CA-Easytrieve®

Plus

For OS/390, VM, VSE

Application Guide

6.2
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.


All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.
Chapter 1: About This Guide

Purpose and Audience ................................................................. 1–1
Organization .............................................................................. 1–1
Other CA-Easytrieve/Plus Publications ........................................ 1–3
Related Publications .................................................................. 1–4
Documentation Conventions ...................................................... 1–4
    Variable Parameters ............................................................... 1–5

Chapter 2: Overview

Introduction ................................................................................ 2–1
    Capabilities .......................................................................... 2–1
    Application ........................................................................... 2–2
Structure .................................................................................... 2–7
    Environment Definition Section ............................................ 2–8
    Library Section ...................................................................... 2–8
    Activity Definition Section .................................................. 2–8
Rules of Syntax ............................................................................ 2–10
    Statement Structure ............................................................. 2–10
    Words .................................................................................. 2–11
    Comments ............................................................................ 2–12
    Continuations ....................................................................... 2–13
    Environment Definition ....................................................... 2–13
    PARM Statement .................................................................. 2–13
    SYNTAX Parameter ............................................................. 2–14
    COMPILE Parameter .......................................................... 2–14
Chapter 3: Library

Introduction ................................................................. 3–1
FILE Statement .......................................................... 3–2
  File-type Parameters ................................................. 3–4
  Device-type Parameters ............................................ 3–5
  Record Format Parameters ....................................... 3–6
DEFINE Statement ....................................................... 3–8
  Field-name Parameter ............................................... 3–9
  Location Parameter ................................................... 3–9
  Attributes Parameter ............................................... 3–10
  MASK Parameter ..................................................... 3–12
  VALUE Parameter ..................................................... 3–14

Chapter 4: Activity Definition

Introduction ................................................................. 4–1
JOB Statement ........................................................... 4–3
SORT Statement .......................................................... 4–6
  SELECT Statement ..................................................... 4–7

Chapter 5: Data Manipulation

Assignment Statement ................................................... 5–1
  Equivalence .............................................................. 5–1
  Arithmetic Expression ................................................. 5–2

Chapter 6: Decision and Branching Logic

Introduction ................................................................. 6–1
  Conditional Expressions ............................................. 6–3
Field Relational Condition ............................................. 6–5
Field Class Condition ................................................... 6–6
Field Series Condition ................................................... 6–7
Field Presence Condition ............................................. 6–8
Field Presence Series Condition ..................................... 6–8
Record Relational Condition ........................................... 6–9
IF, ELSE, and END-IF Statements ........................................ 6–9
    IF Statement .................................................................... 6–9
    ELSE Statement ............................................................. 6–10
    END-IF Statement .......................................................... 6–10
    Nesting IF Statements .................................................... 6–10
DO and END-DO Statements ............................................... 6–11
    DO Statement ................................................................... 6–11
    END-DO Statement ......................................................... 6–11
    Nesting DO Loops ............................................................ 6–12
GOTO Statement ............................................................... 6–12
Statement Labels ............................................................... 6–13
Procedure Processing ....................................................... 6–14
    PROC and END-PROC Statements ................................. 6–14
    PERFORM Statement ..................................................... 6–15
STOP Statement ............................................................... 6–15
EXECUTE .......................................................................... 6–15

Chapter 7: Input/Output Specification

Introduction ........................................................................ 7–1
DISPLAY Statement ........................................................... 7–2
    Content and Spacing Parameters ..................................... 7–3
    Rules for Use ................................................................ 7–4
    Debugging .................................................................... 7–4
PRINT Statement ................................................................ 7–5
GET Statement ................................................................. 7–9
PUT Statement ................................................................. 7–9
    Example ....................................................................... 7–9
POINT Statement .............................................................. 7–10
READ Statement ............................................................... 7–11
WRITE Statement .............................................................. 7–12

Chapter 8: Report Processing

Introduction ........................................................................ 8–1
Report Types ...................................................................... 8–3
    Standard Reports .......................................................... 8–3
    Label Reports .............................................................. 8–4
REPORT Statement ........................................................... 8–5
SEQUENCE Statement ....................................................... 8–11
CONTROL Statement ......................................................... 8–11
Chapter 9: File Processing

Introduction .......................................................... 9–1
  Control of Input/Output (I/O) ........................................ 9–1
  Record Formats ..................................................... 9–2
  System-Defined Fields ............................................ 9–3
  Error Conditions .................................................... 9–3
  Data Availability Tests ........................................... 9–3
  Opening and Closing Files ...................................... 9–3
SAM Files .............................................................. 9–4
  Input .................................................................. 9–4
  Output ................................................................. 9–5
VFM Files .............................................................. 9–5
ISAM Files .............................................................. 9–5
  Sequential Processing ............................................. 9–6
  Skip-Sequential Processing ...................................... 9–6
  Random Processing .............................................. 9–7
VSAM Files ............................................................ 9–7
  File Loading ........................................................ 9–8
  Input .................................................................. 9–8
  Record Addition ................................................... 9–10
  Record Deletion ................................................... 9–10
  Record Update .................................................... 9–11
Synchronized File Processing .............................................................. 9–11
   Input ................................................................................................. 9–14
   Conditional Expressions ................................................................. 9–14
   File Presence Condition ................................................................. 9–15
   File Presence Series Condition ...................................................... 9–15
   Record Relational Condition ......................................................... 9–16

Chapter 10: Table Processing
Table Definition ............................................................................. 10–1
   Instream Tables .............................................................................. 10–2
   External Tables ............................................................................... 10–2
SEARCH Statement ......................................................................... 10–3

Chapter 11: IMS/DLI Processing
Introduction ..................................................................................... 11–1
FILE Statement ............................................................................... 11–2
RECORD Statement ....................................................................... 11–3
RETRIEVE Statement .................................................................... 11–4
Automatic Input with RETRIEVE ................................................. 11–6
   Sweep of a Database ..................................................................... 11–6
   Tickler File Control ..................................................................... 11–6
   Input Definition (Paths) .............................................................. 11–6

Chapter 12: OS/390 JCL
Introduction ..................................................................................... 12–1
Sample Short Report Output Program ............................................ 12–2
Mailing Label Output Program ....................................................... 12–3
Synchronized File Processing Program .......................................... 12–4
Compile and Link-Edit Load Module ............................................. 12–6
Previously Compiled and Link-Edited Programs ......................... 12–6
Chapter 13: VSE JCL

Introduction ................................................................................ 13–1
Sample Short Report Output Program ........................................... 13–2
Mailing Label Output Program ...................................................... 13–3
Synchronized File Processing Program ......................................... 13–5
Compile and Link-Edit Load Module ............................................ 13–7
Previously Compiled and Link-Edited Programs ......................... 13–7

Chapter 14: Applications

Introduction ................................................................................ 14–1
Application Overview ................................................................. 14–1
Program Formatting Standards .................................................... 14–2
Program Output Standards .......................................................... 14–2
Inventory Sample File ............................................................... 14–3
Personnel Sample File ............................................................... 14–4

Chapter 15: Basic Examples

Introduction ................................................................................ 15–1
Employees in Region 1 ................................................................ 15–2
Proposed Salary Schedules ......................................................... 15–3
Employee Letters ..................................................................... 15–5
Mailing Labels .......................................................................... 15–11
Tally Reports ............................................................................. 15–13
Women's Phone Numbers .......................................................... 15–17
Salary Tally Report ................................................................... 15–18
File Expansion .......................................................................... 15–19
Average Regional Gross Salary .................................................. 15–21
Central Region Employees .......................................................... 15–22
Inventory Report by City ............................................................ 15–26
Expanded Inventory Report ........................................................ 15–27
Error Correction ....................................................................... 15–30
Inventory Reduction .................................................................. 15–31
Inventory File Update ............................................................... 15–33
Reorder Notification Report ...................................................... 15–35
Chapter 16: Advanced Techniques

Introduction ................................................................. 16–1
Selected Control Break Processing .................................. 16–2
Summary File Processing .................................................. 16–3
Special Report Processing Exits ....................................... 16–5
Sorting Input Files .......................................................... 16–8
Synchronized File Facility - File Update ............................ 16–10
Reformat Printed Output from IDCAMS ............................. 16–12
VSAM File Processing ...................................................... 16–15
  Defining and Loading VSAM Data Sets with Alternate Indexes 16–16
  Load Base Clusters Through CA-Easytrieve/Plus .................... 16–17
  Defining and Building Alternate Indexes and Define Paths .......... 16–18
  Updating a VSAM KSDS Cluster ........................................ 16–19
  Sequentially Reading VSAM File through Non-unique Alternate Index 16–21
Updating a VSAM ESDS File .............................................. 16–22
  Deleting and Adding Records of VSAM KSDS File ................. 16–23
GETDATE Macro ............................................................. 16–25
CONCAT Macro .............................................................. 16–26
Processing JCL Parameters .............................................. 16–28

Chapter 17: Bank System

Introduction ................................................................. 17–1
Online Processing .......................................................... 17–2
  Initialize Customer File ................................................... 17–2
  BANKLIB Macro ........................................................... 17–6
  Bank File Program .......................................................... 17–7
Batch Processing .............................................................. 17–22
  Detail Report ................................................................. 17–22
  Mass Mailing ................................................................. 17–27
  Summary Report ........................................................... 17–32

Chapter 18: Project Management System

Introduction ................................................................. 18–1
  Master File Layout .......................................................... 18–1
Programs ..................................................................... 18–3
  File Maintenance ............................................................ 18–3
  Project Status - EXAMPLE 18.1 ........................................ 18–6
Output Reports ............................................................... 18–16
.File Update Reports ................................................................. 18–22
  Project Status - EXAMPLE 18.2 ........................................... 18–22
Report Generation ................................................................. 18–30
  Project Status - EXAMPLE 18.3 ........................................... 18–30
Project Summary ................................................................. 18–33
  Project Summary - EXAMPLE 18.4 ....................................... 18–33
  Project Completion - EXAMPLE 18.5 .................................. 18–34

Appendix A: Table of Statements
Functions ............................................................................... A–1

Appendix B: Cross-References
Cross-Reference of Statement ............................................... B–1

Index
Purpose and Audience

The purpose of this *CA-Easytrieve/Plus Application Guide* is to help you use CA-Easytrieve/Plus for report generation and file processing without requiring extensive data processing training and experience.

This guide covers a subset of CA-Easytrieve/Plus statements. The statements are described briefly, along with the associated parameters.

Examples of common uses of CA-Easytrieve/Plus in a variety of business applications are also presented. These examples include the required coding and illustrations of the output reports. If you want more information about any individual CA-Easytrieve/Plus statement, function, or operation, refer to the *CA-Easytrieve/Plus Reference Guide*.

The *CA-Easytrieve/Plus Application Guide* is written for the business-oriented professional. Using this guide enables you to manipulate files, and to design and print reports, without having to wait for available time from data processing personnel.

Organization

This guide is organized into four major divisions:

- Chapters 2 through 13 describe CA-Easytrieve/Plus statements, grouped by functional requirements, and report and file processing.
- Chapters 14 through 18 provide examples of report types which are common to many different businesses.
- Appendix A presents the CA-Easytrieve/Plus statements covered in this Guide in a condensed, tabular form for easy reference.
- Appendix B contains a cross reference of CA-Easytrieve/Plus statements by the examples described in Chapters 14 through 18.
This book is divided into several chapters:

- “Overview” introduces you to this information retrieval and data management system, designed to simplify typical programming to almost any business-oriented task.
- “Library” describes the data your program processes in terms of files, records, and fields.
- “Activity Definition” contains the statements to perform tasks you created, such as reading in, processing and writing out data.
- “Data Manipulation” shows you several ways to manipulate data within your program by using the Assignment statement.
- “Decision and Branching Logic” describes the statements that execute your program by means of decision and branching logic.
- “Input/Output Specification” describes three levels to control your I/O--automatic, controlled, and database.
- “Report Processing” features the ease with which CA-Easytrieve/Plus makes the task of producing reports.
- “File Processing” enables you to use CA-Easytrieve/Plus to process any existing file, to read it, change records within it, add new records, or delete existing records.
- “Table Processing” describes table processing through typical examples.
- “IMS/DLI Processing” explains the use of IMS/DLI, interfacing with CA-Easytrieve/Plus, to provide facilities for information retrieval from databases.
- “OS/390 JCL” provides some general information about OS/390 JCL requirements.
- “VSE JCL” provides some general information about VSE JCL requirements.
- “Applications” introduces Part II of this guide and offers a composite sample of CA-Easytrieve/Plus jobs which perform typical data processing functions.
- “Basic Examples” illustrates the use of CA-Easytrieve/Plus to solve a variety of basic data processing problems.
- “Advanced Techniques” provides examples some advanced processing techniques available in CA-Easytrieve/Plus.
- “Bank System” combines online and batch processing to illustrate CA-Easytrieve/Plus adaptability to a wide range of environments and to demonstrate a variety of coding techniques.
- “Project Management System” describes how CA-Easytrieve/Plus can be used for a classical data processing application--a Project Management System.
- “Table of Statements” lists and briefly describes the subset of CA-Easytrieve/Plus statements covered in this guide.
“Cross References” presents a cross-reference listing of CA-Easytrieve/Plus statements to the specific examples in Chapters 13 through 17.

“Index” provide listings to facilitate references to screens and tables, terms and procedures.

## Other CA-Easytrieve/Plus Publications

In addition to this *CA-Easytrieve/Plus Application Guide*, Computer Associates provides the following CA-Easytrieve/Plus documentation:

<table>
<thead>
<tr>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA-Easytrieve/Plus Reference Guide</td>
<td>Contains descriptions of all product features and functions and summaries of each CA-Easytrieve/Plus version.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus CA-Activator Supplement</td>
<td>Explains how to install and maintain CA-Easytrieve/Plus on your OS/390 system, using the CA-Activator.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus Installation Guide</td>
<td>Describes the process of installing and tuning the CA-Easytrieve/Plus system.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus User Guide</td>
<td>Provides new users with the information they need to become productive quickly. It includes a six-lesson tutorial and a format designed to make the material more interesting and easier to comprehend.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus Extended Reporting Facility Guide</td>
<td>Describes support of extended reporting capabilities for Impact Dot, Ink Jet, and Electro Photographic printers.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus Interface Option Guides</td>
<td>Short guides available for users of various system options. These consist of manuals for IMS/DLI processing, CA-IDMS and IDD processing, TOTAL processing, SQL processing, CA-Datacom/DB processing, SUPRA processing, and other CA-Easytrieve/Plus options.</td>
</tr>
<tr>
<td>CA-Easytrieve/Plus Pocket Reference</td>
<td>A quick reference guide to all CA-Easytrieve/Plus syntax.</td>
</tr>
</tbody>
</table>
Related Publications

The following publication, not produced by Computer Associates, is either referenced in this publication or is recommended reading:

- IBM IMS/DLI Applications Programming Manual

Documentation Conventions

The following conventions are used throughout this manual for illustrative purposes.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ braces }</td>
<td>Mandatory choice of one of these entries.</td>
</tr>
<tr>
<td>[ brackets ]</td>
<td>Optional entry or choice of one of these entries.</td>
</tr>
<tr>
<td>( OR bar )</td>
<td>Choice of one of these entries.</td>
</tr>
<tr>
<td>( parentheses )</td>
<td>Multiple parameters must be enclosed in parentheses.</td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate that you can code the immediately preceding parameters multiple times.</td>
</tr>
<tr>
<td>CAPS</td>
<td>All capital letters indicate a keyword, name, or field used in a program example.</td>
</tr>
<tr>
<td>lowercase</td>
<td>Lowercase letters represent variable information in statement syntax. If the same variable types recur within a statement, they are made unique by adding a numeric suffix, such as literal-2.</td>
</tr>
</tbody>
</table>
Variable Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>field-name</td>
<td>Data field defined in your program.</td>
</tr>
<tr>
<td>file-name</td>
<td>Unique file name defined in the library section of your program.</td>
</tr>
<tr>
<td>index name</td>
<td>Name of an INDEX data item.</td>
</tr>
<tr>
<td>integer</td>
<td>Numeric literal (whole number greater than zero).</td>
</tr>
<tr>
<td>job-name</td>
<td>Name a JOB activity.</td>
</tr>
<tr>
<td>letter</td>
<td>Single alphabetic character, such as an edit mask identifier.</td>
</tr>
<tr>
<td>literal</td>
<td>Text string enclosed in quotes or a numeric constant.</td>
</tr>
<tr>
<td>proc-name</td>
<td>Name of a procedure.</td>
</tr>
<tr>
<td>program-name</td>
<td>Name of a program written in a language other than CA-Easytrieve/Plus, such as COBOL or Assembler.</td>
</tr>
<tr>
<td>record-name</td>
<td>Name of an IMS/DLI or CA-IDMS entity.</td>
</tr>
<tr>
<td>report-name</td>
<td>Name of a REPORT.</td>
</tr>
<tr>
<td>sort-name</td>
<td>Name of a SORT activity.</td>
</tr>
</tbody>
</table>
Overview

Introduction

CA-Easytrieve/Plus is an information retrieval and data management system designed to simplify typical programming tasks. Almost any business-oriented task can be accomplished using CA-Easytrieve/Plus. It is simple enough for a beginner to use without additional training, and sophisticated enough to enable a data processing expert to perform complex tasks.

