapter of code
	RETURN	Returns processing to code immediately following the related PERFORM statement

F.3 Summary of advanced capabilities for standard files

Summary of advanced capabilities for standard files

Task	CA-Culprit instruction
Print totals only.	OUT T Type 8 logic Type 6 edit line
Manipulate system-maintained totals.	Type 8 parameter
Define items to be totaled.	Type 6 parameter
Format total lines.	
Obtain current work field value.	Type 6 parameter Type 8 parameter Subscripted work fields on type 6 or type 8 parameters
Obtain a sort key value.	Control break Subscripted work fields on type 6 or type 8 parameters
Reference repeating groups.	Explicit, implied, or zero subscripts
Reference multiply-occurring input fields.	Subscripts using GROUP REC and ELMNT REC
Access file definitions stored in the data dictionary.	DATABASE DICTNAME= FN=
Override parameters generated by the data dictionary.	IN REC GROUP REC/ELMNT REC
Match files.	IN parameter with MK=option
Insert the current value of a field into a heading.	Type 4 parameter Control breaks SORT/NOSORT B HEAD

Task	CA-Culprit instruction
Create nonprinted output.	OUT parameter with optional descriptors
	Format codes: FB, FU, FW, FP, or FZ
	DD=SYS020
Create standard files from the database.	DATABASE DICTNAME= IN DB SS= PATH DD=SYS020 (delete SYS010)
Copy the contents of a data file.	=MACRO AMLIST
Copy stored code.	USE =MACRO/=MEND =COPY
Change inline or copied code.	USE WITH VALUES CHANGE DROP/KEEP RENUMBER =MACRO/=MEND =DROP =CHANGE =COPY RPTNO= INPUT=
Assign default values to symbolic parameters.	USE
Nest copied code.	USE */ END

F.4 Summary of parameters for using database data

Summary of parameters for using database data

Function	What CA-Culprit needs	Use
Access the database.	Data dictionary name (if not primary dictionary)	DATABASE DICTNAME= IN DB
	Data source	SS=
	Subschema used	
Navigate the database.	Route through the database	PATH
Determine the database route used.	Keyword	PATH-ID
Use database information as a standard file.	Predefined logical records	PATH logical-record-name
Select logical records.	Conditional statements	WHERE
Search for defined values.	Comparison statements	CONTAINS MATCHES

F.5 Summary of advanced capabilities for accessing the database

Summary of advanced capabilities for accessing the database

TASK	CA-Culprit instruction
Retrieve partial information.	Alternate path
Select record types by record name.	SELECT/BYPASS
Direct record retrieval by key value.	KEY
Use a key file to retrieve records.	KEYFILE
Test for record occurrences.	PATH-- DB-EXIT facility
Access more than one relationship between records.	PATH-- DB-EXIT facility
Retrieve all occurrences of a set.	PATH-- DB-EXIT facility
Access multiple-member sets.	PATH parameter ALL-MEMBERS option

F.6 Minimum coding requirements for using data tables

Minimum input and output coding requirements for using data tables

Task	CA-Culprit parameter/keywords
Identify the source of conventional file input to create a table.	INPUT
Define fields from non-IDMS files to CA-Culprit.	REC
Define printed output.	OUTPUT
Define an output data table.	OUTPUT TABLE= TYPE=CREATE USER= PW= CATALOG=
Name the columns in a details-only table.	Type 5 (at least one)
Name the columns in a totals-only table.	Type 6
Retrieve a data table (copy a view).	INPUT TABLE= TYPE=COPY USER= PW= CATALOG=
Store additional rows in a previously defined data table.	OUTPUT TABLE= TYPE=ADD USER= PW= CATALOG=
Remove all rows and place new rows in a previously stored data table.	OUTPUT TABLE= TYPE=REPLACE USER= PW= CATALOG=
Delete a previously stored data table and all associated data.	OUTPUT TABLE= TYPE=DELETE USER= PW= CATALOG=
Modify the table definition of an existing table.	OUTPUT TABLE= TYPE=GENERATE USER= PW= CATALOG=

F.7 Minimum coding requirements for using SQL tables

Minimum input and output coding requirements for using SQL tables

Task	CA-Culprit parameter/keywords
Identify the source of conventional file input to create an SQL table.	INPUT
Define fields from non-IDMS files to CA-Culprit.	REC
Define printed output.	OUTPUT
Identify the input source as an SQL table; specify table location.	IN DB(Q); DICTIONARY= SCHEMA=
Define the SQL query to CA-Culprit.	SQL select-statement
Define an alias for an SQL column within the SQL query.	AS
Define an output data table.	OUTPUT SQLTABLE= TYPE=CREATE DICTIONARY= SCHEMA=
Name the columns in a details-only table.	Type 5 (at least one)
Name the columns in a totals-only table.	Type 6
Indicate an SQL column cannot contain null values	NOT NULL
Define a NULL indicator for SQL columns that are allowed to be NULL.	FB SZ=4
Store additional rows in a previously defined SQL table.	OUTPUT SQLTABLE= TYPE=ADD DICTIONARY= SCHEMA=
Remove all rows and place new rows in a previously stored SQL table.	OUTPUT SQLTABLE= TYPE=REPLACE DICTIONARY= SCHEMA=
Delete a previously stored SQL table and all associated data.	OUTPUT SQLTABLE= TYPE=DROP DICTIONARY= SCHEMA=

Appendix G. How Totals Processing Works

G.1	The extract processing phase (CULL)	G-6
G.2	The output phase (CULE)	G-7
G.3	The processing steps when an extract file is read	G-8

CA-Culprit processes data in two input/output phases:

1. The **extract phase (CULL)**:
 - a. Executes type 7 logic against the input data.
 - b. Outputs a temporary work file, called the **extract file**.
2. The **output phase (CULE)**:
 - a. Reads the extract file *after it has been sorted*.
 - b. Outputs the report.

When CA-Culprit sends the detail report records to the printer, it looks for control breaks and executes type 8 logic.

Code for Sample Totals-Processing report: This is a typical CA-Culprit run where sorting and control breaks are processed by employee status within department. At each control break the average starting age for employees is calculated in type 8 logic.

The code

```
IN 200
REC LAST-NAME      15   10
REC STATUS         82   2
REC START-YEAR    97   2   2
REC BIRTH-YEAR    109   2   2
REC DEPARTMENT    115   25
01SORT DEPARTMENT - STATUS 0      $Fields are written to the extract file
013 AVERAGE START AGE BY DEPARTMENT AND STATUS
010 TOTAL-MESSAGE '
010 EMPLOYEE-COUNT 1
010 AGE
010 AVERAGE-AGE
0151*000 EMPLOYEE-COUNT           $Type 5 lines build the
0151*010 DEPARTMENT   HH 'DEPARTMENT'   $extract file with the
0151*020 LAST-NAME     HH 'EMPLOYEE NAME' $current values of each
0151*030 STATUS        HH 'STATUS' 'CODE' $field
0151*040 START-YEAR SZ=5 HH 'START' ' YEAR'
0151*050 BIRTH-YEAR SZ=5 HH 'BIRTH' '  YEAR' $Type 5 lines also
0151*055 AGE SZ=2       HH 'AGE'           $format the print line
0161*045-TOTAL-MESSAGE
0161*050 EMPLOYEE-COUNT
0162*0450'AVERAGE AGE'
0162*055 AVERAGE-AGE SZ=2
017010 START-YEAR - BIRTH-YEAR AGE   $Writes the value to the extract file
018010 IF LEVL EQ 1 100             $Type 8 logic executes when control
018020 IF LEVL EQ 2 200             $breaks are encountered
018030 IF LEVL EQ 3 300
018100 MOVE 'NUMBER OF EMPLOYEES BY STATUS CODE:' TO TOTAL-MESSAGE
018110 B 400
018200 MOVE 'NUMBER OF EMPLOYEES BY DEPARTMENT:' TO TOTAL-MESSAGE
018210 B 400
018300 MOVE 'NUMBER OF EMPLOYEES FOR THE COMPANY:' TO TOTAL-MESSAGE
018310 B 400
018400 COMPUTE AGE / EMPLOYEE-COUNT AVERAGE-AGE   $Uses extract file data
018     TAKE
```

REPORT NO. 01		AVERAGE START AGE BY DEPARTMENT AND STATUS			09/19/99	PAGE	1
DEPARTMENT	EMPLOYEE NAME	STATUS CODE	START YEAR		BIRTH YEAR	AGE	
ACCOUNTING AND PAYROLL	BLOOMER	01	80		60	20	
ACCOUNTING AND PAYROLL	HUTTON	01	77		41	36	
ACCOUNTING AND PAYROLL	JENSON	01	80		48	32	
ACCOUNTING AND PAYROLL	KIMBALL	01	78		49	29	
ACCOUNTING AND PAYROLL	KING	01	80		60	20	
ACCOUNTING AND PAYROLL	NICEMAN	01	80		55	25	
				NUMBER OF EMPLOYEES BY STATUS CODE:		6	
				AVERAGE AGE		27	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		6	
				AVERAGE AGE		27	
BLUE SKIES	CLOUD	01	77		45	32	
BLUE SKIES	DONOVAN	01	81		51	30	
BLUE SKIES	MOON	01	78		44	34	
				NUMBER OF EMPLOYEES BY STATUS CODE:		3	
				AVERAGE AGE		32	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		3	
				AVERAGE AGE		32	
BRAINSTORMING	BREEZE	01	79		34	45	
BRAINSTORMING	CROW	01	79		44	35	
BRAINSTORMING	LANCHESTER	01	75		32	43	
BRAINSTORMING	MAKER	01	78		45	33	
BRAINSTORMING	MUNYON	01	80		50	30	
BRAINSTORMING	WAGNER	01	78		34	44	
				NUMBER OF EMPLOYEES BY STATUS CODE:		6	
				AVERAGE AGE		38	
BRAINSTORMING	ANDALE	03	78		60	18	
				NUMBER OF EMPLOYEES BY STATUS CODE:		1	
				AVERAGE AGE		18	
BRAINSTORMING	ARM	05	77		34	43	