Capabilities

Following is a list of some important CA-Easytrieve/Plus capabilities:

File Processing

- Accepts any number of input files.
- Processes SAM, ISAM, VSAM, or IMS/DLI files.
- Allows fixed, variable, undefined, or spanned record formats.
- Processes data in alphabetic, numeric, packed, packed-unsigned, or binary format.
- Searches files and performs logical data selection based on input or calculation.
- Edits and updates files.
- Matches an unlimited number of files.
- Creates subfiles containing selected records from a master file.
Operations

- Performs extensive computations through user logic; including percentages, averages, and other calculations.
- Sorts on any number of keys.
- Calls your programs and subroutines written in other languages and integrates them into the job.

Output

- Outputs any number of files or reports on one pass of the input file(s).
- Automatically formats output with all totals calculated internally.
- Provides summary reports and output files with no limits on the number and size of control break fields or total fields.
- Makes it easy for you to define and print specially formatted output, such as for W-2 forms, audit confirmations, labels, form letters, and preprinted forms.
- Permits you to vary page sizes within a report, and insert additional header and footer information.
- Enables you to write reports directly to microfiche.

Application

CA-Easytrieve/Plus is designed to make it easy for you to manipulate files and produce reports. It is suitable for beginners in data processing techniques because it is easy to learn. The next exhibit presents a sample program which is used throughout Chapters 2 through 13 of this guide to demonstrate the use of CA-Easytrieve/Plus statements. This sample program is contrived to exemplify a large selection of CA-Easytrieve/Plus statements.
Sample Program

1 PARM  DEBUG(FLOW FLDCHK)
2 *
3 FILE PERSNL FB(150 1800)

4 NAME  17 16  A
5   LAST-NAME NAME  8  A
6 PAY-GROSS  94  4 P  2
7 DEPT  98  3 N
8 DATE-OF-HIRE  136  6 N
9 HIRE-MM DATE-OF-HIRE  2 N
10 HIRE-DD DATE-OF-HIRE +2  2 N
11 HIRE-YY DATE-OF-HIRE +4  2 N
12 SALARY W  4 P  2
13 BONUS W  4 P  2
14 RAISE W  4 P  2
15 SERVICE W  2 N
16 CURR-DATE S  6 N
17 CURR-MM CURR-DATE  2 N
18 CURR-DD CURR-DATE +2  2 N
19 CURR-YY CURR-DATE +4  2 N
20 *
21 FILE ERRPRINT PRINTER
22 *
23 JOB INPUT PERSNL
24 %GETDATE CURR-DATE
25 SALARY = PAY-GROSS * 52
26 PERFORM SERVICE-CALC
27 IF SERVICE LT 1
28     GO TO JOB
29 END-IF
30 PERFORM RAISE-CALC
31 BONUS = 0
32 IF SERVICE GT 14
33     PERFORM BONUS-CALC
34 END-IF
35 SALARY = SALARY + RAISE + BONUS
36 PRINT UPD-RPT
37 *
38 SERVICE-CALC. PROC
39     SERVICE = CURR-YY - HIRE-YY
40     IF CURR-MM LT HIRE-MM
41       SERVICE = SERVICE - 1
42     END-IF
43     IF CURR-MM NE HIRE-MM
44       GOTO QUIT-SERV-CALC
45     END-IF
46     IF CURR-DD LT HIRE-DD
47       SERVICE = SERVICE - 1
48     END-IF
49   QUIT-SERV-CALC
50   END-PROC
51 *
52 RAISE-CALC. PROC
53     IF DEPT LT 940
54       RAISE = SALARY * 0.1
55     ELSE
56       RAISE = SALARY * 0.15
57     END-IF
58   END-PROC
59 *
60 BONUS-CALC. PROC
61     IF SALARY GT 29999
62       DISPLAY ERRPRINT, LAST-NAME, +5, +
63          'INELIGIBLE FOR BONUS'
The program illustrated in the above exhibit processes a Personnel Master File named PERSNL that contains the department numbers, names, salaries, and dates of hire of all employees in an imaginary company.

Six working storage fields contain the results of calculations used in the program and printed on the resulting reports.

Using the three procedures, SERVICE-CALC, RAISE-CALC, and BONUS-CALC, each employee's length of service, annual raise, and eligibility for and amount of a bonus is calculated.

Finally, two reports are produced. The first presents a list of all salaried employees, with the new values for length of service, amount of raise, and salary. The second lists only those employees who received a bonus, their length of service, and the amount of the bonus.

This type of file updating and reporting is a typical application for CA-Easytrieve/Plus. It illustrates many of the statements most commonly used. Portions of this program are referenced throughout this Guide as the various statements and operations are described in detail.

The two exhibits that follow illustrate the reports generated by the sample program. The third exhibit illustrates the printout of the error file ERRPRINT.
### Sample Update Report

#### ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

<table>
<thead>
<tr>
<th>DEPT</th>
<th>NAME</th>
<th>SERV</th>
<th>RAISE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>WALTERS</td>
<td>10</td>
<td>2,204.80</td>
<td>24,252.80</td>
</tr>
<tr>
<td>901</td>
<td></td>
<td></td>
<td>2,204.80</td>
<td>24,252.80</td>
</tr>
<tr>
<td>903</td>
<td>WIMN</td>
<td>30</td>
<td>1,942.72</td>
<td>23,369.92</td>
</tr>
<tr>
<td>903</td>
<td></td>
<td></td>
<td>1,942.72</td>
<td>23,369.92</td>
</tr>
<tr>
<td>911</td>
<td>ARNOLD</td>
<td>13</td>
<td>2,316.60</td>
<td>25,482.60</td>
</tr>
<tr>
<td></td>
<td>GREEN</td>
<td>12</td>
<td>1,901.12</td>
<td>20,912.32</td>
</tr>
<tr>
<td></td>
<td>HAFFER</td>
<td>11</td>
<td>634.14</td>
<td>6,975.54</td>
</tr>
<tr>
<td></td>
<td>ISAAC</td>
<td>16</td>
<td>1,638.72</td>
<td>18,937.92</td>
</tr>
<tr>
<td></td>
<td>KRUSE</td>
<td>21</td>
<td>1,268.48</td>
<td>15,865.28</td>
</tr>
<tr>
<td></td>
<td>LARSON</td>
<td>15</td>
<td>1,476.38</td>
<td>17,240.22</td>
</tr>
<tr>
<td></td>
<td>POST</td>
<td>12</td>
<td>1,518.40</td>
<td>16,762.40</td>
</tr>
<tr>
<td></td>
<td>POWELL</td>
<td>26</td>
<td>1,264.64</td>
<td>15,912.08</td>
</tr>
<tr>
<td></td>
<td>REYNOLDS</td>
<td>20</td>
<td>905.58</td>
<td>11,961.38</td>
</tr>
<tr>
<td></td>
<td>SMOOTH</td>
<td>26</td>
<td>1,639.04</td>
<td>20,029.44</td>
</tr>
<tr>
<td></td>
<td>STRIDE</td>
<td>13</td>
<td>2,009.28</td>
<td>22,102.08</td>
</tr>
<tr>
<td></td>
<td>YOUNG</td>
<td>11</td>
<td>1,638.72</td>
<td>17,937.92</td>
</tr>
<tr>
<td>911</td>
<td></td>
<td></td>
<td>18,187.10</td>
<td>210,658.14</td>
</tr>
<tr>
<td>912</td>
<td>LOYAL</td>
<td>28</td>
<td>1,535.04</td>
<td>18,885.44</td>
</tr>
<tr>
<td>912</td>
<td></td>
<td></td>
<td>1,535.04</td>
<td>18,885.44</td>
</tr>
<tr>
<td>914</td>
<td>CROCI</td>
<td>17</td>
<td>1,955.20</td>
<td>22,507.20</td>
</tr>
<tr>
<td></td>
<td>GRECO</td>
<td>18</td>
<td>5,220.80</td>
<td>57,420.80</td>
</tr>
<tr>
<td></td>
<td>MANHART</td>
<td>16</td>
<td>1,792.96</td>
<td>20,722.56</td>
</tr>
<tr>
<td></td>
<td>RYAN</td>
<td>11</td>
<td>2,075.84</td>
<td>22,834.24</td>
</tr>
<tr>
<td></td>
<td>VETTER</td>
<td>31</td>
<td>1,452.67</td>
<td>17,979.39</td>
</tr>
<tr>
<td>914</td>
<td></td>
<td></td>
<td>12,497.47</td>
<td>141,472.19</td>
</tr>
<tr>
<td>915</td>
<td>CORNING</td>
<td>11</td>
<td>760.03</td>
<td>8,360.35</td>
</tr>
<tr>
<td>915</td>
<td></td>
<td></td>
<td>760.03</td>
<td>8,360.35</td>
</tr>
<tr>
<td>917</td>
<td>TALL</td>
<td>19</td>
<td>2,559.75</td>
<td>29,157.27</td>
</tr>
<tr>
<td>917</td>
<td></td>
<td></td>
<td>2,559.75</td>
<td>29,157.27</td>
</tr>
</tbody>
</table>
### ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

<table>
<thead>
<tr>
<th>DEPT</th>
<th>NAME</th>
<th>SERV</th>
<th>RAISE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>918</td>
<td>BRANDOW</td>
<td>09</td>
<td>4,184.12</td>
<td>46,925.40</td>
</tr>
<tr>
<td></td>
<td>EPERT</td>
<td>11</td>
<td>1,614.08</td>
<td>17,754.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5,798.20</td>
<td>63,780.28</td>
</tr>
<tr>
<td>919</td>
<td>DENNING</td>
<td>15</td>
<td>706.42</td>
<td>8,770.62</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>706.42</td>
<td>8,770.62</td>
</tr>
<tr>
<td>920</td>
<td>MILLER</td>
<td>07</td>
<td>1,630.72</td>
<td>17,937.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1,630.72</td>
<td>17,937.92</td>
</tr>
<tr>
<td>921</td>
<td>HUSS</td>
<td>21</td>
<td>1,876.16</td>
<td>22,637.76</td>
</tr>
<tr>
<td></td>
<td>PETRIK</td>
<td>21</td>
<td>1,148.16</td>
<td>14,629.76</td>
</tr>
<tr>
<td></td>
<td>WARD</td>
<td>12</td>
<td>955.50</td>
<td>10,510.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3,979.82</td>
<td>47,778.02</td>
</tr>
<tr>
<td>923</td>
<td>LACH</td>
<td>15</td>
<td>1,614.08</td>
<td>18,754.88</td>
</tr>
<tr>
<td></td>
<td>THOMPSON</td>
<td>11</td>
<td>1,302.08</td>
<td>14,322.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2,916.16</td>
<td>33,977.76</td>
</tr>
<tr>
<td>924</td>
<td>ROGERS</td>
<td>20</td>
<td>1,710.80</td>
<td>20,818.80</td>
</tr>
<tr>
<td></td>
<td>ZOLTAN</td>
<td>13</td>
<td>650.00</td>
<td>7,150.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2,360.80</td>
<td>27,968.80</td>
</tr>
<tr>
<td>931</td>
<td>FORREST</td>
<td>18</td>
<td>71.76</td>
<td>1,789.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>71.76</td>
<td>1,789.36</td>
</tr>
<tr>
<td>932</td>
<td>BYER</td>
<td>12</td>
<td>2,062.73</td>
<td>22,690.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2,062.73</td>
<td>22,690.09</td>
</tr>
<tr>
<td>935</td>
<td>NAGLE</td>
<td>08</td>
<td>2,882.88</td>
<td>31,711.68</td>
</tr>
<tr>
<td></td>
<td>OSMON</td>
<td>31</td>
<td>3,265.60</td>
<td>35,921.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,148.48</td>
<td>67,633.28</td>
</tr>
<tr>
<td>940</td>
<td>JONES</td>
<td>23</td>
<td>6,277.44</td>
<td>48,127.04</td>
</tr>
<tr>
<td></td>
<td>KELLY</td>
<td>11</td>
<td>1,541.28</td>
<td>11,816.48</td>
</tr>
<tr>
<td></td>
<td>PHILPS</td>
<td>08</td>
<td>1,975.42</td>
<td>15,144.94</td>
</tr>
<tr>
<td></td>
<td>WEST</td>
<td>12</td>
<td>5,740.80</td>
<td>44,012.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15,534.94</td>
<td>119,101.26</td>
</tr>
</tbody>
</table>

### ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

<table>
<thead>
<tr>
<th>DEPT</th>
<th>NAME</th>
<th>SERV</th>
<th>RAISE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>942</td>
<td>JOHNSON</td>
<td>17</td>
<td>5,559.84</td>
<td>42,625.44</td>
</tr>
<tr>
<td></td>
<td>MALLOW</td>
<td>22</td>
<td>2,282.72</td>
<td>18,887.52</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7,762.56</td>
<td>61,512.96</td>
</tr>
<tr>
<td>943</td>
<td>BERG</td>
<td>26</td>
<td>5,921.76</td>
<td>45,400.16</td>
</tr>
<tr>
<td></td>
<td>JUDAR</td>
<td>16</td>
<td>4,611.36</td>
<td>35,353.76</td>
</tr>
<tr>
<td></td>
<td>MCMAHON</td>
<td>19</td>
<td>3,013.92</td>
<td>24,106.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>13,547.04</td>
<td>104,860.64</td>
</tr>
<tr>
<td>944</td>
<td>NORIDGE</td>
<td>13</td>
<td>2,527.20</td>
<td>19,375.20</td>
</tr>
<tr>
<td></td>
<td>TALUS</td>
<td>15</td>
<td>3,594.24</td>
<td>28,555.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6,121.44</td>
<td>47,931.04</td>
</tr>
</tbody>
</table>

Total: 108,327.98 1080,388.14
Sample Bonus Report

ANNUAL BONUS REPORT - SENIOR EMPLOYEES

<table>
<thead>
<tr>
<th>DEPT</th>
<th>LAST-NAME</th>
<th>SERVICE</th>
<th>BONUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>903</td>
<td>WIMN</td>
<td>30</td>
<td>2,000.00</td>
</tr>
<tr>
<td>911</td>
<td>ISAAC</td>
<td>16</td>
<td>1,000.00</td>
</tr>
<tr>
<td>911</td>
<td>KRUSE</td>
<td>21</td>
<td>2,000.00</td>
</tr>
<tr>
<td>911</td>
<td>LARSON</td>
<td>15</td>
<td>1,000.00</td>
</tr>
<tr>
<td>911</td>
<td>POWELL</td>
<td>26</td>
<td>2,000.00</td>
</tr>
<tr>
<td>911</td>
<td>REYNOLDS</td>
<td>20</td>
<td>2,000.00</td>
</tr>
<tr>
<td>911</td>
<td>SMOOTH</td>
<td>26</td>
<td>2,000.00</td>
</tr>
<tr>
<td>912</td>
<td>LOYAL</td>
<td>28</td>
<td>2,000.00</td>
</tr>
<tr>
<td>914</td>
<td>CROCI</td>
<td>17</td>
<td>1,000.00</td>
</tr>
<tr>
<td>914</td>
<td>MANHART</td>
<td>16</td>
<td>1,000.00</td>
</tr>
<tr>
<td>914</td>
<td>VETTER</td>
<td>31</td>
<td>2,000.00</td>
</tr>
<tr>
<td>917</td>
<td>TALL</td>
<td>19</td>
<td>1,000.00</td>
</tr>
<tr>
<td>919</td>
<td>DENNING</td>
<td>15</td>
<td>1,000.00</td>
</tr>
<tr>
<td>921</td>
<td>HUSS</td>
<td>21</td>
<td>2,000.00</td>
</tr>
<tr>
<td>921</td>
<td>PETRIK</td>
<td>21</td>
<td>2,000.00</td>
</tr>
<tr>
<td>923</td>
<td>LACH</td>
<td>15</td>
<td>1,000.00</td>
</tr>
<tr>
<td>924</td>
<td>ROGERS</td>
<td>20</td>
<td>2,000.00</td>
</tr>
<tr>
<td>931</td>
<td>FORREST</td>
<td>18</td>
<td>1,000.00</td>
</tr>
<tr>
<td>942</td>
<td>MALLOW</td>
<td>22</td>
<td>2,000.00</td>
</tr>
<tr>
<td>943</td>
<td>MCMAHON</td>
<td>19</td>
<td>1,000.00</td>
</tr>
<tr>
<td>944</td>
<td>TALUS</td>
<td>15</td>
<td>1,000.00</td>
</tr>
</tbody>
</table>

Sample Error File Printout

BERG  INELIGIBLE FOR BONUS
WEST  INELIGIBLE FOR BONUS
OSMON INELIGIBLE FOR BONUS
GRECO INELIGIBLE FOR BONUS
JOHNSON INELIGIBLE FOR BONUS
JONES  INELIGIBLE FOR BONUS
JUDAR  INELIGIBLE FOR BONUS

Structure

A CA-Easytrieve/Plus program can be composed of up to three sections: one is optional, one is customary, and one is mandatory, as illustrated next.