REPORT NO. 01		AVERAGE START AGE BY DEPARTMENT AND STATUS			09/19/99	PAGE	2
DEPARTMENT	EMPLOYEE NAME	STATUS CODE	START YEAR		BIRTH YEAR	AGE	
				NUMBER OF EMPLOYEES BY STATUS CODE:		1	
				AVERAGE AGE		43	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		8	
				AVERAGE AGE		36	
COMPUTER OPERATIONS	CRANE	01	77		42	35	
COMPUTER OPERATIONS	FERNDALE	01	79		58	21	
COMPUTER OPERATIONS	FONRAD	01	80		50	30	
COMPUTER OPERATIONS	GARDNER	01	81		59	22	
COMPUTER OPERATIONS	KLWELLEN	01	78		55	23	
COMPUTER OPERATIONS	KRAAMER	01	81		54	27	
COMPUTER OPERATIONS	LIPSICH	01	81		53	28	
				NUMBER OF EMPLOYEES BY STATUS CODE:		7	
				AVERAGE AGE		27	
COMPUTER OPERATIONS	TERNER	02	82		60	22	
				NUMBER OF EMPLOYEES BY STATUS CODE:		1	
				AVERAGE AGE		22	
COMPUTER OPERATIONS	KAHALLY	05	79		52	27	
				NUMBER OF EMPLOYEES BY STATUS CODE:		1	
				AVERAGE AGE		27	
				NUMBER OF EMPLOYEES BY DEPARTMENT:		9	
				AVERAGE AGE		26	
EXECUTIVE ADMINISTRATION	HENDON	01	73		33	40	
EXECUTIVE ADMINISTRATION	PAPAZEUS	01	78		35	43	
EXECUTIVE ADMINISTRATION	RUPEE	01	75		33	42	
EXECUTIVE ADMINISTRATION	WILDER	01	79		55	24	
				NUMBER OF EMPLOYEES BY STATUS CODE:		4	
				AVERAGE AGE		37	

REPORT NO. 01				AVERAGE STATUS	START	AGE BY DEPARTMENT AND STATUS	09/19/99	PAGE	3
DEPARTMENT	EMPLOYEE NAME	CODE	YEAR				BIRTH YEAR	AGE	
NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE								4	37
51 25 47 34 45 32 58 22 56 25 40 41 48 34 55 25 54 24 55 25									
INTERNAL SOFTWARE	DOUGH	01	76						
INTERNAL SOFTWARE	GALLWAY	01	81						
INTERNAL SOFTWARE	GARFIELD	01	77						
INTERNAL SOFTWARE	GRANGER	01	80						
INTERNAL SOFTWARE	HEAROWITZ	01	81						
INTERNAL SOFTWARE	JACOBI	01	81						
INTERNAL SOFTWARE	JENSEN	01	82						
INTERNAL SOFTWARE	LITERATA	01	80						
INTERNAL SOFTWARE	O'HEARN	01	78						
INTERNAL SOFTWARE	TYRO	01	80						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								10	29
56 25 51 29 49 32									
NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE								10	29
PERSONNEL	FITZHUGH	01	81						
PERSONNEL	ORGRATZI	01	80						
PERSONNEL	PEOPLES	01	81						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								3	29
PERSONNEL	JOHNSON	05	77						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								45	32
1 32									
NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE								4	30
PUBLIC RELATIONS	ANGELO	01	79						
PUBLIC RELATIONS	BANK	01	78						
PUBLIC RELATIONS	BOWER	01	77						
PUBLIC RELATIONS	JACKSON	01	77						

REPORT NO. 01				AVERAGE STATUS	START	AGE BY DEPARTMENT AND STATUS	09/19/99	PAGE	4
DEPARTMENT	EMPLOYEE NAME	CODE	YEAR				BIRTH YEAR	AGE	
PUBLIC RELATIONS	MCDOUGALL	01	80						
PUBLIC RELATIONS	PENMAN	01	77						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								6	28
PUBLIC RELATIONS	ZEDI	05	76						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								40	36
1 36									
NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE								7	29
THERMOREGULATION	CLOTH	01	79						
THERMOREGULATION	FINN	01	79						
THERMOREGULATION	TIME	01	81						
THERMOREGULATION	WILCO	01	79						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								4	32
THERMOREGULATION	KASPAR	04	80						
NUMBER OF EMPLOYEES BY STATUS CODE: AVERAGE AGE								40	40
1 40									
NUMBER OF EMPLOYEES BY DEPARTMENT: AVERAGE AGE								5	33
56 33									
NUMBER OF EMPLOYEES FOR THE COMPANY: AVERAGE AGE								56	31

It is easier to understand how subtotaling and control break processing works if we look at the role each parameter plays in the sample report.

G.1 The extract processing phase (CULL)

- A single type 7 logic statement and an automatic TAKE combine to calculate an employee's age.

When the implied TAKE is executed in type 7 logic, a record consisting of the two fields on the SORT parameter and the seven fields on the type 5 parameters is written to the extract file.

- Type 5 lines perform two functions, in this order:
 1. Build an extract record, which consists of all the fields on the type 5 parameters and all of the fields on the SORT parameter.
 2. Establish the format of a detail line on the report from the size and auto-header code included on the line.

Because there is no DROP or SELECTION logic, the processing of every record from the input file results in each record being written to the extract file. No sorting or totaling is done by CA-Culprit during this input processing; only the instructions coded in type 7 logic are executed.

When CA-Culprit is through processing the input file, the sort specified on the SORT parameter is executed by the installation's sort utility. The extract file is sorted in the order of the sort keys, which are part of each extract record. The extract file is then ready for output processing.