<table>
<thead>
<tr>
<th>(Optional)</th>
<th>Environment Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Customary)</td>
<td>Library</td>
</tr>
<tr>
<td>(Mandatory)</td>
<td>Activity Definition</td>
</tr>
</tbody>
</table>
Environment Definition Section

This section is optional, and if used, must be the first section of your program. It consists of the PARM statement which can be used to establish a customized operating mode for the duration of your program.

Library Section

This section is also called the data definition section and is usually necessary for file processing and report generation. It follows the PARM statement and contains the FILE statement and field definitions. These statements describe the data to be processed by your program and initialize the required working storage (see Chapter 3, “Library”). The library section of the sample program is illustrated earlier in this chapter.

Activity Definition Section

This section is required. It contains the CA-Easytrieve/Plus statements which accomplish the task for which you created your program (see Chapter 4, “Activity Definition”). It can consist of any number of either or both of two types of activities - JOB and SORT:

- JOB activities read information from input files, examine and manipulate information, write information to output files, and produce printed reports.
- SORT activities create sequenced output files that contain all or part of the records from another (input) file.

Your program can contain any number of JOB and SORT activities, in any order. Within each of these activity types are statements, procedures, and subactivities which specify the tasks your program intends to accomplish, as follows:

A JOB activity is composed of:

- A JOB statement
- One or more CA-Easytrieve/Plus statements
- One or more procedures (optional)
- One or more report subactivities (optional).

A Procedure is composed of:

- A PROC statement
- One or more CA-Easytrieve/Plus statements
- An END-PROC statement.
A Report subactivity is composed of:
- A REPORT statement
- One or more report declaratives
- Report procedures (optional).

A SORT activity is composed of:
- A SORT statement
- Sort procedures (optional).

Procedures are discussed in Chapter 6, “Decision and Branching Logic.” The REPORT statement and associated declaratives and procedures are described in Chapter 8, “Report Processing.”

The next exhibit illustrates the structure of a CA-Easytrieve/Plus program containing the items listed on the previous pages.

<table>
<thead>
<tr>
<th>(Optional)</th>
<th>Environment Definition Section</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PARM Statement .....</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Customary)</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FILE Statement .....</td>
</tr>
<tr>
<td></td>
<td>(field definitions...)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(Mandatory)</th>
<th>Activity Definition Section</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JOB Statement .....</td>
</tr>
<tr>
<td></td>
<td>(Other statements...)</td>
</tr>
<tr>
<td></td>
<td>(Your procedures .....</td>
</tr>
<tr>
<td>REPORT Statement</td>
<td>Report declaratives...]</td>
</tr>
<tr>
<td></td>
<td>(Special-name procedures)</td>
</tr>
<tr>
<td>SORT Statement</td>
<td>(Your procedures .....</td>
</tr>
</tbody>
</table>
Rules of Syntax

CA-Easytrieve/Plus statements have a free-form, English-like structure and a simple, consistent syntax which is easy to understand and remember.

Statement Structure

Each of your program statements (source statements) is a record of 80 characters. As CA-Easytrieve/Plus reads each one, positions 73 through 80 are ignored. These positions are expected to contain optional information, such as statement sequence numbers, and program identifiers. Positions 1 through 72 are expected to contain CA-Easytrieve/Plus statements. All 80 characters are printed on your listing, as illustrated in the next exhibit.

<table>
<thead>
<tr>
<th>CA-Easytrieve Plus Statement</th>
<th>Seq.No. or Ident</th>
</tr>
</thead>
<tbody>
<tr>
<td>******************** statement area ********************</td>
<td>PROGNAME</td>
</tr>
</tbody>
</table>

A statement area can contain more than one CA-Easytrieve/Plus statement or, in the case of continuations, a portion of a statement. In general, a CA-Easytrieve/Plus statement begins with a keyword and is terminated by a period or the end of the statement area, whichever is first. This technique enables you to code more than one statement in a statement area, or to continue a statement which is too large for one statement area.

To enter multiple statements on one line, follow each statement with a period and a space. The next statement is considered to begin in the next available position after the space. For example:


Continued statements are discussed later in this section.
Words

Statements are made up of one or more words. A word can be a keyword, a field name (also called a data name), or a literal, described below. All words begin with a nonblank character and are terminated either by the end of the statement area or by one of the following word delimiters:

<table>
<thead>
<tr>
<th>Word Delimiter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space</td>
</tr>
<tr>
<td>(</td>
<td>Left parenthesis</td>
</tr>
<tr>
<td>)</td>
<td>Right parenthesis</td>
</tr>
<tr>
<td>'</td>
<td>Apostrophe</td>
</tr>
<tr>
<td>.</td>
<td>Period</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>:</td>
<td>Colon</td>
</tr>
</tbody>
</table>

The basic word delimiter is the space. At least one space must follow all other delimiters except the left parenthesis.

Keywords

Keywords are words with specific meanings to CA-Easytrieve/Plus. Some keywords are reserved for the use of CA-Easytrieve/Plus only; the nonreserved words can be used as data names in the appropriate context. Appendix B lists all keywords and identifies those which are reserved.

Field Names

Field names are composed of a combination of not more than 40 characters chosen from the following:

- Alphabetic characters, A through Z, lowercase and uppercase
- Decimal digits 0 through 9
- All special characters, except delimiters.

The first character of a field name must be an alphabetic character or a decimal digit. In addition, a field name must contain at least one alphabetic or special character to distinguish the field name from a number. All working storage field names must be unique, also, all field names must be within a single file. If you use the same field name in more than one file or working storage field, you must qualify the field name with the file name or the word WORK.
A qualified field name consists of the qualifying word followed by a colon and the field name. You can use any number of spaces, or no spaces, to separate the colon from either the qualifying word or the field name.

For example:

PERSNL: SALARY
WORK: SALARY
FILEX : SALARY

Valid Field Names

EMPLOYEE#
TIME-OF-DAY
TOTAL$DOLLARS-FOR-1988

Invalid Field Names

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOCIAL SECURITY NUMBER</td>
<td>Embedded spaces</td>
</tr>
<tr>
<td>EMP’L-NO</td>
<td>Apostrophe not allowed</td>
</tr>
<tr>
<td>$AMOUNT</td>
<td>Must begin with a letter</td>
</tr>
</tbody>
</table>

Literals

Literals can be either alphabetic or numeric. Alphabetic literals are enclosed within apostrophes and can be up to 254 characters long. If an apostrophe occurs naturally within an alphabetic literal, you must code two apostrophes together. For example:

'Judge O''Connor'

Alphabetic literals can contain both letters and numbers, but the numbers are treated the same as letters. For example:

'769 ENTERPRISE DR., OAK BROOK, ILL 60521'

The numbers 709 and 60521 are not numeric values on which an arithmetic operation can be performed.

Numeric literals consist of the characters 0 through 9, and can be up to 18 digits long. They can be prefixed by a plus symbol (+) or a minus symbol (-) to indicate the algebraic sign of the number and can contain a single decimal point to indicate a maximum precision of up to 18 decimal positions. For example:

1126
+112632
-11.2632

Comments

If the first nonblank character of a statement is an asterisk (*), the remainder of that statement area is a comment. You can put comments in your program at any place, except between the portions of a continued statement.
Continuations

A statement is terminated by a period or the last nonblank character in the statement area, unless that character is a hyphen (-) or a plus (+). The hyphen indicates that the statement continues with the first position in the next statement area (which can be a blank).

The plus symbol indicates that the statement continues with the first nonblank character in the next statement area (which could be in the first position); leading blanks are ignored. For example, the LINE statement, which indicates the contents of a report, is as follows:

```
LINE EMPLOYEE# NAME STREET CITY STATE ZIP TELEPHONE +
   REGION DIVISION BRANCH GROSS NET DEDUCTIONS +
   QUARTER YEAR-TO-DATE
```

Environment Definition

The environment under which your CA-Easytrieve/Plus program runs can be determined by one or more of three sources:

- The options table established by your data center at installation. Normally, the default setting of these options is used. The examples and instructions in this guide assume that the defaults are in effect. If you get unexpected results from your program, contact your data center to identify modified installation options.
- The optional PARM statement which overrides the options table. If used, it must be the first statement in your program.
- Parameters of the FILE, SORT, and REPORT statements which, when specified, override the options table and the PARM statement.

PARM Statement

The parameters of the PARM statement provide a method for customizing the operating environment for the duration of one program's compilation and execution.

The two most often used are:

- SYNTAX - Syntax check source statements
- COMPILERE - Syntax check and compile source statements.
SYNTAX Parameter

The SYNTAX parameter terminates CA-Easytrieve/Plus processing after completion of the syntax check operation. For example, use of this parameter enables early checkout of a program before the data files necessary for execution are available.

COMPILE Parameter

The COMPILE parameter terminates CA-Easytrieve/Plus processing after completion of the syntax check and compile operations.

If you do not use the PARM statement, the default is syntax check, compile, and execute.
The library section of your program describes the information that your program processes. This description is in terms that CA-Easytrieve/Plus can understand, that is, files, records, and fields.

**File**

A file is a group of records whose attributes (such as the type of file, the type of device on which it resides, and the format of its records) are provided in the FILE statement parameters.

**Record**

A record is a collection of fields, organized in a consistent format. For example, in a file which contains a payroll history for each employee in a company, a record is all the information about one employee.

**Field**

A field is an elementary item of information. A field represents a single attribute of a single record. For example, in a record which contains all the information about one employee, a field is a single attribute (such as age or length of service) of that employee. The DEFINE statement parameters specify the characteristics of a field (such as location, length, and data format).

The library section of your program provides:

- A general description of the groups of data (files) on which your program is to operate (through the FILE statement).
- A specific description of the individual items of data (fields) within each record of the files or within working storage (through the DEFINE statement).
The next exhibit illustrates the library section of the Sample Program depicted in Chapter 2, “Overview” under the topic Application.

```
2 *
3 FILE PERSNL FB(150 1800)
4 NAME                     17 16   A
5 LAST-NAME NAME 8 A
6 PAY-GROSS                94 4 P 2
7 DEPT                     98 3 N
8 DATE-OF-HIRE             136 6 N
9 HIRE-MM DATE-OF-HIRE 2 N
10 HIRE-DD DATE-OF-HIRE+2 2 N
11 HIRE-YY DATE-OF-HIRE+4 2 N
12 SALARY                  W 4 P 2
13 BONUS                   W 4 P 2
14 RAISE                   W 4 P 2
15 SERVICE                  W 2 N
16 CURR-DATE                5 6 N
17 CURR-MM CURR-DATE 2 N
18 CURR-DD CURR-DATE+2 2 N
19 CURR-YY CURR-DATE+4 2 N
20 *
21 FILE ERRPRINT PRINTER
22 *
```

FILE Statement

The FILE statement describes the files and/or the databases your program references. This description is provided by parameters coded following the keyword FILE. Not all parameters are used with any one file. The next exhibit diagrams the most commonly used FILE parameters.

```
FILE file-name +
   [SYSxxx] +
   [ ]
   [IS]
   File ==> [VIRTUAL]
   Type ==>
   [DLI (dbdname [literal-1])] [ ] +
   [VS ([ES] [PASSWORD 'literal-2'] [CREATE [RESET]])]
   [ ]
   [ ]
   [CARD]
   Device ==> [PRINTER] +
   [DISK]
   [TAPE]
```

FILE Statement

The FILE statement describes the files and/or the databases your program references. This description is provided by parameters coded following the keyword FILE. Not all parameters are used with any one file. The next exhibit diagrams the most commonly used FILE parameters.
FILE Statement

File-name Parameter

FILE file-name

This is a name you give to each of your files. It is the only FILE statement parameter which is mandatory under every circumstance. It must start with a letter, can contain letters, numbers, and a few special characters and can be from one- to eight-characters long (one to seven in VSE). Within your program, the name of each file must be unique — no two files can have the same name.

In the FILE statement sample program (shown earlier), the input file-name is PERSNL.

FILE PERSNL

SYSxxx Parameter (VSE Only)

FILE file-name [SYSxxx]

This optional parameter establishes the logical unit assignments. Valid entries are:

- SYSLST
- SYSPCH
- SYSIPT
- SYS000 through SYS240.

Check with your data processing department to learn if you must supply this parameter.
File-type Parameters

This parameter specifies your file-type. If you do not supply it, CA-Easytrieve/Plus assumes that your file is sequentially ordered. If it is not, you must specify this parameter to identify your file-type. This subject is covered in more detail in Chapter 9, “File Processing.” The file-types are:

<table>
<thead>
<tr>
<th>File-Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS</td>
<td>Indexed Sequential Access Method (ISAM)</td>
</tr>
<tr>
<td>VIRTUAL</td>
<td>CA-Easytrieve/Plus virtual file manager (VFM)</td>
</tr>
</tbody>
</table>
| DL/I      | Designates an IMS/DLI database:  
Dbd= name is alphabetic and names the Database Definition (DBD) in the Program Specification Block (PSB) to be processed. (See Chapter 11, “IMS/DLI Processing.”)  
Literal-1 is numeric and specifies the relative occurrence of the desired DBD in the PSB. |
| VS        | Virtual Storage Access Method (VSAM):  
ES - code this option to indicate that your file accesses as an Entry Sequenced data set.  
PASSWORD - literal-2 is an optional one- to eight-character alphabetic or hexadecimal password for the VSAM file. Enclose the literal in single quotes.  
CREATE - code the CREATE option to load a VSAM file. CREATE by itself implies a new file; include the RESET subparameter to reload an existing file which has been defined as reusable.  
UPDATE - code the UPDATE option to update this file with the PUT or WRITE statements. |
## Device-type Parameters

<table>
<thead>
<tr>
<th>Device-type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CARD]</td>
<td>This option retrieves your file data from the system input stream (SYSIN for OS/390, SYSIPT for VSE). If your operating mode is the default (syntax check, compile, and execute), your file data must follow an END statement within your program, as illustrated below. Only one file in your program can use the CARD option; this file must contain 80-character unblocked records.</td>
</tr>
<tr>
<td>[PUNCH]</td>
<td>The PUNCH option indicates punched card output. Files created with this option are 80-character unblocked records.</td>
</tr>
<tr>
<td>[PRINTER]</td>
<td>The PRINTER option indicates print output files, referenced by the DISPLAY and REPORT statements.</td>
</tr>
<tr>
<td>DISK/TAPE</td>
<td>This option (required only for VSE) indicates the device on which your file resides. Specify this option only if your file is on a device other than the default established at installation.</td>
</tr>
</tbody>
</table>
**Record Format Parameters**

This parameter is required for VSE programs, but not necessarily required for OS/390 programs. CA-Easytrieve/Plus obtains the record format from OS/390 when the file is opened. It can be useful in OS/390, however, for output files. Record format codes are:

- **F** = fixed unblocked
- **V** = variable unblocked
- **U** = undefined
- **FB** = fixed blocked
- **VB** = variable blocked
- **VBS** = variable blocked spanned

**literal-3** = record length  
**literal-4** = block size

The specified record length for a file with variable length records must include four bytes for the Record Descriptor Word (RDW). If the file is blocked, the specified block size must include an additional four bytes for the Block Descriptor Word (BDW). This construction is illustrated next:

```
<table>
<thead>
<tr>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDW</td>
<td>RDW</td>
<td></td>
<td>RDW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>←---Record-----</td>
<td>←---Record-----</td>
</tr>
<tr>
<td></td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
```

**Note:** A block size designation of FULLTRK specifies that the output block size be set to the maximum track capacity of the disk or the next lower multiple of the record size for FB files. For devices with track lengths greater than 32760, FULLTRK indicates a block size of 32760.
TABLE

[       [INSTREAM ]]  
[TABLE [       [   ]]  
[       [literal-5]]

This option identifies a file that you are specifying as a table. The format of table data must follow some strict rules, but its use is very efficient (see Chapter 10, “Table Processing”). The information in this file is accessed by the SEARCH statement. The table data can reside within your program (INSTREAM), or you can store it external to your program.

INSTREAM Tables

INSTREAM directs CA-Easytrieve/Plus to look for the table data within your program immediately following the associated FILE statement. This table is created by coding the data at the same time you code your program; it is established at the time your program is compiled. The size of an INSTREAM table is limited only by the amount of available memory. Instream tables are very useful for decoding information into a more usable format, such as printing department names instead of department numbers on a report.

External Tables

If you specify the TABLE option with no subparameter, the file is an external table whose maximum number of entries is limited by a value in the options table established at installation. Check with your data center to determine this value.

If the number of entries in your external table is larger than the default value, you can code literal-5 to specify the maximum number of entries. External tables are established for use during initiation of the JOB activity that contains the SEARCH statement which references them.
**DEFINE Statement**

The DEFINE statement (with or without the keyword DEFINE) describes data fields within files or within working storage. Optionally, you can omit the DEFINE keyword when the field definitions immediately follow the associated FILE statement. The next exhibit illustrates the DEFINE statement.