G.2 The output phase (CULE)

- The sorted extract file records provide the input for control breaks and printing the report.
- The only fields from the extract record used in the type 8 logic or on the type 6 lines are those that are coded on the SORT parameter or on the type 5 lines. Work fields that are not part of the extract record are also available for use.
- Type 8 logic is executed whenever a control break is encountered on STATUS or DEPARTMENT.
- The TAKE after sequence 400 sends the type 6 line to be printed.
- One subtotal accumulator (bucket) is set up for each field at each break level for numeric fields named on type 5 lines and referenced in type 8 logic or on type 6 parameters (AGE and EMPLOYEE-COUNT). No other fields are automatically subtotalized.
- There are three control break levels:

LEVL 1 - STATUS

LEVL 2 - DEPARTMENT

LEVL 3 - GRAND TOTAL

- AGE and EMPLOYEE-COUNT are defined as work fields, and used as such in type 7 logic., are not used In type 8 logic, AGE and EMPLOYEE-COUNT are not used as work fields because they are part of the extract file record in the output processing phase. (Note that AVERAGE-AGE could be used as a work field in both type 7 and type 8 logic because it is not part of the extract record.)

G.3 The processing steps when an extract file is read

CA-Culprit performs the following steps when an extract file is read:

1. Sends each detail extract record to the printer. Grand total break processing is performed at the end-of-file.
2. Updates all subtotal accumulators.

In our example, the detail values for AGE and EMPLOYEE-COUNT are added to all three accumulators. Since none of the other numeric fields from the type 5 line are referenced in type 6 or type 8 code, no other automatic totaling is performed. EMPLOYEE-COUNT serves as a counter because it always has a value of 1 on the type 5 line. It is not printed on the detail line of the report because position *000 (0000 will work too) is specified on the type 5 line, but it is still subtotalled.

3. Looks ahead to the next detail extract record, searching its sort keys for a control break. If there is no control break, then it processes that next detail extract record (step 1 above).

If there is a control break:

- CA-Culprit holds that next detail extract record aside and executes type 8 logic for the appropriate control break level.
- TAKE in type 8 logic causes all of the type 6 lines to be sent to the printer.

If a control break occurs on more than one level at the same time, CA-Culprit executes type 8 logic once for each level starting with the lowest level number first.

In our example, when DEPARTMENT breaks, STATUS always has to break because it is a LEVL 1. Type 8 logic is executed for STATUS first.

4. Executes control break spacing after exiting type 8 logic because of a TAKE or a DROP.

If multiple control breaks occur, such as when DEPARTMENT and STATUS break together, the control break spacing is only performed for the highest level.

5. Moves 0 to the accumulator for each field being totaled at the control break level being processed.
6. Continues with step 1 (above) for the next record read from the extract file.
7. Looks again to the next detail record from the extract file.
8. Moves the values of any sort key fields on the record to an internal header control storage area if any of the sort key fields are coded on a type 4 parameter.

Index

Special Characters

=COPY 12-18
=MACRO */=MEND 12-20
=MACRO/=MEND 12-20, 12-24

A

accessing multiple-member sets 14-24
ADD 8-15
ALL MEMBERS 14-24
alphanumeric data 1-9
alternate path 14-5
alternate path identifier 14-5
 alternate 14-5
 identifier 14-5
 null 14-5
AMLIST routine 12-24
AMLIST3 12-24
AND 4-10
area name field 8-5
AREA= 8-6
arithmetic operations 5-9—5-11
ascending order 4-8
ASF 8-13
 See also tables
 area name field 8-6
 comment field 8-5
 dialogs 8-6
 maps 8-6
Automatic Service Facility
 See ASF

B

bill-of-materials 14-21
binary data 1-9, 13-4
bit 1-9
block size 1-8
buffer dumps B-5, C-8
bytes 1-8

C

CA-Culprit
 capabilities of 1-3
 processing steps G-8
 user modules D-3

CA-IDMS/DB 8-8, 14-4, 14-21
 accessing multiple-member sets 14-24
 bill-of-materials 14-21
 CALC key 14-10
 db-key 14-10
 IDMS-STATUS 14-14
 index key 14-10
 INPUT (IN) DB 14-4
 LR-STATUS 14-14
 null path 14-19
 retrieval by key 14-10
 retrieve all occurrences 14-16
 retrieving partial paths 14-5
 retrieving selected records 14-7
 retrieving stand-alone records 14-14
 SELECT/BYPASS 14-7
 testing record occurrences 14-19
 walking the set 14-16
CALC key 14-10
calculations on totals 9-4—9-5
card file 13-4
catalog
 identification of 8-4, 8-13, 8-24
 keyword 8-4
CATALOG= 8-4, 8-13, 8-24
 INPUT (IN) 8-13
CHANGE 12-11
CHANGE= 8-6
clauses
 DEFAULT 12-8
 DROP 12-14
 IN PATH 14-7
 KEEP 12-13
 RENUMBER 12-15
 WHERE 7-8, 8-23
 WITH VALUES 12-5, 12-8, 12-10, 12-24
code
 copying of 12-4
 debugging of 1-5, B-3—B-10
 execution of 1-5
 modified 12-11—12-23
coding
 See also column headings
 See also column size
 See also data formatting
 See also input definition
 See also page, formatting of
 line length 3-16