<table>
<thead>
<tr>
<th>Environment Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library Section</td>
</tr>
<tr>
<td>FILE ...</td>
</tr>
<tr>
<td>file field</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>FILE</td>
</tr>
<tr>
<td>file field</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>work field</td>
</tr>
<tr>
<td>work field</td>
</tr>
</tbody>
</table>

| Activities Section |

There are three conditions which apply to data fields either in a file or within working storage, as follows:

- Any number of fields can be defined.
- Field-names must be unique within a file or within working storage. There can be no duplicates. The same field-name can be defined in multiple files.
- A field must be DEFINEd before you can use it in your program.

The description of these fields is provided by the parameters of the DEFINE statement. The field-name, location, and attributes parameters are mandatory; MASK and VALUE are optional. The next exhibit diagrams the DEFINE statement and these parameters.

```
[DEFINE] field-name +

{literal-1} +
{field-name-2 [+nn]} +
{W} +
{S} +

{[A]} +
{[N]} +

{P} [literal-3] +
{B} +
{U} +

[MASK ([letter] [BWZ] [literal-4])] +

[VALUE literal-5]
```
**Field-name Parameter**

```
[DEFINE]  field-name
```

This is the name you give to the field you are defining. It must start with a letter; can contain letters, numbers, and special characters; and can be from 1 to 40 characters long.

The field-names in the sample program are illustrated next.

```
FILE PERSNL FB(150 1800)
1 NAME                     17 16 A
2 LAST-NAME                B A
3 PAY-GROSS                 94  4 P 2
4 DEPT                      98  3 N
5 DATE-OF-HIRE            136 6 N
6 HIRE-MM                   2 N
7 HIRE-DD                   +2 2 N
8 HIRE-YY                   +4 2 N
9 SALARY                    W 4 P 2
10 BONUS                     W 4 P 2
11 RAISE                     W 4 P 2
12 SERVICE                   W 2 N
13 CURR-DATE                 S 6 N
14 CURR-MM                   2 N
15 CURR-DD                   +2 2 N
16 CURR-YY                   +4 2 N
17 CURR-MM                   2 N
18 CURR-DD                   +2 2 N
19 CURR-YY                   +4 2 N
20 *
```

**Location Parameter**

```
{literal-1   }
{field-name-2}
Location =>    {            } [+nn]
{W           }
{S           }
```

This parameter identifies the location of the named field within a record or identifies it as a working storage field. The codes to specify location are:

- `literal-1`
  - Specifies the location of the file field's leftmost byte. It is the starting position of this field relative to the first position of the record (position one (1)).

- `field-name-2`
  - Specifies the location of the leftmost byte of a file field as the relative displacement from the start of a previously defined field.
(W or S)

Establishes a working storage field. Fields coded as W are spooled to report (work) files; fields coded as S are not (see Chapter 8, “Report Processing”).

In the field definitions sample program (shown earlier), the first designation to the right of the field-name is the location parameter. Four of the fields, NAME, PAY-GROSS, DEPT, and DATE-OF-HIRE are specified with a numeric value which indicates the starting position of each of these fields relative to the beginning of the record.

Four fields, LAST-NAME, HIRE-MM, HIRE-DD, and HIRE-YY are subfields, specified with a relative displacement to their primary fields: NAME and DATE-OF-HIRE.

Six fields, VAC-HRS, SALARY, BONUS, RAISE, SERVICE, and CURR-DATE are located in working storage. CURR-DATE also has three subfields: CURR-MM, CURR-DD, and CURR-YY.

Attributes Parameter

{ {A} }            
{ {N} }            

Attributes ==> {literal-2} {P} {literal-3}}
{ {B} }            
{ {U} }            

This parameter is specified as three components: field length, data format and number of decimal positions, if any. These values are interdependent in many cases.

Field Length (in bytes)

Specified by literal-2. This value is constrained by the associated data format. See the Field Attribute Relationships table below.

Data Format

Select one of the following codes:

A - alphabetic. Use when none of the numeric data types apply to this field.

N - zoned decimal. The field contains digits 0 through 9 in external decimal form (for example, 0 = X'FO').
**P** - packed decimal. The field contains numbers which meet IBM's definition of internal packed decimal. For example, a two-byte packed field containing the value 123 looks like X'123F'.

**B** - binary. The field contains binary data. Depending on their field length, binary fields can contain values whose maximum is equivalent to the following number of decimal digits:

<table>
<thead>
<tr>
<th>Length in Bytes</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>

**U** - unsigned packed decimal. It is the same as packed decimal, but with the sign stripped off. A two-byte unsigned packed field containing the value 123 looks like X'0123'. This lets you reference part of a packed field without allowing for its sign position.

**Number of Decimal Positions**

Specified by literal-3. Specification of this parameter designates the field as signed quantitative, which is required for performing signed arithmetic. In addition, during control report processing, all fields for which decimal positions are specified are automatically totaled. If this parameter is not specified for a numeric field type, the field is considered unsigned (positive) and is printed with leading zeros by default. Literal-3 is invalid when data format is A.

The following table delineates the relationship between field length, data format, and the valid number of decimal positions for each field.

**Field Attribute Relationships**

<table>
<thead>
<tr>
<th>Data Format Code</th>
<th>Maximum Length (bytes)</th>
<th>Number of Decimal Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>254</td>
<td>not valid</td>
</tr>
<tr>
<td>N</td>
<td>18</td>
<td>0 - 18</td>
</tr>
<tr>
<td>P</td>
<td>10</td>
<td>0 - 18</td>
</tr>
<tr>
<td>B</td>
<td>4</td>
<td>0 - 10</td>
</tr>
<tr>
<td>U</td>
<td>9</td>
<td>0 - 18</td>
</tr>
</tbody>
</table>
The field attribute specifications in the Sample Program Library Section, shown earlier, can be read as illustrated in the following table.

<table>
<thead>
<tr>
<th>Field-name</th>
<th>Length</th>
<th>Format</th>
<th>Decimal Places</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>16 bytes</td>
<td>A</td>
<td>None</td>
</tr>
<tr>
<td>PAY-GROSS</td>
<td>4 bytes</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>DEPT</td>
<td>3 bytes</td>
<td>N</td>
<td>None</td>
</tr>
<tr>
<td>DATE-OF-HIRE</td>
<td>6 bytes</td>
<td>N</td>
<td>None</td>
</tr>
<tr>
<td>VAC-HRS</td>
<td>3 bytes</td>
<td>N</td>
<td>None</td>
</tr>
<tr>
<td>SALARY</td>
<td>4 bytes</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>BONUS</td>
<td>4 bytes</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>RAISE</td>
<td>4 bytes</td>
<td>P</td>
<td>2</td>
</tr>
<tr>
<td>SERVICE</td>
<td>2 bytes</td>
<td>N</td>
<td>None</td>
</tr>
<tr>
<td>CURR-DATE</td>
<td>6 bytes</td>
<td>N</td>
<td>None</td>
</tr>
</tbody>
</table>

Refer to the Sample Update Report in Chapter 2, “Overview,” to see how the data fits into these field attribute specifications.

**MASK Parameter**

MASK ([letter] [BWZ] [literal-4])

This optional parameter can specify a pattern (edit mask) for printing a numeric field on a report. Alphabetic fields cannot be edited. The subparameters are:

[letter]

Letter is an alphabetic identifier for a print mask which is:
- Specified in this DEFINE statement with literal-4
- Specified previously in the program
- Specified by your data center at installation. Check with them.

This identifier can be any letter A to Y. If there is no currently established mask with this identifier, the mask in literal-4 is associated with this identifier and applied to the field, named in this statement, for subsequent print references.

[BWZ]

BWZ (blank when zero) suppresses printing a numeric field when it contains all zeros.
Literal-4 is the print edit mask to use. It is an alphabetic literal created with a combination of the following characters:

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Causes any digit to print.</td>
</tr>
<tr>
<td>Z</td>
<td>Causes any digit except leading zeros to print.</td>
</tr>
<tr>
<td>*</td>
<td>Causes an asterisk to replace leading zero digits.</td>
</tr>
<tr>
<td>-</td>
<td>Causes a minus sign to print before the first or after the last digit of a negative number.</td>
</tr>
<tr>
<td>$</td>
<td>Causes a currency symbol to print before the first nonzero digit.</td>
</tr>
<tr>
<td>X</td>
<td>Permits any character to be printed with the edited data.</td>
</tr>
</tbody>
</table>

Any character, except Z, placed beyond the rightmost digit of a signed quantitative field prints if the field contains a negative value.

The system default masks for numeric fields with decimal positions are illustrated next.

<table>
<thead>
<tr>
<th>Number of Decimals</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>ZZZZZZZZZZZZZZZZZZZZ *</td>
</tr>
<tr>
<td>0</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ-</td>
</tr>
<tr>
<td>1</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, Z9-</td>
</tr>
<tr>
<td>2</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99-</td>
</tr>
<tr>
<td>3</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999-</td>
</tr>
<tr>
<td>4</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 9999-</td>
</tr>
<tr>
<td>5</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99999-</td>
</tr>
<tr>
<td>6</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999999-</td>
</tr>
<tr>
<td>7</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 9999999-</td>
</tr>
<tr>
<td>8</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99999999-</td>
</tr>
<tr>
<td>9</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999999999-</td>
</tr>
<tr>
<td>10</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 9999999999-</td>
</tr>
<tr>
<td>11</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99999999999-</td>
</tr>
<tr>
<td>12</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999999999999-</td>
</tr>
<tr>
<td>13</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 9999999999999-</td>
</tr>
<tr>
<td>14</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99999999999999-</td>
</tr>
<tr>
<td>15</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999999999999999-</td>
</tr>
<tr>
<td>16</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 9999999999999999-</td>
</tr>
<tr>
<td>17</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 99999999999999999-</td>
</tr>
<tr>
<td>18</td>
<td>ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, 999999999999999999-</td>
</tr>
</tbody>
</table>

* For zoned decimal fields with no decimals, the default mask is '999999999999999999'.

[Literal-4]
The next exhibit illustrates some print masks and their purposes.

<table>
<thead>
<tr>
<th>Mask</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>'(999) 999-9999'</td>
<td>Telephone Number</td>
</tr>
<tr>
<td>'999-99-9999'</td>
<td>Social Security Number</td>
</tr>
<tr>
<td>'99/99/99'</td>
<td>Date</td>
</tr>
<tr>
<td>'$$$$$$9.99-.'</td>
<td>Money (with floating $)</td>
</tr>
<tr>
<td>'********999.99-.'</td>
<td>Protected check amount</td>
</tr>
</tbody>
</table>

**VALUE Parameter**

With CA-Easytrieve/Plus, a field defined with a location of working storage (W or S), and a data format of A (alphabetic), is initialized to blanks. Numeric working storage fields are initialized to zeros. To initialize a working storage field to another value, use the VALUE parameter.

For example, if you are defining an alphabetic field whose name is MONTH, and you want to initialize it to the value JANUARY, your statement might read:

```define month w 10 a value 'january'```

where:

- **DEFINE** is the keyword which identifies your statement
- **MONTH** is the name of the field being defined
- **W** is the location parameter = working storage
- **10** is the field length parameter = 10 bytes
- **A** is the data format parameter = alphabetic
- **VALUE** specifies the initial contents of MONTH = JANUARY.

Initialization of a numeric field might read:

```define year w 4 n value 1999```

where:

- **DEFINE** is the keyword
- **YEAR** is your field-name
- **W** locates your field in working storage
- **4** indicates a field length of 4 bytes
- **N** specifies a zoned decimal data format
- **VALUE** initializes field YEAR to 1999.
Chapter 4

Activity Definition

Introduction

The activity definition section of your program contains the CA-Easytrieve/Plus statements that perform the tasks for which you created your program: reading in, processing, and writing out data. These tasks are divided into two activity types - JOB activities and SORT activities.

JOB Activities

JOB activities, identified by the JOB statement, read data from input files described in the library section of your program. See Chapter 3, “Library.” Examine and manipulate this data as directed by CA-Easytrieve/Plus statements, and write data to output files and the appropriate report declaratives.

SORT Activities

SORT activities, initiated by the SORT statement, sequence files in the order specified by parameters of this statement. These sequenced files can in turn be processed by one or more JOB activities.

Note: You can code any number of JOB and/or SORT activities in your program.

This chapter discusses the JOB and SORT statements and their associated parameters. These statements provide CA-Easytrieve/Plus with the information required for automatic input and output of data. Data under your control is input with the GET and READ statements, and output with the PUT and WRITE statements.
The next exhibit illustrates the activity portion of the Sample Program including the REPORT declaratives. The Sample Program is depicted in Chapter 2, “Overview,” under the topic Application.

Sample Program Activity Section

```plaintext
1 PARM DEBUG(FLOW FLDCHK)
2 *
3 FILE PERSNL FB(150 1800)
4 NAME 17 16 A
5 LAST-NAME NAME 8 A
6 PAY-GROSS 94 4 P 2
7 DEPT 98 3 N
8 DATE-OF-HIRE 136 6 N
9 HIRE-MM DATE-OF-HIRE 2 N
10 HIRE-DD DATE-OF-HIRE +2 2 N
11 HIRE-YY DATE-OF-HIRE +4 2 N
12 SALARY W 4 P 2
13 BONUS W 4 P 2
14 RAISE W 4 P 2
15 SERVICE W 2 N
16 CURR-DATE S 6 N
17 CURR-MM CURR-DATE 2 N
18 CURR-DD CURR-DATE +2 2 N
19 CURR-YY CURR-DATE +4 2 N
20 *
21 FILE ERRPRINT PRINTER
22 *
23 JOB INPUT PERSNL
24 %GETDATE CURR-DATE
25 SALARY = PAY-GROSS * 52
26 PERFORM SERVICE-CALC
27 IF SERVICE LT 1
28 GO TO JOB
29 END-IF
30 PERFORM RAISE-CALC
31 BONUS = 0
32 IF SERVICE GT 14
33 PERFORM BONUS-CALC
34 END-IF
35 IF SERVICE GT 14
36 PERFORM RAIS-CALC
37 END-IF
38 SALARY = SALARY + RAISE + BONUS
39 PRINT UPD-RPT
40 *
41 SERVICE-CALC. PROC
42 SERVICE = CURR-YY - HIRE-YY
43 IF CURR-MM < HIRE-MM
44 SERVICE = SERVICE - 1
45 END-IF
46 IF CURR-MM NE HIRE-MM
47 GOTO QUIT-SERV-CALC
48 END-IF
49 QUIT-SERV-CALC
50 END-PROC
51 *
52 RAISE-CALC. PROC
53 IF DEPT LT 940
54 RAISE = SALARY * 0.1
55 ELSE
56 RAISE = SALARY * 0.15
57 END-IF
58 END-PROC
59 *
```
JOB Statement

The JOB statement identifies the files whose records are automatically provided to your program (automatic input). The next exhibit diagrams the JOB statement and associated parameters.

```
JOB [NAME job-name] [NULL]

[INPUT (file-name [KEY(field-name...)] ...) ...]
```

This parameter is optional. It identifies the automatic input as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file-name</td>
<td>Provides the name of the file you want to have controlled automatically. This can be any name previously coded on a FILE statement.</td>
</tr>
<tr>
<td>KEY field-name</td>
<td>Use this subparameter to identify fields within the above-named file when it is used in synchronized file processing (see Chapter 9, “File Processing”). The files are processed in the order in which these keys appear in the JOB statement. There is no limit on the number of fields that you can use as keys.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>Code this subparameter to inhibit automatic input. Normally, a job is implicitly stopped when the automatic input file(s) is exhausted. However, if you code NULL, the program continues running until a STOP statement is executed.</td>
</tr>
<tr>
<td>NAME job-name</td>
<td>Names the JOB activity. Job-name can be up to 40 characters long; the first character must be alphabetic. This parameter is used only for documentation purposes.</td>
</tr>
</tbody>
</table>

If you do not specify the INPUT parameter, CA-Easytrieve/Plus provides an automatic input file. The default input file is chosen as follows:

1. First choice is the file created by a SORT operation which immediately preceded this JOB activity. If there is no such SORT file, see number 2 below.
2. Second choice is the first file which you specified in the library section of this program.
The next exhibit illustrates the processing flow of a JOB activity:
SORT Statement

The SORT statement orders any file which can be processed sequentially. Use this statement if you want to output a sorted file. If you do not need a sorted output file, but simply want a report to be printed in a specific order, you can accomplish this task through the SEQUENCE statement in the REPORT declaratives, as illustrated in the Sample Program Activity Section shown earlier. Refer to the illustration of the Sample Update Report in Chapter 2, “Overview,” under the topic Application to see the result of the SEQUENCE statement in the sample program.

The next exhibit diagrams the SORT statement and associated parameters.