coding (*continued*)
 size 3-16
coding conventions 1-5
coding, how to
 alternate path 14-5
 bill-of-materials access 14-21
 column placement 2-6
 column width 3-10
 control break 6-13
 database record retrieval 14-4
 DB-EXIT facility 14-16
 DEFAULT clause 12-8
 efficiency of C-3—C-12
 levels of field names 14-21
 match-file runs 11-7
 multiple report 4-4
 multiple-member sets 14-24
 nested USE parameters 12-10
 nonprint reports 13-4
 OUTPUT (OUT) 13-4
 output definition 2-6
 report number 2-6
 required parameters 2-5
 row identifier 2-6
 spacing 2-6
 subscripts 10-4—10-23
coding, parameters
 =COPY parameter 12-18
 BYPASS 4-10
 BYPASS parameter 4-10
 INPUT DB parameter 7-4
 KEY parameter 14-10
 KEYFILE parameter 14-10
 NOSORT parameter 13-12
 OUTPUT (OUT) 6-9
 OUTPUT (OUT) parameter 6-9, 6-11
 SELECT 4-10
 SELECT BUFFER 11-3
 SELECT parameter 4-10, 11-3
 SORT 4-8, 6-13
 SORT parameter 4-8, 6-13
 type 3 parameter 3-4
 type 4 3-8
 type 4 parameter 3-8
 type 5 parameter 2-6
 type 6 6-5
 type 6 parameter 6-5, 6-7
 type 7 5-4
 type 7 parameter 5-4
 type 8 parameter 6-18
 USE parameter 12-4

column headings 3-6—3-8
 HF 3-6
 HH 3-6
 HR 3-6, 13-14
 multiple-line 3-7
 on type 4 parameters 3-7
 on type 5 parameters 3-6
 single line 3-6
column placement
 See coding
column size
 SZ= 3-10
column width modification 3-10—3-12
COLUMN= 8-5
columns
 definition of 8-4
 placement in large tables 8-5
comment field 8-5
COMMENT= 8-5
compare table fields 8-24
computations
 complex 5-9
 simple 5-9
COMPUTE 5-9
concatenate
 See table, consolidation of
concepts, data processing
 See data processing concepts
conditional statements 5-9
 See also clauses
 type 7 5-9
CONSOLIDATE 8-23
CONTAINS 7-8
control break 5-9, 6-13, 6-24, 9-3—9-14
 forced 9-10
 LEVL= 6-18, 9-4
 SORT 6-24, 9-4
 type 6 6-16, 9-4
 type 8 9-4
conventions, coding 1-5
conversions
 zoned decimal to packed decimal 13-6
copied code
 sequential parameter listing 12-18
COPY 8-13, 8-23
 INPUT (IN) 8-23
copy data tables 8-13
copying code 12-4—12-9, 12-18—12-23
 =COPY 12-18
 USE * 12-10

correcting errors
 See debugging
count, total of 5-8
counters 5-5—5-8
 as subscripts 10-6
 sequential count 5-6
 total of 5-7
 type 6 5-7
 type 7 5-9
CREATE 8-4
creating tables
 column definition 8-4
 definition of 8-4
 from an CA-IDMS/DB database 8-4, 8-8
 from an existing table 8-5, 8-10
 from ISAM files 8-4
 from sequential files 8-4, 8-6
 from VSAM files 8-4
 input definition parameters 8-4
 special options 8-5
 totals-only 8-12
CULE G-3, G-6
CULL G-3, G-6
 type 7 G-6
 type 8 G-6
CULLUS48 B-3, B-8
CVMACH= 8-24

D

data dictionary 7-3, 14-4
 non-database files 13-14
data formatting
 comma suppression 3-13
 dollar sign insertion 3-13
 in defined patterns 3-13
 numeric 3-13
 social security numbers 3-13
data processing concepts
 block size 1-8
 bytes 1-8
 fields 1-7
 file 1-8
 files 1-7
 records 1-7
 records, format of 1-8, 1-10
 records, size 1-8
 types of files 1-9
data structure 7-3
data table reports 8-4

data tables
 See tables
data types
 alphanumeric 1-9
 binary 1-9
 bit 1-9
 numeric 1-9
 unsigned packed decimal 1-9
 zoned decimal 1-9
data, binary 1-9, 13-4
database
 See CA-IDMS/DB
DATABASE parameter 7-4, 7-6, 8-4, 11-25
 DICTNAME= 13-14
database specification
 See DATABASE parameter
DB-EXIT facility 14-14, 14-16
 dummy 14-14
db-key 14-10
 logical record field 14-10
DD= 11-7
 REC 11-7
 SELECT/BYPASS 11-7
debugging 1-5, B-3—B-10
 See also input parameter listing
 See also run-time messages
 See also sequential parameter listing
 CULLUS48 B-8
 sequence of B-3
decimal place indicator 2-6
DEFAULT 12-8
default values
 See symbolic fields
define fields
 See REC parameter
define files
 See INPUT (IN) parameter
define table output 8-4
 DATABASE 8-4
 IN DB 8-4
 OUTPUT (OUT) 8-4
 REC 8-4
 type 5 8-4
 type 6 8-4
defining input 2-5
 INPUT (IN) 2-5
 REC 2-5
defining output 2-6
defining tables 8-4, 8-13, 8-23
DELETE 8-15

descending order 4-8
detail lines, multiple 4-13—4-14
 type 5 4-13
details only (OUT D) 5-9
details-only table 8-5
dialogs 8-6
dictionary
 See catalog
DICTNAME= 7-4
DISPLAY= 8-6
double precision data 13-4
DP= 2-6
DROP 6-18, 9-4, 12-14
DROP/KEEP 12-11

E

efficient coding C-3
SELECT/BYPASS C-5
 type 7 C-5
eliminate field occurrences 9-10
ELMNT 10-11—10-23
END 12-10
EQ 4-10
ERASE= 8-6
error correction
 See debugging
executing code 1-5
explicit subscripts 10-4
extract file G-8