SORT  file-name-1  +
    TO  file-name-2  +
    USING (field-name [D]...)  +
    [BEFORE proc-name]  +
    [NAME sort-name]

file-name-1

This is the name of your input file (the file to be sorted). This name must have been previously coded on a FILE statement in the library section of your program and must reference a file-type that can be accessed sequentially, such as SAM, VSAM, ISAM, or VFM (see Chapter 3, “Library”).

TO file-name-2

This parameter provides the name of the sorted output file. If your SORT activity is the permanent reordering of one file, this name can be the same as file-name-1 (not permitted with VSAM or ISAM files). Otherwise, requirements similar to those for file-name-1 must be met for this TO filename, that is, the name must have been previously provided in a FILE statement in your program’s library and the file type must be SAM, or VFM.

USING (field-name [D]...)

The USING parameter identifies data fields within the input file (file-name-1) which you can use as sort keys. You can choose any number of fields for sort keys, up to the limit of your installation’s sort program.

These data fields must be DEFINEd in the library section before your program can use them.

The subparameter D, following the field-name, causes that field to be sorted in descending order. If you do not specify D, the sort default is ascending order.
[BEFORE proc-name]

This optional parameter identifies a procedure that prescreens, modifies, and selects input records for the sort. Proc-name is the name that appears on the PROC statement which identifies your procedure. CA-Easytrieve/Plus supplies input records to your sort procedure one at a time. If you use a BEFORE procedure, a SELECT statement must be executed for each record that you want to sort.

**SELECT Statement**

If you SELECT a record more than once, it still appears only once on the SORTed file. The next exhibit illustrates the use of the SELECT statement in a BEFORE procedure.

* 
FILE PERSNL FB(150 1800)  
  OLD-EMP#     9 5 N 
  PAY-GROSS    94 4 P 2 
* 
FILE SORTPER F 150 VIRTUAL 
* 
SORT PERSNL + 
  TO SORTPER + 
  USING OLD-EMP# + 
  BEFORE SCREENER 
* 
SCREENER. PROC 
  IF PAY-GROSS LT 29999 
    SELECT 
    END-IF 
  END-PROC 
* 

[NAME sort-name]

The optional NAME parameter names the SORT activity. Sort-name can be up to 40 characters long. The first character must be alphabetic. This parameter is used only for documentation purposes.
Assignment Statement

This chapter describes several ways CA-Easytrieve/Plus enables you to manipulate data within your program. The primary vehicle is the Assignment statement.

The Assignment statement establishes the value of a field by one of two means:
- Equivalence by copying the data from another (named) field, or from a specified literal
- As the result of an arithmetic expression.

Equivalence

The format used to copy data from one field to another, or from a literal to a field, is diagrammed in the next exhibit.

```
{= } {field-name-2}
field-name-1 { } {            }
{EQ} {literal     }
```

The value of field-name-1 is set equal to the value of field-name-2 or literal, whichever is specified. There are certain restrictions on this function, as follows:
- You can specify only one equivalent; that is, either field-name-2 or literal.
- If field-name-1 has been DEFINEd as alphabetic, literal must also be alphabetic. If literal is shorter than field-name-1, padding is on the right.
- If both field-name-1 and field-name-2 are alphabetic, but not the same size, padding or truncation, as appropriate, occurs on the right.
- If field-name-1 is alphabetic and field-name-2 is numeric, the resulting value in field-name-1 is zoned decimal, with padding or truncation on the left, as necessary.
- If field-name-1 is numeric, field-name-2 or literal must be numeric.
Arithmetic Expression

An arithmetic expression produces a numeric value by adding, subtracting, multiplying, or dividing any number of numeric quantities. Field-name-1 is set to the result, as diagrammed in the next exhibit. All fields and literals in this statement must be numeric.

```
{  } {               } {* } {            }
{= } {field-name-2 } {/ } {field-name-3}
field-name-1 {  } {               } {  } {            }  ...
{EQ} {literal-1      } {+ } {literal-2   }
{  } {               } {- } {            }
```

The Sample Program Assignment Statements, shown below, illustrates the use of this type of Assignment statement in the sample program; specifically, the statement which reads:

```
SALARY = PAY-GROSS * 52
```

This statement directs CA-Easytrieve/Plus to multiply the value in the field named PAY-GROSS by the literal 52 and place the result into a field named SALARY. This calculates annual salary by multiplying the weekly wage by the number of weeks in a year. The SALARY field does not exist in file PERSNL, but is calculated on a temporary basis by defining it as a working storage field.

```
22 *
23 JOB INPUT PERSNL
24 %GETDATE CURR-DATE
42 SALARY = PAY-GROSS * 52
43 PERFORM SERVICE-CALC
44 IF SERVICE LT 1
45 GO TO JOB
46 END-IF
47 PERFORM RAISE-CALC
48 BONUS = 0
49 IF SERVICE GT 14
50 PERFORM BONUS-CALC
51 END-IF
52 SALARY = SALARY + RAISE + BONUS
53 PRINT UPD-RPT
54 *
```

Another Assignment statement in the above exhibit is:

```
SALARY = SALARY + RAISE + BONUS
```

This statement uses multiple addition operations to calculate the value of field SALARY. There is no limit to the number of arithmetic operations which can be specified to the right of the equal sign.

CA-Easytrieve/Plus normally performs arithmetic operations in the following order:

* multiplication or / division
* addition or - subtraction
This customary evaluation order is illustrated in the next exhibit.

$$11 + \frac{5 \times 8}{16 + 4} \quad \text{Step 1}$$

$$11 + \frac{40}{16 + 4} \quad \text{Step 2}$$

$$11 + 3 \quad \text{Step 3}$$

$$\frac{51}{3} + 4 \quad \text{Step 4}$$

$$48 + 4 \quad \text{Step 5}$$

$$52$$

You can override the normal order of evaluation by using parentheses; expressions within parentheses are evaluated first. Any level of parenthesis nesting is permitted; evaluation proceeds from the innermost level to the outermost, as illustrated in the next exhibit.

$$11 + 5 \times \left( \frac{(8 - 48)}{16 + 4} \right) \quad \text{Step 1}$$

$$11 + 5 \times \left( \frac{-40}{16 + 4} \right) \quad \text{Step 2}$$

$$11 + 5 \times \left( -\frac{2.5}{4} \right) \quad \text{Step 3}$$

$$11 + \frac{1.5}{4} \quad \text{Step 4}$$

$$11 + 7.5 \quad \text{Step 5}$$

$$18.5$$
Chapter 6
Decision and Branching Logic

Introduction

A group of CA-Easytrieve/Plus statements controls the execution of your program by means of decision and branching logic. Decisions are made in response to an evaluation of conditional expressions coded as parameters of CA-Easytrieve/Plus decision statements. As a result of the decision, subsequent statements can or cannot be executed, or execution can branch out of the customary top-to-bottom line of flow to another place in the program. This group of statements includes:

- IF, ELSE, and END-IF
- DO and END-DO
- GOTO
- PERFORM
- STOP.

IF and DO contain the conditional expressions on which the decisions are based, as illustrated in the next two exhibits.

Construction of IF Statement

```
IF  condition
  *Statements executed if condition is true*

[ELSE                                        ]   Optional
[ *Statements executed if condition is false*]
END-IF
```

Construction of DO Statement

```
DO WHILE  condition
  *Statements executed repetitively if condition is true*
END-DO
```
The next two exhibits illustrate the processing which takes place when an IF or DO statement is executed.

GOTO and PERFORM cause a branch to another location in your program. STOP halts execution of the activity.
Conditional Expressions

Conditional expressions are evaluated by asking the question: Is this condition true? Which of the following statements are executed, or whether the program branches and where it goes, depends on whether the answer is yes or no. Conditional expressions can be:

Single:
- One condition, or a choice of one of several individual conditions.

Combined:
- Any number of conditions, all of which must be considered in the evaluation.

Combinations can be between like conditions, such as two field relational conditions:
IF NAME EQ 'ANDERSON', AND EMPL# EQ 41552

Or unlike conditions, such as one field class condition and one field relational condition:
IF EMPL# NUMERIC, AND EMPL# GT 15555

The next exhibit illustrates some of the conditional expressions used in the sample program.

* IF SERVICE GT 19
  BONUS = 2000
  ELSE
    BONUS = 1000
  END-IF
* IF SERVICE EQ 6 THRU 10
  VAC-HRS = 120
  END-IF
*

The statement which reads:
IF SERVICE GT 19

includes a field relational condition. The value in field SERVICE is compared to the literal 19. If SERVICE is greater than (GT) 19, field BONUS is set to 2000. Otherwise (ELSE), field BONUS is set to 1000.
The statement which reads:

```plaintext
IF SERVICE EQ 6 THRU 10
```

includes a field series condition. If the value in field SERVICE is outside the specified range, execution skips the Assignment statement and resumes with the statement following END-IF.

There are four simple conditions (having at most two operands) and two extended conditions (having potentially an unlimited number of operands). The simple conditions are:

- Field Relational
- Field Class
- File Presence
- Record Relational.

File presence and record relational are useful only with synchronized file processing and are discussed in detail in Chapter 9, “File Processing.” The extended conditions are:

- Field Series
- File Presence Series.

All conditions, either simple or extended, can be combined using the logical connectors AND or OR in any combination. Combined conditions are evaluated, as follows:

- Conditions connected by AND are evaluated first; the combined condition is true if ALL of the connected conditions are true.
- Conditions connected by OR are evaluated next; the combined condition is true when ANY of the connected conditions are true.

Parentheses can be used to group combined conditions. This overrides the normal evaluation order of the AND or OR relationships. The next exhibit presents examples of combined conditions.

```plaintext
IF NAME EQ 'ANDERSON', AND EMPL# EQ 41552
IF DEPT# EQ 911 THRU 921, OR NAME = 'AMAN' THRU 'LYON'
IF EMPL# NUMERIC, AND EMPL# GT 15555
IF NET GT GROSS, OR NET ZEROS, OR +
    DEDUCTIONS NE (GROSS - NET)
```

In the above exhibit, the first IF statement combines two field relational conditions to test for a specific name and a specific employee number. The second IF statement combines two field series conditions to test for a numeric and an alphabetic range. The third IF combines a field class and a field relational condition. The field class condition tests to see if field EMPL# is numeric. The field relational condition tests to see if the value of this field is greater than 15555.
The last IF statement also combines the field relational and field class conditions. The field class condition tests to see if field NET is zeros; the two field relational conditions test to see if the value of field NET is greater than the value of field GROSS, or if the value of field DEDUCTIONS does not equal the result of the arithmetic expression (GROSS - NET).

The most commonly used condition formats are:

- Field Relational
- Field Class
- Field Series.

These formats are described in detail next.

**Field Relational Condition**

This condition compares a specified field with another field, an alphabetic or numeric literal, or an arithmetic expression, as diagrammed in the next exhibit.

\[
\begin{align*}
&\{ \text{EQ} = \} \\
&\{ \text{NE} \neq \} \\
&\{ \text{LT} < \} \{ \text{field-name-2} \} \\
&\{ \text{LE} \leq \} \{ \text{literal} \} \\
&\{ \text{GT} > \} \{ \text{arithmetic expression} \} \\
&\{ \text{GE} \geq \}
\end{align*}
\]

Valid operators for the field relational condition are:

- **EQ** = - Equal
- **NE** ¬= - Not equal
- **LT** < - Less than
- **LE** \(\leq\) - Less than or equal to
- **GT** > - Greater than
- **GE** \(\geq\) - Greater than or equal to

The following rules apply to the use of this condition:

1. If field-name-1 is alphabetic, it can be compared to an alphabetic or numeric field or an alphabetic literal. It cannot be compared to an arithmetic expression. A numeric field is converted to zoned decimal before the comparison is made.

2. If field-name-1 is numeric, it can be compared to a numeric field, a numeric literal, or an arithmetic expression. It cannot be compared to an alphabetic field or literal.
The next exhibit presents some examples of field relational conditions:

*  
FILE PAYFILE
  EMPL#    9 S N
  NAME     17 20 A
  NET      90 4 P 2
  GROSS    94 4 P 2
  DEDUCTIONS W 4 P 2
*  
JOB INPUT PAYFILE
  IF NAME NE 'ANDERSON'
  IF NET LT GROSS
  IF EMPL# GT 18555
  IF DEDUCTIONS EQ (GROSS - NET)
*

Field Class Condition

This condition determines whether a named field does or does not contain a certain class of data, specifically, alphabetic, numeric, zero, or space characters. The format is diagrammed in the next exhibit.

```
{ ALPHABETIC }
{ NUMERIC    }
{ SPACE      }
field-name  [NOT] { SPACES     }
{ ZERO       }
{ ZEROS      }
{ ZEROES     }
```

There is no relational operator in this conditional expression. The named field is tested for the presence of the specified class of data, unless the optional NOT parameter is supplied. In this case, the field is tested for the absence of the specified class of data. The data class tests are performed as follows:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALPHABETIC</td>
<td>Each byte of the field is tested for either letters A through Z or a space character.</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>The field is tested for digits 0 through 9 in the correct format for the field's defined data type. In the case of data types N and P, the low-order position of the field is tested for a valid sign.</td>
</tr>
<tr>
<td>SPACE, SPACES</td>
<td>Each byte of the field is tested for the space character.</td>
</tr>
<tr>
<td>ZERO, ZEROS, ZEROES</td>
<td>The field is tested for a zero value in the correct format for the field's defined data type.</td>
</tr>
</tbody>
</table>
The next exhibit illustrates the use of the field class condition.

```
* FILE PAYFILE
  EMPL# 9  5  N
  NAME 17  20  A
  GROSS 94   4  P  2
*
* JOB INPUT PAYFILE
  IF  NAME ALPHABETIC
  IF  EMPL# NUMERIC
  IF  GROSS NOT ZERO
*
```

**Field Series Condition**

The field series condition compares a specified field with a series or range of values in other fields, alphabetic or numeric literals, or any combination of these, as shown in the next exhibit.

```
{IF           }              {EQ = } {field-name-2 [field-name-3 ]}
{DO WHILE     } field-name-1 {     } {             [              ] }  ...
{RETRIEVE...WHILE}              {NE ¬=} {literal-1    [THRU literal-2]}
```

Valid operators for the field series condition are:

- **(EQ = )** - Equal
- **(NE ¬=)** - Not equal

You can code any number of fields and/or literals to the right of the operator. The following rules apply to the use of this condition:

- If field-name-1 is alphabetic, it can be compared to alphabetic or numeric fields, and/or alphabetic literals. Numeric fields are converted to zoned decimal before the comparison is made.

- If field-name-1 is numeric, it can be compared to numeric fields and/or literals. It cannot be compared to alphabetic fields or literals.

- Each value in the series of values to the right of the operator represents either a single value (for example, 10555) or a range of values (for example, 10555 through 15555).

- A field series conditional expression using the equal operator is tested by comparing field-name-1 to each value in the series. If the value is a single value, the test is for equality between field-name-1 and field-name-2 (or literal-1). If the value is a range of values, the test is for field-name-1 within the range defined by field-name-2 (or literal-1) and field-name-3 (or literal-2). The field series conditional expression is true if at least one test is true.
A field series conditional expression using the not-equal operator is tested by comparing field-name-1 to each value in the series. If the value is a single value, the test is for inequality between field-name-1 and field-name-2 (or literal-1). If the value is a range of values, the test is for field-name-1 outside the range defined by field-name-2 (or literal-1) and field-name-3 (or literal-2). The field series conditional expression is true only if all tests are true.

The next exhibit presents some examples of field series conditions.

```plaintext
* FILE PAYFILE
  EMPL#   9   S   N
  NAME    17  20  A
  NET     90   4  P  2
  GROSS   94   4  P  2
  DEPT#   98   3  N
  DEDUCTIONS W  4  P  2
*
* JOB INPUT PAYFILE
  IF NAME   EQ  'ANDERSON', 'BAKER', 'CARROLL'
  IF NET    NE  GROSS, EMPL#, DEDUCTIONS, 9999.9
  IF EMPL#  EQ  10555, 11555, 12550, 15550, 15555
  IF DEDUCTIONS NE  GROSS, NET, 999, 1111.34
*
* JOB INPUT  PAYFILE
  IF NAME   EQ  'ANDERSON' THRU 'CARROLL'
  IF DEPT#  EQ  911 THRU 921
  IF EMPL#  NE  10555 THRU 15555
  IF NET    NE  DEDUCTIONS THRU GROSS
*
```

### File Presence Condition

This condition determines if a record of the named input file is available for processing. It is discussed in detail in Chapter 9, “File Processing.”

### File Presence Series Condition

This condition is used in a JOB with synchronized file processing to determine whether or not the records from more than one file have the same key. This condition is discussed in detail in Chapter 9, “File Processing.”
Record Relational Condition

This condition is used in a JOB with synchronized file processing to test for duplicate records within one file. The current record of the named file is compared to the previous and next records of the same file. This condition is discussed in detail in Chapter 9, “File Processing.”

IF, ELSE, and END-IF Statements

These three statements are used together. For every IF statement, you must also provide an END-IF statement.