F

F\$ 3-13
FB 13-4
FD 3-13
field name levels 14-21
fields 1-7
file description
 See also data types
 example of 2-4
 length 1-9
 start position 1-9
file matching
 buffer contents 11-3, 11-4
 BYPASS 11-10
 coding for 11-7
 data dictionary 11-25, 11-26
 DATABASE 11-25
 file status byte 11-5—11-6
 INPUT (IN) 11-25

file matching (*continued*)
JCL (OS/390) assignments 11-7, 11-27
M*ID 11-5—11-6
master file 11-7
match-key name 11-26
MB=KEEP 11-4, 11-10, 11-13, 11-15, 11-18
missing masters 11-18
missing transactions 11-15
multiple transactions 11-10, 11-13
qualified fields 11-27
REC 11-25, 11-26
SELECT 11-13
SELECT BUFFER 11-15, 11-18
single occurrence files 11-8
transaction file 11-7
VSAM file 11-25
 with data dictionary definitions 11-27

file name

See FN=

file status 11-5—11-6

files 1-7, 1-9

See also data processing concepts

=MACRO/=MEND 12-24

card file 13-4

indexed sequential file 13-4

prerun listing of 12-24

sequential file 13-4

USE 12-24

VSAM file 11-25, 13-4

fixed fields 10-11

floating fields 10-15

 REC 10-15

FM 3-13

FN 3-13

FN= 11-25, 11-27, 13-14, 14-11

 DATABASE 13-14

 REC 13-14

forcing control breaks 9-10

 SORT 9-10

 type 7 9-10

 type 8 9-10

formatting

See data formatting

formatting codes

 F\$ 3-13

 FB 13-4

 FD 3-13

 FM 3-13

 FN 3-13

 FP 13-4

 FS 3-13

formatting codes (*continued*)

 FU 13-4
 FW 13-4
 FZ 13-4

formatting numeric data 3-13—3-15

 FP 13-4
 FS 3-13
 FU 13-4

functions

See also table
 INPUT (IN) 8-15
 type 5 8-15
 type 7 8-15
 type 8 8-15

 FW 13-4
 FZ 13-4

G

GENERATE 8-15
GROUP 10-11—10-23
GT 4-10

H

headings, variable 13-11—13-13
HF 3-6
HH 3-6
HR 3-6

I

IDMS-STATUS 14-14
IN DB parameter 8-4
IN PATH 14-7
index key 14-10
indexed sequential file 13-4
indexed table rows 8-5
initializing tables 11-21
INPUT (IN) DB parameter 7-4, 7-6, 14-24
INPUT (IN) parameter 2-5, 8-13, 8-15, 8-23
 FN= 11-25, 11-27, 13-14
 MB=KEEP 11-4, 11-7
 MK= 11-7, 11-27
 suppression of 12-18
input buffer
 size, database retrievals 14-4
input data processing 5-4—5-14
input definition 2-5
input definition, table 8-4

INPUT parameter 11-3

input parameter listing 8-7, B-4

See also statistics

insert dollar signs 3-13

instructions

See also keywords
 COMPUTE 5-9

IX= 8-5

J

JCL

See Job Control Language

Job Control Language (JCL) C-11

See also file matching

file specifications C-11

function of 1-5

non-print specification 13-4

SYS010 11-27

SYS011 11-27

SYS020 13-4

K

KEEP 12-13
KEY parameter 14-10
KEYFILE parameter 14-10, 14-24
keywords
 ALL MEMBERS 14-24
 AREA= 8-6
 CATALOG= 8-4, 8-13, 8-24
 CHANGE= 8-6
 CONTAINS 7-8
 CVMACH= 8-24
 DICTNAME= 7-4
 DISPLAY= 8-6
 DROP 6-18, 9-4
 ELMNT 10-11, 10-13, 10-15
 ERASE= 8-6
 FN= 14-11
 GROUP 10-11, 10-13, 10-15
 KF= 14-10
 LOAD= 8-6
 LOCATION= 8-24
 LRFNAME= 14-11
 MATCHES 7-8
 MB=KEEP 11-4
 MOVE 6-22
 ONLINE= 8-6
 OWNER= 8-24
 PERFORM/RETURN 5-12

keywords (*continued*)

PW= 8-4, 8-13
required 8-4
RPTNO= 12-18
SS= 7-4
SYSCTL= 8-24
TABLE= 8-4, 8-13, 8-24
TAKE 6-18, 6-22, 9-4, 9-6
TYPE= 8-4, 8-13, 8-24
USER= 8-4, 8-13, 8-24
VALIDATE= 8-24
WHERE 8-24

keywords, table
optional 8-5, 8-6, 8-24
required 8-13, 8-24

KF= 14-10
REC 14-11
SELECT/BYPASS 14-11

L

LEVEL= 6-18, 9-4
type 6 6-21
type 8 6-21

listing a standard file 2-3—2-8

listing database occurrences 7-4—7-5
DATABASE 7-4
INPUT (IN) DB 7-4
REC 7-4
type 5 7-4

listing, input parameter 8-7

literal subscripts 10-4

LOAD= 8-6

loading tables
See creating tables

locating data 2-4
INPUT (IN) 2-5
REC 2-5
required parameters 2-5
type 5 2-5

LOCATION= 8-24

logical operations on totals 9-4—9-5

logical operators
AND 4-10
OR 4-10

logical records 7-6, 14-10
DATABASE 7-6
INPUT (IN) DB 7-6
KEY 14-10
KEYFILE 14-10
LR-STATUS 14-14

logical records (*continued*)