Code the ELSE statement to take alternate measures in the case of a condition testing false. ELSE is not used in any other context than with the IF statement.

The next exhibit presents portions of the sample program which illustrate the use of these statements.

IF Statement

The IF statement controls the execution of subsequent CA-Easytrieve/Plus statements which are associated with it. As a general rule, these associated statements should be indented below the IF statement so their relationship is immediately noticeable.
In the above exhibit, the first IF statement contains a field relational conditional expression which tests to see if the value of field SERVICE is less than 1.

- If this condition is true, the next statement (GO TO JOB) returns control to the JOB statement, where the next input record is read.
- If the condition is not true (SERVICE is 1 or greater), the GO TO statement is not executed and execution continues with the statement following END-IF (PERFORM RAISE-CALC).

The GO TO statement is discussed later in this chapter.

**ELSE Statement**

The ELSE statement identifies CA-Easytrieve/Plus statements which are to be executed when the result of the condition test in the IF statement is false. The second IF statement in the exhibit shown previously contains a field relational condition which tests to see if DEPT is less than 940.

- If this is true, a raise is calculated at 10 percent of SALARY. The statement following ELSE is bypassed and execution resumes with the statement following END-IF.
- If the condition is not true, that is, the value in field DEPT is not less than 940, the statement between IF and ELSE (the Assignment statement calculating RAISE at 10 percent) is bypassed and the statement following the ELSE (the Assignment statement calculating RAISE at 15 percent) is executed. Execution then continues with the statement following END-IF.

**END-IF Statement**

The END-IF statement terminates the processing associated with the IF statement. The END-IF statement indicates the end of the IF construct, and the statement following the END-IF (in the exhibit shown previously, PERFORM RAISE-CALC) is the next statement to be executed.

**Nesting IF Statements**

Whenever one or more statements following an IF statement is another IF, the IFs are considered to be nested. The format of nested IFs is simply that any statement following an IF can be another IF statement. All IFs must be terminated by an END-IF.
DO and END-DO Statements

These statements are called loop control statements and, with conditional expressions, are used to control repetitive program tasks. The next exhibit provides a brief illustration of the use of these statements.

```plaintext
* FILE PAYFILE
  REC-KEY 1 3 N
*
JOB INPUT NULL
  GET PAYFILE
  DO WHILE (REC-KEY > 500, AND REC-KEY < 600, +
  AND NOT EOF PAYFILE)
    PRINT PAY-RPT
    GET PAYFILE
  END-DO
  STOP
*
REPORT PAY-RPT LINESIZE 80
...```

DO Statement

This statement identifies one or more CA-Easytrieve/Plus statements which are to be executed WHILE the conditional expression tests true. When the condition tests false, the statements are bypassed. The conditional expression must have the possibility to be false eventually or DO loops forever. In the above exhibit, each record of file PAYFILE is read. While the value of the key is between 501 and 599, the record is output to report PAY-RPT. Otherwise, the job is terminated by the STOP statement.

END-DO Statement

This statement terminates the loop processing when the condition in the DO statement tests false. Execution branches to the next executable statement following the END-DO statement.
Nesting DO Loops

Any of the statements following the DO can also be a DO statement. You must take care to close inner loops in proper sequence, as illustrated in the next exhibit.

```
DO ...
  DO ...
    DO ...
    END-DO
END-DO
END-DO
```

GOTO Statement

This statement causes an immediate branch out of the normal top-to-bottom flow of program execution. Its format is diagrammed in the next exhibit.

```
{ GOTO } { label }
{ GO TO } { JOB }
```

If the statement specifies JOB, execution control is transferred immediately to the first executable statement of the current JOB activity. If the GOTO specifies a statement label, execution control is transferred immediately to the first executable CA-Easytrieve/Plus statement following that label; processing continues at that location. The specified label must be located in the same activity or procedure.
Statement Labels

Statement labels are names that you can code to identify the destination of a GOTO statement. They are subject to the same restrictions as field names; that is, they must start with a letter, can be up to 40 characters long, and can be composed of letters, digits, and some special characters.

Not all CA-Easytrieve/Plus statements can be labeled. Following is a list of statements which can be preceded by labels:

- Assignment
- DISPLAY
- DLI
- DO
- END-DO
- END-IF
- END-PROC
- GET
- IF
- PERFORM
- POINT
- SELECT
- PRINT
- PUT
- READ
- SEARCH
- STOP
- WRITE

The next exhibit presents portions of the sample program which illustrate the use of the GOTO statement.

```plaintext
22 *
23 JOB INPUT PERSNL
24 %GETDATE CURR-DATE
42 SALARY = PAY-GROSS * 52
43 PERFORM SERVICE-CALC
44 IF SERVICE LT 1
45 GO TO JOB
46 END-IF

54 *
55 SERVICE-CALC. PROC
57 SERVICE = CURR-YY - HIRE-YY
58 IF CURR-MM < HIRE-MM
59 SERVICE = SERVICE - 1
60 END-IF
61 IF CURR-MM NE HIRE-MM
62 GOTO QUIT-SERV-CALC
63 END-IF
64 IF CURR-DD < HIRE-DD
65 SERVICE = SERVICE - 1
66 END-IF
67 QUIT-SERV-CALC
68 END-PROC
69 *
```
Procedure Processing

A procedure is a set of CA-Easytrieve/Plus statements that are grouped together to accomplish a task. Once you have created a procedure and given it a name, you can reference it in your program by name without having to repeat the lines of code each time you want to execute them.

Procedures are defined using the PROC and END-PROC statements. They are invoked from within your program with the PERFORM statement.

PROC and END-PROC Statements

These statements identify the beginning and end of a procedure. Their format is diagrammed in the next exhibit.

proc-name. PROC
  (Statement 1)
  ...
  (Statement n)
END-PROC

The procedure is constructed as follows:

Proc-name

The name you assign to the procedure. It must start with a letter, can be up to 40 characters long, and can include letters, numbers, and some special characters. This name must be followed by a period, a space, and the keyword PROC.

Statement 1 through Statement n

The CA-Easytrieve/Plus statements which accomplish the procedure's task. There is no restriction on the statements or commands that you can code in the procedure (with the exception of input/output statements which cannot be included in a procedure invoked during SORT or REPORT processing).

END-PROC

This keyword terminates the procedure and returns control to the point in your program where the procedure was invoked.

Code any procedures immediately after their associated activity (JOB or SORT) or subactivity (REPORT). Procedures which you define are invoked by PERFORM statements. In addition, there are special-name report procedures which are used in report processing (see Chapter 8, “Report Processing”).
PERFORM Statement

PERFORM transfers execution control to the procedure named in this statement. Its format is diagrammed in the next exhibit.

\[
\text{PERFORM } \text{proc-name}
\]

Execution of this statement results in an immediate branch to the named procedure. When processing of the procedure is complete, control returns to the statement following the PERFORM statement.

The sample program contains three procedures (SERVICE-CALC, RAISE-CALC, and BONUS-CALC) which are executed by PERFORM statements in the JOB activity.

STOP Statement

This statement terminates CA-Easytrieve/Plus activities. You can use it for premature termination of activities using automatic input. The STOP statement must be used to terminate JOB activities which have INPUT NULL. This statement is diagrammed in the next exhibit.

\[
\text{STOP } [\text{EXECUTE}]
\]

EXECUTE

The EXECUTE parameter immediately terminates the current activity and any subsequent activities. If you do not code this parameter, only the current activity is terminated. The next exhibit illustrates the use of the STOP statement in a revised version of one of the procedures from the sample program.

\* 
\* BONUS-CALC. PROC
  IF SALARY GT 29999
    DISPLAY ERRPRINT, LAST-NAME, +5, +
    'INELIGIBLE FOR BONUS'
    STOP
  END-IF
  IF SERVICE GT 19
    BONUS = 2000
  ELSE
    BONUS = 1000
  END-IF
  PRINT BONUSRPT
END-PROC
\*
If you want to stop the current activity and bypass all subsequent activity, use the EXECUTE parameter. The next exhibit provides an example of complete termination.

```
... WRITE PAYFILE, STATUS IF PAYFILE:FILE-STATUS NE 0  
    DISPLAY 'I/O ERROR ON WRITE' STOP EXECUTE
END-IF
...```

Introduction

For most applications, the ability of CA-Easytrieve/Plus to control your input and output is quite satisfactory. For more complex jobs, however, you can control it yourself.

CA-Easytrieve/Plus provides three levels of input/output (I/O):

- Automatic
- Controlled
- Database.

Automatic I/O

Automatic I/O provides for the automatic sequential reading of a data file and the production of one or more reports. The statements which support this level of I/O are:

- JOB which specifies the input file
- PRINT which initiates report output
- DISPLAY which produces printed output not directly supported by a report (for example, error messages).
Controlled I/O

Controlled I/O provides the capability to process any sequential or keyed file (ISAM or VSAM). These statements require a comprehensive understanding of the file structure in use. The controlled I/O statements are:

- GET which sequentially reads one record
- POINT which positions a keyed file to a particular record for subsequent sequential I/O
- PUT which sequentially writes one record
- READ which reads one keyed record
- WRITE which rewrites, adds, or deletes one keyed record.

Database I/O

The most complex level of input/output involves the use of databases. Refer to Chapter 11, “IMS/DLI Processing.”

This chapter presents brief descriptions and examples of the statements used in automatic and controlled I/O. For an extensive discussion of their use, see Chapter 9, “File Processing.”

**DISPLAY Statement**

Use the DISPLAY statement to output data to the system printer or a named file. This data is spaced according to the specified parameters. Its format is diagrammed in the next exhibit.

```
[                   ] [ literal-3      ]
[ NEWPAGE           ] [ field-name     ]
DISPLAY [file-name] [ SKIP literal-1    ] [ +literal-4     ]
[ CONTROL literal-2 ] [ -literal-4     ]
[                   ] [  COL literal-5 ]
```

[file-name]

This parameter is optional. If it is specified, it names the file which is the destination of the DISPLAYed data. This can be any file-name you specified in your program's library section, however, the PRINTER parameter must be included on the FILE statement. If you do not code a name, the default is SYSPRINT (SYSLST for VSE).

DISPLAY ERRPRINT
Specify a unique file-name to avoid interspersing DISPLAY output with an unsequenced report. This is especially useful for error messages. If the error message file is printed prior to the report, you can use it to determine if the report should be printed or if a severe error occurred which makes the report output invalid.

The NEWPAGE option specifies a skip to a new page before the data is printed. The SKIP option specifies the number of lines (literal-1) to be skipped before the data is printed. The CONTROL option sets the printer carriage control character for the print line. Valid alphabetic values for literal-2 are 0 through 9, +, -, A, B, or C. CONTROL is not valid in REPORT procedures. No automatic page skipping is provided by the DISPLAY statement. It is your responsibility to issue a DISPLAY NEWPAGE when you reach the bottom of the page.

DISPLAY ERRPRINT SKIP 10

Content and Spacing Parameters

The data to be displayed is specified by either a field-name or literal-3. You can code as many of these as you like, in the order you want them to appear on the printed line. The only limitation is that the data must fit on a single print line. The first data entry appears in column one of the print line. The first character of each additional item immediately follows the last character of the preceding one. No spaces are left between items unless specified by additional options: +literal-4, -literal-4, or COL literal-5.

DISPLAY ERRPRINT, DEDUCTIONS, GROSS

The option for adjustment of the horizontal spacing between displayed items is +literal-4 or -literal-4 counted in character positions. For instance, +5 specifies five spaces between the last item and the next (DEDUCTIONS and GROSS); -3 specifies that the next item is three spaces to the left of where it would otherwise print. The value of literal-4 can be any amount that does not extend your data beyond the end of the line to be printed.

DISPLAY ERRPRINT, DEDUCTIONS, +5, GROSS
The COL option specifies precisely where your data is placed on the print line. Literal-5 specifies the column number where the first character of the next data item appears, counting from the left of the page. In the following example, DEDUCTIONS starts in column 1 and GROSS starts in column 40. Each character position is one column. The value of literal-5 can be any amount that does not extend your data beyond the end of the print line.

DISPLAY ERRPRINT, DEDUCTIONS, COL 40, GROSS

Rules for Use

The DISPLAY statement sends a line to the printer as soon as the statement is executed. For this reason, you must take care how you use it in your program. If your program produces an unsequenced report, each PRINT statement in your JOB activity sends a line of its associated report to the printer, after being formatted according to the report declaratives.

Unless otherwise specified, any data in DISPLAY statements within your JOB activity goes directly to the printer and is interspersed with the lines of your report. If your report is sequenced, all of the DISPLAYed data precedes all of the PRINTed data. The DISPLAYed data goes directly to the printer, but the PRINTed data is spooled until the JOB activity processing is finished.

Some typical uses of the DISPLAY statement include:

- Printing specially formatted lines in a report which are outside the capabilities of the REPORT declaratives. This should be done in a procedure coded at the end of the REPORT declaratives. The procedure is executed at the time the report data is formatted. The DISPLAYed data appears in the place you want.

- Printing error messages when abnormal conditions are encountered. You can avoid interspersing displayed data with your report data by specifying the file-name option on the DISPLAY statement, as illustrated within the BONUS-CALC procedure in the Sample Program under the Application topic in Chapter 1, “Overview.”

Debugging

You can use a special format of the DISPLAY statement to produce a hexadecimal and character dump of a specified field-name or of the current record of a specified file-name. This can be very useful for debugging, as illustrated in the next exhibit.

```
[ NEWPAGE        ] [ HEX field-name ]
DISPLAY [file-name] [ ] [ ]
           [ SKIP literal-1 ] [ file-name ]
```
PRINT Statement

The PRINT statement initiates report output by causing the named report to extract the current values of the fields to be output and to format them according to the specifications in the report declaratives. The report can be printed immediately or deferred.

- If the report is not sequenced, the PRINT statement outputs data to a print file from which the report is produced immediately.
- If the report is sequenced, or if another report is already using the associated print file, the PRINT statement outputs data to a work file which is spooled until the associated JOB activity processing is complete.

The next exhibit diagrams the format of the PRINT statement.

```
PRINT  report-name
```

The report-name parameter is the name of the report which contains the data being output with the PRINT statement.

**Note:** It is important to understand the sequence of events initiated by the PRINT statement. In any CA-Easytrieve/Plus program, the next statement to be executed after PRINT is the associated REPORT statement. CA-Easytrieve/Plus immediately extracts the data required for the report, formats it in the specified manner and, if the report is not sequenced, outputs this data to the printer. Execution then resumes with the statement immediately following the PRINT statement.

Refer to the *CA-Easytrieve/Plus Reference Guide* for a detailed discussion of the ways to use this debugging aid.
The next exhibit illustrates this process.

- **Print**
  - **Collect Current Data**
    - **Spooled?**
      - Yes: **Write Workfile Record**
      - No: **Format Output**
        - **Print Report**
          - **Return**

* Issue Print Statement.

* Extract current values from specified data fields.

* If print file is in use or if the report is sequenced, spool output to workfile.

* Format data according to REPORT specifications.

* Print output on report.

* Return to JOE Activity.
If the report is sequenced, the data is output to a work file which is sorted before the report is printed. The next exhibit illustrates this process.
PRINT Statement

Report Wrap-Up

- Was the print file in use?

Report Spooled?

- Yes

- Sort Workfile

- If report is sequenced, sort workfile per specifications.

- Format data according to REPORT specifications.

- Print output on report.

Report Sequenced?

- No

- End of Work File?

- Yes

- Report Final Processing

- Flush label line. Final control break process. Endpage and termination.

- Return to JOB Activity.

- No

Format Output

Print Report

- Is this the last record?
GET Statement

The GET statement makes the next sequential record of the named file available for processing. Its format is diagrammed in the next exhibit.

```
GET  file-name
```

file-name

The file-name parameter is required. It can be any file defined in the library section of your program. See the POINT statement which also provides an example of the GET statement.

PUT Statement

The PUT statement outputs data to a sequential file whose name is specified in the statement. The format is diagrammed in the next exhibit.

```
PUT  file-name-1  [FROM file-name-2]
```

PUT creates new sequential files (SAM, VFM, VSAM), or adds consecutive records to an existing VSAM file.

file-name-1

This parameter names the output file being created or being added to. This file must be defined in the library section of your program.

FROM file-name-2

This parameter is optional. If it is provided, PUT copies the current record of file-name-2 to file-name-1. If the record lengths are not the same, the length of the record from file-name-2 is adjusted to fit the record length specified for file-name-1.

Example

The next exhibit illustrates the use of the PUT statement.

```
JOB  START  POINTER  INPUT  PAYFILE
   IF  REC-KEY  GE  600
      STOP
   END-IF
   SALARY  =  SALARY  *  1.1
   PUT  SALUPD  FROM  PAYFILE
   *  PRINT  UPD-RPT
   *  POINTER.  PROC
      POINT  PAYFILE  GE  500
   END-PROC
   *
```
In the above exhibit, the statements retrieve those records with keys between 500 and 599 inclusive from file PAYFILE, increase the value in the SALARY field of each record by 10 percent, and output each updated record to file SALUPD and report UPD-RPT.

**POINT Statement**

The POINT statement initiates a search for a position within an indexed or relative-record file, based on a comparison between keys in the file and a search value specified in the statement. Its format is diagrammed in the next exhibit.

```
{EQ} {= } {field-name}
POINT  file-name {  } {          }
{GE} {literal   }
{>=}
```

The POINT statement only locates the specified position of the record in the file. You must still issue a GET statement to retrieve the data for processing.

**file-name**

This must be the name of a file with an indexed or relative-record filetype (IS or VS).

**Relational Operator**

The equal operator (EQ or =) directs CA-Easytrieve/Plus to search for an exact match between a key in the file and the search value specified in the POINT statement. An error results if the exact match is not found. The greater-than-or-equal operator (GE or >=) searches for a key in the file which is equal to or greater than the specified search value; a condition which is more easily satisfied.

**Search Value Parameters**

```
{literal }  
{ }  
{field-name}
```

These parameters can be any literal or any field-name defined in your library. If the search value is higher than any key in the file, the file presence conditional expression IF EOF file-name tests true.
The next exhibit illustrates the use of the POINT statement.

```
FILE PAYFILE VS ...
  REC-KEY 1 3 N
*
JOB INPUT NULL
  POINT PAYFILE GE 500
  GET PAYFILE
  DO WHILE (REC-KEY < 600, AND NOT EOF PAYFILE)
    GET PAYFILE
    PRINT PAY-RPT
    END-DO
  STOP
*
REPORT PAY-RPT ...
...
```

The statements in the above exhibit retrieve those records with keys between 500 and 599 inclusive from file PAYFILE and output them to report PAY-RPT.

**READ Statement**

The READ statement provides random access to keyed and relative-record VSAM and ISAM files. Its format is diagrammed in the next exhibit.

```
READ  file-name  KEY field-name  [STATUS]
```

- **file-name**
  
  This parameter identifies the file you want to access. It must have been defined as a VSAM or ISAM file in your program's library section.

- **KEY field-name**
  
  This parameter serves as a search value to identify the specific record to be retrieved. The contents of the specified field-name must match the contents of the key of the desired record.

- **[STATUS]**
  
  This parameter is optional. If you include it, execution of the READ statement sets a return code in the FILE-STATUS field of your input file to indicate the success or failure of the operation. A successful READ returns a value of 0, any other value is a code identifying the reason for failure. Check with your data center to learn the meaning of the codes in this field. They are explained in an IBM manual about your system.
The next exhibit illustrates the use of the READ statement.

```plaintext
FILE PAYFILE VS UPDATE
   EMPL# W 5 N
   NAME 6 20 A

* 
JOB INPUT NULL
   EMPL# = 44152
   READ PAYFILE, KEY EMPL#, STATUS
   IF FILE-STATUS NOT ZERO
      GOTO ERRTASK
   END-IF
   IF NAME EQ 'OLDNAME,M.'
      NAME EQ 'NEWNAME,M.'
      WRITE PAYFILE UPDATE
   ELSE
      GOTO ERRTASK
   END-IF
   STOP
*
```

The statements in the above exhibit search file PAYFILE for a record whose key matches the value in EMPL#. PAYFILE is keyed by employee number. The value of EMPL# is 44152.

- If the READ is not successful, execution branches to the location labeled ERRTASK.
- If the NAME field of this record is equal to 'OLDNAME,M.', the NAME field is changed to 'NEWNAME,M.' and the record is written back to PAYFILE.
- If the name comparison tests false, execution branches to ERRTASK, bypassing the Assignment and WRITE statements.

## WRITE Statement

Use the WRITE statement to maintain keyed and relative-record VSAM files (ISAM files are read-only). WRITE updates or deletes the current record of the named file, or adds new records. Its format is diagrammed in the next exhibit.

```
[DELETE]
WRITE  file-name-1  [UPDATE]  [FROM file-name-2]
   [ADD   ]
```

`file-name-1`

This parameter names the file to be modified. It must have been coded in the FILE statement with the UPDATE subparameter included.

```
[ DELETE ]
[ UPDATE ]
[ ADD   ]
```

These parameters specify the maintenance activity to be performed. They are required for deleting or adding records. It is optional for an update activity. The default is UPDATE if this parameter is not coded.
[FROM file-name-2]

This parameter is optional. If it is included, the WRITE statement copies the current record of file-name-2 to file-name-1 for either an UPDATE or an ADD operation. This parameter is not valid for a DELETE operation.

If the record lengths are not the same, the length of the record from file-name-2 is adjusted to fit the record length specified for file-name-1.

The READ statement exhibit, shown earlier, also presents an example of the WRITE statement.
Introduction

The most noticeable thing about CA-Easytrieve/Plus report processing is how easy it makes the task of producing reports. You can design your reports any way you prefer, such as to set up column headings, to request different types of information, and to decide which kinds of totals you want.

You have to tell CA-Easytrieve/Plus what you have decided by using a few easy-to-remember English words. These words are either coded on the REPORT statement as parameters or immediately follow the REPORT statement as subsequent but related statements. These are called report declaratives.

You can let the CA-Easytrieve/Plus report processor handle the details for you or you can choose to specify every detail of your report to describe the data you want reported and the appearance of the printed result. This facility is so powerful and easy to use that no special programming skill is required.

Within the JOB activity section of your program, the statements which send data to reports are:

- The PRINT statement, which initiates the report facility
- The DISPLAY statement, which produces single print lines.

Both of these CA-Easytrieve/Plus statements are described in detail in Chapter 7, “Input/Output Specification.” The discussion of CA-Easytrieve/Plus automatic report processing in this chapter uses the PRINT statement exclusively.
The desired reports are defined by a set of statements at the end of the JOB activity. These statements specify the report type, format, sequence, and content, as follows:

```
REPORT
   SEQUENCE
   CONTROL
   TITLE
   HEADING
   LINE
   report procedures
   BEFORE-BREAK
   TERMINATION
```

You must code these statements in the order listed above.

You can generate as many reports as you like from a JOB activity. The Sample Program produces the Sample Update Report and Sample Bonus Report (illustrated in Chapter 2, “Overview,” under the topic Application). The report declarative portion of the sample program is illustrated below.

```
Sample Program Report Declaratives

93 *
94 REPORT UPD-RPT PAGESIZE 51 LINESIZE 63 NODATE NOPAGE
95   SEQUENCE DEPT LAST-NAME
96   CONTROL DEPT
97   TITLE 1 'ANNUAL UPDATE REPORT - SALARIED EMPLOYEES'
98   HEADING LAST-NAME 'NAME'
99   HEADING SERVICE 'SERV'
100  LINE DEPT LAST-NAME SERVICE RAISE SALARY
101 *
102 REPORT BONUSRPT LINESIZE 60 NODATE NOPAGE
103   SEQUENCE DEPT LAST-NAME
104   TITLE 1 'ANNUAL BONUS REPORT - SENIOR EMPLOYEES'
105  LINE DEPT LAST-NAME SERVICE BONUS
106 *
```

The first report specified in the above exhibit is described in seven lines of code which supply the following information:

- The report name is UPD-RPT. Each page of the printed output is 51 lines long and 63 columns wide. Neither the date nor the page number is printed on the first title line of each page.
- The report is ordered (sequenced) by two levels: first, by department number and, within each department, in order by last name.
- The dollar values are subtotaled for each department, and the report is segmented by department.
- The title ANNUAL UPDATE REPORT - SALARIED EMPLOYEES is centered across the top of the report page.
The column heading for field LAST-NAME reads NAME, and for field SERVICE reads SERV.

There are five columns spaced three characters apart across the 63-character-wide report. The columns contain the data in the fields: DEPT, LAST-NAME, SERVICE, RAISE, and SALARY in that order from left to right.

The result of this specification is the Sample Update Report which is illustrated in Chapter 2, “Overview,” under the topic Application.

Report Types

There are two basic report formats: standard format and label format. The reports produced by the sample program are standard format reports. Label format reports include mailing labels, form letters, and other special-purpose reports.

Standard Reports

The default is the standard format illustrated below.

```
| LINESIZE |

<table>
<thead>
<tr>
<th>TOP MARGIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE AREA (Optional)</td>
</tr>
<tr>
<td>HEADING AREA (Optional)</td>
</tr>
<tr>
<td>REPORT BODY</td>
</tr>
<tr>
<td>BOTTOM MARGIN</td>
</tr>
</tbody>
</table>
```
TOP MARGIN

The top margin is the space between the physical top of the form and the point to which the printer positions the paper when a top-of-form order is issued to the printer. The size of the top margin is controlled by the printer carriage tape or forms control buffer.

TITLE AREA

The Title Area consists of 1 to 99 optional title lines plus the blank lines, usually three, between the last title line and the first heading line.

HEADING AREA

The Heading Area consists of 1 to 99 optional heading lines plus a blank line between the last heading line and the report body.

REPORT BODY

The Report Body consists of one or more line groups. Each group consists of 1 to 99 lines plus, optionally, one or more blank lines between line groups.

BOTTOM MARGIN

The bottom margin is the area remaining between the bottom of the report body and the physical bottom of the page.

The CA-Easytrieve/Plus default values for report spacing are sufficient for most applications.

Label Reports

The second report format prints a variety of reports, an example of which is mailing labels. The structure of this report is illustrated below.

```
1  2  3  4
down
5  6  7  8
size
linesize
4 across
```
Each individual label is one line group. Each PRINT statement in the JOB activity of your program produces one label, formatted on the lines of that label according to the report declaratives. The DOWN and SIZE parameters specify the label dimensions.

**REPORT Statement**

REPORT is the first statement of the report declaratives. It establishes the type and characteristics of your report. Although there are several parameters available which provide a flexible capability to tailor your reports, you can probably produce most reports using default parameter values.

REPORT statement parameters fall into four categories:

- Format determination
- File directing
- Spacing control
- Testing aids.

The syntax of the REPORT statement is diagrammed below.

```
REPORT  report-name +
    [SUMMARY]   +
    LABELS ([ACROSS literal-1] + Format
            [DOWN literal-2] + Determination
            [SIZE literal-3] +
            [NEWPAGE])       +
    [PRINTER file-name] + File Directing
    [PAGESIZE literal-4] +
    [LINESIZE literal-5] +
    [SPREAD] + Spacing
    [NOSQUEADJUST] + Control
    [NODATE] +
    [NOPAGE] +
    [LIMIT literal-6] + Testing
    [EVERY literal-7] + Aids
```

**report-name**

REPORT  report-name

This parameter names the report. The report-name can be from 1-to-40 characters long and must start with a letter. It is unique within each JOB activity and is correlated with matching entries on PRINT report-name statements. In the Sample Program Report Declaratives, shown earlier, the report-names are UPD-RPT and BONUSRPT.
REPORT Statement

[SUMMARY]

This option inhibits printing of detail data on CONTROL reports - only totals are printed. Since CA-Easytrieve/Plus totals only quantitative fields (those fields which are defined as having decimal positions), SUMMARY produces a report with entries only in the control fields and the fields which are totaled. For appearance, the LINE statement should contain only these field-names. If the LINE statement contains names of fields which are not totaled, the headings print with no entries under them.

LABELS

LABELS ((ACROSS literal-1) +
[DOWN literal-2] +
[SIZE literal-3] +
[NEWPAGE])

This option defines your report as having the label format. The associated subparameters control the spacing of the labels on the report page (see the Label Reports exhibit shown earlier).

ACROSS literal-1

Specifies the number of labels printed side-by-side across the page.

DOWN literal-2

Specifies the number of print lines on each label. The value of literal-2 is the number of print lines between the first line of each label (including any physical space between labels).

SIZE literal-3

Specifies the width of each label, counted in print positions from left to right. The value of literal-3 is the number of print positions between the first character of each label (including any physical space between labels).

NEWPAGE

Directs the printer to print the first line of each label at the top of a page.

The overall width of labels on a report page is constrained by the following formula:

\[
\text{LINESIZE} \geq (\text{ACROSS} - 1) \times \text{SIZE} + \text{(number of print positions on an individual label)}
\]

The exhibit below illustrates a modification of the sample program which produces a set of labels. The next exhibit, Labels Produced by Mailing Label Programs, illustrates the output produced by this program.
**Mailing Label Program**

```sql
1 PARM DEBUG(FLOW FLDCHECK)
2 *
3 FILE PERSNL FB(ISO 1808)
4 NAME 17 16 A
5 LAST-NAME 17 8 A
6 FIRST-NAME 25 8 A
7 ADDRESS 37 39 A
8 STREET 37 20 A
9 CITY 57 12 A
10 STATE 69 2 A
11 ZIP 71 5 N
12 DATE-OF-HIRE 136 6 N
13 HIRE-MM 136 2 N
14 HIRE-DD 138 2 N
15 HIRE-YY 140 2 N
16 SERVICE W 2 N
17 CURR-DATE S 6 N
18 CURR-MM CURR-DATE 2 N
19 CURR-DD CURR-DATE +2 2 N
20 CURR-YY CURR-DATE +4 2 N
21 *
22 JOB INPUT PERSNL
23 %GETDATE CURR-DATE
24 PERFORM SERVICE-CALC
25 IF SERVICE GT 19
26 PRINT MAILOUT
27 END-IF
28 *
29 SERVICE-CALC. PROC
30 SERVICE = CURR-YY - HIRE-YY
31 IF CURR-MM < HIRE-MM
32 SERVICE = SERVICE - 1
33 END-IF
34 IF CURR-MM NE HIRE-MM
35 GOTO QUIT-SERV-CALC
36 END-IF
37 IF CURR-DD < HIRE-DD
38 SERVICE = SERVICE - 1
39 END-IF
40 QUIT-SERV-CALC
41 END-PROC
42 *
43 REPORT MAILOUT LABELS (ACROSS 2 DOWN 4 SIZE 30)
44 SEQUENCE LAST-NAME
45 LINE 1 FIRST-NAME -3 LAST-NAME
46 LINE 2 STREET
47 LINE 3 CITY -3 STATE ZIP
48 *
```
REPORT Statement

Labels Produced by Mailing Label Programs

NANCY BERG  PATTI HUSS
3710 JENIFER ST NW  1355 TOWKESBURY PLAC
BALTIMORE MD 21055  CLEARWATER FL 33712

ALFRED JONES  MAX KRUSE
2070 BELMONT ROAD NW  2161 N PIERCE STREET
LOS ANGELES CA 90052  ATLANTA GA 30345

NED LOYAL  TERRY MALLOW
17 KENNEDY STREET  2515 K STREET NW APT
RALEIGH NC 27516  MINNEAPOLIS MN 55329

SAMUEL OSMON  KATHY PETRIK
4281 CATHEDRAL AVE N  5005 BENTON AVE
CHICAGO IL 60618  WASHINGTON DC 20032

CAROL POWELL  WILLIAM REYNOLDS
5023 AMES STREET N E  4126 CROSSWICK TURN
ATLANTA GA 30316  DALLAS TX 75244

PAT ROGERS  CINDY SMOOTH
1625 FRANKLIN ST N E  4120 18TH STREET NE
CHICAGO IL 60618  DALLAS TX 75219

DENISE VETTER  GLORIA WIMN
7311 KEYSTONE LANE 4  430 M ST SW 107
RALEIGH NC 27591  BOSTON MA 02005

[PRINTER file-name]

This option identifies a file-name other than the default as the destination of the printed report. The default is SYSPRINT for OS/390 and SYSLST for VSE. If a file-name is specified, the PRINTER parameter must be specified on the associated FILE statement.

[PAGESIZE literal-4]

This option specifies a value for the top-to-bottom length of the printed report page. The value of literal-4 sets the number of print lines per page. At least one line group must fit on a report page. The default is commonly 58 lines. Check with your data center to determine the default value for your installation.

[LINESIZE literal-5]

This option specifies a value for the left-to-right width of each line of your report. The value of literal-5 is the number of print columns on each report line. The maximum you can specify is one character less than the physical length (record size) of the printer file receiving the report. The default is commonly 132 characters, which is one less than the actual size of the typical printer file record (133 characters). The first character is used for vertical form control (carriage control). Check with your data center to determine the default value for your installation.
LINESIZE must be able to accommodate the maximum size of all the fields listed across your report, including extra characters for totals when requested. A LINESIZE of 60 or 63 characters is specified for the reports in the sample program to enable them to fit on the pages of this guide.

[SPREAD ]
[NOSPREAD]

This option adjusts the spacing of the columns of your report. SPREAD directs CA-Easytrieve/Plus to maximize the number of spaces between columns. NOSPREAD deactivates the SPREAD option. In most cases, NOSPREAD is the default, which puts three characters between columns and centers the report on the printer page.

[NOADJUST]

This option left-justifies your report on the printer page. Centering is usually the default.

[NODATE]

This option suppresses printing of the date in the leftmost eight columns of the first line of the report title. This is useful with NOADJUST, since without it the date overprints the first eight characters of the report title.
REPORT Statement

[NOPAGE]

This option suppresses printing of the characters PAGE and the page number in the rightmost 11 columns of the first report title line.