LFNAME= 14-11
type 5 7-6
LR-STATUS 14-14
LFNAME= 14-11
LT 4-10

M

M*ID 11-5—11-6
maps 8-6
master file 11-7
match-file facility
See also file matching
table initialization 11-21

match-key name
See file matching

MATCHES 7-8

MB=KEEP 11-4, 11-7, 11-10, 11-13, 11-15, 11-18
INPUT (IN) 11-7

messages, run-time 8-8, 8-27

MK= 11-7, 11-27

modified code
sequential parameter listing 12-12, 12-13, 12-16, 12-21

modify tables 8-15

modifying code 12-11—12-23

MOVE 6-22—6-23

multiple reports 4-4—4-7

multiple-member sets
coding of 14-24

INPUT (IN) DB 14-24

KEYFILE 14-24

N

non-database files 13-14

nonprint reports 13-4—13-10

NOSORT parameter 13-12, C-9
NOSORT C-9
SORT C-9

null path 14-19

numeric data 1-9
formatting of 3-13

O

ONLINE= 8-6

OR 4-10

OUTPUT (OUT) parameter 3-16, 6-9, 6-11, 8-4, 8-24, 13-4

OUTPUT (OUT) 3-16, 6-11

output definition
 See coding
output processing 6-3—6-23
 sequencing report output 4-4
override cv machine name 8-24
override DICTNODE= 8-24
override SYSCTL= 8-24
owner identification 8-24
OWNER= 8-24

P

PERFORM/RETURN 5-12
processing flow 5-12—5-14
processing phases
 CULE G-3, G-6
 CULL G-3, G-6
 input 5-9, G-3, G-6
 output G-3, G-6
 output phase B-6
 output processing 6-3—6-23
PW= 8-4, 8-13

Q

qualified fields
 See file matching

R

REC parameter 2-5, 7-4, 8-4, 10-15, 11-7, 11-25, 11-26,
 13-14, 14-11
record format 1-10
records 1-7
 format of 1-8
 record occurrences 14-19
 size 1-8
 variable length 13-4, 13-9
redefining adjacent storage 10-8
referencing field occurrences 10-4—10-23
RENUMBER 12-11, 12-15
repeated procedures 5-12
repeating field 10-4
REPLACE 8-15
replace table data 8-15
report
 details-only 6-11
 multiple 4-4
 nonprinted 13-4
 number, coding of 2-6
 planning of 1-4
 totals-only 6-9

report (*continued*)
 writing, steps of 2-3
required parameters 2-5
reserved words, tables
 SUBSCHEMA-NAME 8-25
 TABLE-ID 8-25
 TABLE-NAME 8-25
retrieve
 See also CA-IDMS/DB
 data tables 8-13
 subschema name 8-25
 table ID 8-25
 table name 8-25
 retrieving logical records 7-6—7-7
row identifier 2-6
rows, table
 storage of 8-5
RPTNO= 12-18
run-time messages B-5
 See also statistics
 extraction statistics 8-27
 update statistics 8-8

S

SELECT 11-3
SELECT BUFFER 11-3, 11-15, 11-18
SELECT/BYPASS parameter 4-10, 8-23, 11-7, 11-10,
 11-13, 14-7, 14-11, C-5
 SELECT/BYPASS 11-10, 11-13
 type 7 parameter C-5
selected record processing 4-10—4-12
selecting data 4-10, 7-8
selection criteria
 See test criteria
sequence numbers
 renumbering 12-11
sequencing report output 4-8—4-9
sequential files 13-4
sequential parameter listing B-4
 =COPY 12-18
 =MACRO * 12-21
 =MACRO */=MEND 12-20
 =MACRO/=MEND 12-20
 AMLIST3 12-25
 AMLIST3, modified report 12-26
CHANGE 12-12
copied code 12-18
copied lines 12-22
KEEP clause 12-13
modified code 12-21

sequential parameter listing (*continued*)
 RENUMBER clause 12-16
 symbolic fields 12-6
 value substitution 12-22
 single precision data 13-4
SIZE= 8-5
 social security number, formatting of 3-13
 sort key values 6-24—6-25, 6-26—6-28
 alphanumeric 6-26
 numeric 6-24
 sort keys
 subscripted 10-22
 values of 10-22
SORT parameter 4-8, 6-24, 6-26, 9-4, 9-6, 9-10, 10-22,
 13-11, C-9
 SORT 10-22
 type 6 10-22
 type 8 10-22
 sorting 4-8
 spacing 2-6
 type 5 2-6
SS= 7-4
 stand alone records 14-14
 statistics
 See also run-time messages
 output data table 8-7
 status
 See also file status
 INPUT 11-3
STOP-RUN 14-14
 subschema 7-3, 14-4
SUBSCHEMA-NAME 8-25
 subscripts
 ELMNT 10-11, 10-13, 10-15
 explicit 10-4, 10-13
 fixed fields 10-11
 floating fields 10-15
 GROUP 10-11, 10-13, 10-15
 literal 10-4
 occurrence total 10-17
 specific field values 10-19
 variable fields 10-13
 zero 10-8
 subtitles 3-8—3-10
 on type 4 parameters 3-8
 subtotals 6-13
 accumulator initialization 9-7
 multiple level 9-6
 SORT 9-6
 type 6 9-6
 type 8 9-6
 suppressing commas 3-13
 symbolic fields 12-5
 default values 12-8
 sequential parameter listing 12-6
 symbolic fields, use of 12-5
 symbolic parameters 12-20
 See also symbolic fields
 SYS010 11-27
 SYS011 11-27
 stored 12-3
 USE 12-3
 SYS020 13-4
 SYSCTL= 8-24
 SZ= 3-10

T

TABLE-ID 8-25
TABLE-NAME 8-25
TABLE= 8-4, 8-13, 8-24
 tables 8-23
 See also copying tables
 See also creating tables
 See also modifying tables
 See also retrieving tables
 ADD function 8-15—8-17
 ASF 8-5, 8-6, 8-13
 column placement 8-5
 comment field 8-5
 CONSOLIDATE function 8-23—8-27
 consolidation of 8-23
 COPY function 8-13—8-14, 8-23
 CREATE function 8-4—8-12
 data comparison 8-24
 data selection 8-24
 DATABASE parameter 8-4
 defining columns 8-4
 defining input data 8-4, 8-15, 8-23
 defining output 8-4, 8-23
 definition of 8-4
 DELETE function 8-15, 8-18—8-19
 extraction statistics 8-27
 field comparison 8-24
 GENERATE function 8-15, 8-19—8-22
 IN DB parameter 8-4
 INPUT (IN) parameter 8-15, 8-23
 logical record (LR) path 8-6
 modification of 8-15
 OUTPUT (OUT) 8-24
 OUTPUT (OUT) parameter 8-4, 8-24
 output data table 8-7

-
- tables (*continued*)
- REC parameter 8-4
 - regeneration of 8-15
 - REPLACE function 8-15, 8-17—8-18
 - reports, data table 8-4
 - reserved words 8-25
 - retrieval of 8-13
 - row size 8-5
 - row storage 8-5
 - SELECT/BYPASS 8-23
 - totals-only 8-5
 - type 5 8-24
 - type 5 parameter 8-4, 8-15, 8-24
 - type 6 8-24
 - type 6 parameter 8-4, 8-24
 - type 7 8-24
 - type 7 parameter 8-15, 8-24
 - type 8 8-24
 - type 8 parameter 8-15, 8-24
 - update statistics 8-7
 - WHERE 8-23
 - TAKE 6-18, 6-22—6-23, 9-4, 9-6
 - test criteria 4-10
 - multiple 4-11
 - single 4-10
 - testing, record occurrences 14-19
 - titles 3-4—3-5
 - total line placement 6-16
 - SORT 6-18
 - type 6 6-18
 - type 8 6-18
 - totals
 - calculations 9-4
 - labeling of 6-7
 - logical operations 9-4
 - of subscripted fields 10-17
 - printing 6-5, 6-16
 - selective printing of 6-18
 - suppression of 6-11
 - totals only 6-9
 - type 5 10-17
 - type 6 10-17
 - totals-only table 8-12
 - transaction file 11-7
 - type 3 parameter 3-4
 - type 3 3-4
 - type 4 parameter 3-8, 13-11
 - type 5 parameter 2-6, 4-13, 7-4, 7-6, 8-4, 8-15, 8-24, 10-17
 - type 6 parameter 6-5, 6-7, 6-16, 6-18, 6-21, 8-4, 8-24, 9-4, 9-6, 10-17, 10-19, 10-22
- type 6 parameter (*continued*)
- type 6 6-7
 - type 7 parameter 5-4, 8-15, 8-24, 9-10, G-6
 - RENUMBER 12-11
 - type 7 processing 5-4—5-14
 - type 8 parameter 6-18, 6-21, 8-15, 8-24, 9-4, 9-6, 9-10, 10-19, 10-22, G-6
 - RENUMBER 12-11
 - USE 12-11
 - type 8 processing G-6
 - TYPE= 8-4, 8-13, 8-24
 - types of data
 - See* data types
- U**
- unsigned packed decimal data 1-9
 - updating tables
 - See* tables
 - USE * 12-10
 - USE parameter 12-3, 12-4, 12-5
 - assigning values 12-5
 - CHANGE 12-11
 - DEFAULT 12-8
 - DROP/KEEP 12-11
 - END 12-10
 - nesting of 12-10
 - RENUMBER 12-11
 - USE * 12-10
 - with symbolic fields 12-5
 - WITH VALUES 12-5, 12-8, 12-10
 - user identification 8-4, 8-13, 8-24
 - user modules D-3
 - basic F-3
 - for data tables F-9
 - for database access F-9
 - for database data F-7
 - for standard files F-5
 - processing F-4
 - summary of F-3
 - USER= 8-4, 8-13, 8-24
- V**
- VALIDATE= 8-24
 - variable fields 10-13
 - variable headings 13-11—13-13
 - SORT 13-11
 - type 4 13-11
 - VM/ESA
 - See* CVMACH=

VSAM files 11-25, 13-4

W

walking the set 14-16
 null 14-19
WHERE clause 7-8, 8-24
WITH VALUES 12-8, 12-10, 12-24
work field 5-5
 current value of 9-13
 numeric, subscripted 10-19
 type 6 10-19
 type 8 10-19
work field values, sorted 6-26
 SORT 6-26
writing database files 13-10
writing out an entire record 13-8
writing to a sequential file 13-5, 13-8
writing variable-length records 13-9

Z

zero subscripting 10-8
zoned decimal data 1-9, 13-4