Modification of the first REPORT statement and the associated LINE statement in the sample program to include several of these format determination and spacing control parameters, as follows:

```
REPORT UPD-RPT SUMMARY LINESIZE 60 SPREAD NODATE NOPAGE
... LINE DEPT RAISE SALARY
```

produces the SUMMARY Report illustrated below.

```
ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

<table>
<thead>
<tr>
<th>DEPT</th>
<th>RAISE</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>2,284.80</td>
<td>24,252.80</td>
</tr>
<tr>
<td>903</td>
<td>1,942.72</td>
<td>23,369.92</td>
</tr>
<tr>
<td>911</td>
<td>18,187.10</td>
<td>208,858.14</td>
</tr>
<tr>
<td>912</td>
<td>1,535.04</td>
<td>18,885.44</td>
</tr>
<tr>
<td>914</td>
<td>12,497.47</td>
<td>141,472.19</td>
</tr>
<tr>
<td>915</td>
<td>760.03</td>
<td>8,368.35</td>
</tr>
<tr>
<td>917</td>
<td>2,559.75</td>
<td>29,157.27</td>
</tr>
<tr>
<td>918</td>
<td>5,798.20</td>
<td>63,780.28</td>
</tr>
<tr>
<td>919</td>
<td>786.42</td>
<td>8,776.62</td>
</tr>
<tr>
<td>920</td>
<td>1,638.72</td>
<td>17,937.92</td>
</tr>
<tr>
<td>921</td>
<td>3,979.82</td>
<td>47,778.62</td>
</tr>
<tr>
<td>923</td>
<td>2,916.16</td>
<td>33,977.76</td>
</tr>
<tr>
<td>924</td>
<td>2,360.80</td>
<td>27,968.80</td>
</tr>
<tr>
<td>931</td>
<td>71.76</td>
<td>1,789.36</td>
</tr>
<tr>
<td>932</td>
<td>2,062.73</td>
<td>22,989.09</td>
</tr>
<tr>
<td>935</td>
<td>6,148.48</td>
<td>67,633.28</td>
</tr>
<tr>
<td>940</td>
<td>15,534.94</td>
<td>119,101.26</td>
</tr>
<tr>
<td>942</td>
<td>7,762.56</td>
<td>61,512.96</td>
</tr>
<tr>
<td>943</td>
<td>13,547.04</td>
<td>104,860.64</td>
</tr>
<tr>
<td>944</td>
<td>6,121.44</td>
<td>47,931.04</td>
</tr>
<tr>
<td></td>
<td>108,327.98</td>
<td>1080,388.14</td>
</tr>
</tbody>
</table>
```

[LIMIT literal-6]

This option specifies the number of PRINT statements accepted for this report and is useful for testing. The value of literal-6 sets the maximum number of lines desired. For example, you could limit the output of your report to the first 50 PRINT statements to make sure your column spacing is what you want.

[EVERY literal-7]

This option is also used for testing. Literal-7 specifies the occurrence value for processing every Nth PRINT command directed to the report. If you specified LIMIT 50 EVERY 10, you could sample the output from the first 500 PRINT statements for your report.
SEQUENCE Statement

This optional statement specifies the order in which you want the contents of your report to appear. If you do not specify SEQUENCE, the data appears on your report in the same order as it appears in the records of the input file. You can order any report on the contents of one or more fields in the input file or in working storage. These fields do not have to be output to the printed report. The syntax of the SEQUENCE statement is illustrated below.

SEQUENCE field-name-1 [D] [field-name-2 [D]] ... 

field-name

This parameter identifies the field(s) on which your report is ordered. If you specify more than one field, the sequencing is done in the order specified. For example, the first report in the sample program is sequenced first by department number (DEPT) and, within departments, by the last name of the employees (LAST-NAME):

SEQUENCE DEPT LAST-NAME

Inclusion of the optional D following a field-name indicates that the field is sequenced in descending order. The default is ascending order.

CONTROL Statement

This optional statement identifies the field-name(s) on which you want your report controlled. Also, it enables you to specify certain optional results of the control break processing. One result of controlling a report is to produce subtotals of the values in fields which have been specified as having decimal positions. In the sample program, both reports are controlled on department number. A control break occurs each time the value in field DEPT changes and at end-of-report, producing a subtotal of the dollar values in the RAISE and SALARY fields for each department and final totals at the end of the report. The syntax of the CONTROL statement is diagrammed below.

[FIELD-NAME] [NEWPAGE] 
CONTROL [ ] [ ] [NOPRINT] ... 
[FINAL ] [RENUM ]

These parameters identify the field(s) on which you want your report controlled. This can be any defined field in your input file or working storage. Code the FINAL parameter before the first field-name (if any) to specify options for the control break which occurs at end-of-report. Three options alter the normal control break processing:
NEWPAGE

Causes a skip to the top of the next page after control break processing for the specified field is completed.

RENUM

Causes a skip to the top of the next page and resets the page number to 1 on the page following the control break.

NOPRINT

Suppresses printing the summary line for the specified control break. All other control break processing is performed as usual.

TITLE Statement

This optional statement defines the title lines to appear on your report. The TITLE statement syntax is illustrated below.

```
{field-name    }
{'literal-2'   }
TITLE   [literal-1]  {+literal-3    }
{-literal-3    }
{COL literal-4 }
```

Each title line is centered horizontally within the title area of the report. The first title line includes two additional items as follows:

- The current date is printed in the leftmost eight positions unless the NODATE option is specified on the REPORT statement.
- The word PAGE and the current page number are printed in the right-most 11 positions unless the NOPAGE option is specified on the REPORT statement.

[literal-1]

The value of literal-1 specifies the position of the title line within the title area in the case where you have more than one line. Literal-1 does not need to be specified for the first TITLE statement; if it is, its value must be 1. These numbers must be specified in ascending order with no duplicates.

At least one title item, specified by field-name or 'literal-2', must be coded on each TITLE statement.

field-name

Specifies that the contents of the named field appears on the title line. This name can be a field from any active file, a field from working storage, or a system-defined field.
'literal-2'

Specifies a character string for the title item. The character string must be enclosed in single quotes. For example, the TITLE line for the first report in the sample program is:

```
TITLE 1 'ANNUAL UPDATE REPORT - SALARIED EMPLOYEES'
```

You can specify more than one title item on the same line as long as the number of characters in the combined items, plus three characters between items, does not exceed the current LINESIZE value. Two options enable you to adjust the spacing between title items:

```
+literal-3 or -literal-3
```

Specifies the number of characters to be added to or subtracted from the normal three-character space between items. As long as you do not exceed the LINESIZE value, adding spaces enables you to spread out your title items; subtracting spaces enables you to squeeze them together. The numeric value of literal-3 must appear before the title item it pertains to: it affects only that item.

The TITLE statement:

```
TITLE 'PROJECTED INCOME FOR:' +5 REGION-NAME +5 BRANCH
```

produces:

```
PROJECTED INCOME FOR:        SOUTHEAST        TAMPA BAY
```

The whole title line is centered as usual, but additional space is left between the region and branch names to make the title more readable.

```
COL literal-4
```

Specifies the print column number where the first character of the next title item is printed. The value of literal-4 might not force the following title item(s) beyond the end of the value of the associated LINESIZE parameter. COL is permitted only with the NOADJUST option of the REPORT statement.

### HEADING Statement

This statement optionally defines an alternate column heading to be printed on the report in place of the specified field-name. Its syntax is diagrammed below.

```
HEADING  field-name ('literal' ... )
```

This statement enables you to specify another name to appear as a column heading on your report, rather than the field name specified in the library section of your program and on the LINE statement.
field-name

This parameter specifies the name of a field coded on the LINE statement. The value of the literal is the content of the new heading. For example, in the first report of the sample program, the column heading LAST-NAME appears as NAME through the statement:

```
HEADING LAST-NAME 'NAME'
```

Multiple literals within parentheses are stacked vertically over the column when it is printed. The statement:

```
HEADING LAST-NAME ('EMPLOYEE' 'NAME')
```

produces:

```
EMPLOYEE
NAME
```

The report declaratives illustrated later under the Special-name Report Procedures topic, which produce the report illustrated under the REPORT-INPUT topic, use the HEADING statement in this manner.

**LINE Statement**

This statement defines the contents of the lines of the report. The contents of the fields, whose names are specified in this statement, are printed across each line of the report page. The LINE statement syntax is diagrammed below.

```
{ field-name } { 'literal-2' } { +literal-3 } LINE [literal-1] { } ... { -literal-3 } { COL literal-4 } { POS literal-5 } ...
```

The LINE statement in the first report of the sample program is:

```
LINE DEPT LAST-NAME SERVICE RAISE SALARY
```

which directs CA-Easytrieve/Plus to:

- Extract the contents of each of the named fields each time a PRINT statement is issued
- Format these contents as per the other report declaratives
- Print these values across the report page from left to right in the order specified in the LINE statement.
[literal-1]

The value of literal-1 specifies the position of this LINE within the line group when you have multiple lines. An example of this is the specification for the mailing labels illustrated earlier. The first line contains names, the second line contains the street address, and the third line contains the city, state, and zip code, as follows:

```
LINE 1 FIRST-NAME -3 LAST-NAME
LINE 2 STREET
LINE 3 CITY -3 STATE ZIP
```

Literal-1 can be omitted in the first LINE statement. If it is specified, its value must be 1. Position numbers must be specified in ascending order with no duplicates.

At least one line item, specified by field-name or literal-2 must be specified on each LINE statement.

field-name

Specifies that the contents of the named field appear on the print line. This name can be a field from any active file or from working storage. For file and W fields, data is transferred to the print line as soon as the PRINT statement is executed. For S fields, data is transferred to the print line when the line is actually printed.

literal-2

Specifies a character string for the line item. The character string must be enclosed in single quotes.

You can specify more than one line item on the same line as long as the number of characters in the combined items, plus three characters between items, does not exceed the current LINESIZE value. Three options enable you to adjust the spacing between line items:

+literal-3 or -literal-3

Specifies the number of characters to be added to or subtracted from the normal three-character space between items. As long as you do not exceed the LINESIZE value, adding spaces enables you to spread out your line items. Subtracting spaces enables you to squeeze them together. The numeric value of literal-3 must appear immediately before the line item it pertains to; it affects only that item. The LINE statements:

```
LINE 1 FIRST-NAME -3 LAST-NAME
LINE 2 STREET
LINE 3 CITY -3 STATE ZIP
```

produce the names and addresses illustrated earlier, with the last name and the state moved three spaces to the left of where it would otherwise print. This provides more readable labels.
Report Procedures

COL literal-4

Specifies the column number where the first character of the next line item is printed. The value of literal-4 cannot force the following item(s) beyond the end of the value of the LINESIZE parameter. COL is permitted only with the NOADJUST option of the REPORT statement.

POS literal-5

Enables you to position items on lines 2 through 99 so they line up under specified items on line 1. The value of literal-5 corresponds to the item number on line 1 under which the item is to be placed. For example:

<table>
<thead>
<tr>
<th>LINE 1</th>
<th>REGION +</th>
<th>SSN +</th>
<th>NAME +</th>
<th>DATE-OF-BIRTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINE 2</td>
<td>POS 2</td>
<td>PHONE +</td>
<td>STREET +</td>
<td>DATE-OF-HIRE</td>
</tr>
<tr>
<td>POS 3</td>
<td>STREET +</td>
<td>DATE-OF-HIRE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS 4</td>
<td>DATE-OF-HIRE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINE 3</td>
<td>POS 3</td>
<td>CITY -3</td>
<td>STATE -2</td>
<td>ZIP</td>
</tr>
</tbody>
</table>

Line 1 consists of the region, social security number, name, and date of birth of each employee.

Line 2 lists the telephone number under the social security number, the street address under the name, and the date of hire under the date of birth.

Line 3 lists the city, state, and zip code under the name and street address.

The appearance of one line group is:

SOUTHWEST     571-40-8057     Florance M. Smith     11-26-32
785-4815        3250 Prospect Ave.     08-03-81
Riverside  CA 09265

Report Procedures

Although REPORT statements meet the vast majority of all report requirements, some reports depend upon special data manipulation. Report procedures are asynchronous routines which facilitate this requirement.

Code any report procedures at the end of their associated report. The report processor invokes special-name procedures (such as BEFORE-LINE or AFTER-BREAK), as required.
Coding Techniques

Coding report procedures is the same as coding procedures within JOB activities, with the following exceptions:

1. You cannot use the input/output generating statements listed below:
   
   DLI  
   GET  
   IDMS  
   POINT  
   PRINT  
   PUT  
   READ  
   WRITE  

2. You cannot use the STOP statement.

3. Use the DISPLAY statement to perform special report annotations. Use of DISPLAY requires the following extra considerations:
   
   You cannot code the DISPLAY statement's file-name-1 parameter. DISPLAY is only to the associated report.
   
   You cannot code the HEX option of DISPLAY.
   
   DISPLAY lines are counted and included in the end-of-page determination. However, the ENDPAGE procedure is not invoked by these lines.

In report procedures, you can reference any field contained in an active file or in working storage. When control or total fields are referenced, CA-Easytrieve/Plus automatically adjusts so that SUMFILE data is used. This assures access to the field actually used in the report.

LEVEL is a system-defined field provided for control reports. The field is defined as a two-byte binary field. The value in LEVEL indicates the control break level and varies from 0 to 'n + 1' where:

- LEVEL = 0 when processing detail lines
- LEVEL = n for total line processing at each control level
- LEVEL = n + 1 when processing FINAL totals.
Fields contained in S storage exhibit unique properties during report processing. S fields are stored in a static working storage area. Fields in this category are not copied onto report work files. All references to S fields occur at the time the report is actually formatted and printed. Remember, the format and print operation can occur at one of two different times. With this in mind, you should use S storage fields for:

- Temporary work fields for report procedures
- Line annotations controlled from report procedures
- Grand total values from which you can calculate percentages.

**Special-name Report Procedures**

Report procedures are invoked at specific points of the report processing activity. By analyzing these points, you can determine the specific use of the various procedures. The exhibit that follows illustrates the procedures listed below:

**REPORT-INPUT**

Final screening of report input data. Report data can be selected and/or modified.

**BEFORE-LINE**

Detail line has been created but not yet printed. Typical use is to annotate the body of the report before line printing. Detail line data cannot be modified.

**AFTER-LINE**

Detail line has been printed. Typical use is to annotate the body of the report after each line is printed.

**BEFORE-BREAK**

Modification of totals before total line printing. Typical use is to calculate averages on control reports.

**AFTER-BREAK**

Total line has been printed. Typical use is special annotation following total lines on control reports.

**ENDPAGE**

At end-of-page. This procedure can be used to produce footers on each page of the report.
### TERMINATION

At end-of-report. Produce end-of-report information, such as hash or other control totals.

```plaintext
(REPORT-INPUT)---(caused by the first PRINT statement)

<table>
<thead>
<tr>
<th>5/18/84</th>
<th>PROCEDURE USAGE</th>
<th>PAGE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTLN</td>
<td>CTL1</td>
<td>AMT1</td>
</tr>
</tbody>
</table>

**detail**

(BEFORE-LINE)

NA 1A 1

(AFTER-LINE)

(REPORT-INPUT)---(caused by the second PRINT statement)

(BEFORE-LINE)

NA 1A 2

(AFTER-LINE)

(REPORT-INPUT)---(caused by the third PRINT statement)

(BEFORE-BREAK)

NA 1A 3

(AFTER-BREAK)

**detail**

(BEFORE-LINE)

NA 1B 1

(AFTER-LINE)

(REPORT-INPUT)---(caused by the fourth PRINT statement)

(BEFORE-BREAK)

NA 1B 1

(AFTER-BREAK)

**total**

(BEFORE-BREAK)

NA 4

(AFTER-BREAK)

... ...

(ENDPAGE)

(REPORT-INPUT)---(caused by the fifth PRINT statement)

<table>
<thead>
<tr>
<th>5/18/84</th>
<th>PROCEDURE USAGE</th>
<th>PAGE 99</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTLN</td>
<td>CTL1</td>
<td>AMT1</td>
</tr>
</tbody>
</table>

**total**

(BEFORE-BREAK)

... ...

(AFTER-BREAK)

**total**

(BEFORE-BREAK)

xx yy ...

(AFTER-BREAK)

**total**

(BEFORE-BREAK)

xx ...

(AFTER-BREAK)

**total**

(BEFORE-BREAK)

... ...

(AFTER-BREAK)

(ENDPAGE)

(TERMINATION)
```
REPORT-INPUT

A REPORT-INPUT procedure selects and/or modifies report input data. This procedure is performed for each PRINT statement (report input). In order to cause the data to continue into report processing, you must execute a SELECT statement for the associated input data. In other words, input which does not get SELECTed is bypassed for continued processing.

When the report data has been spooled (because the report had been SEQUENCEd or the printer file had been in use), the REPORT-INPUT procedure is invoked as each spooled record is read to produce this report.

Although you can code the logic within the JOB activity itself, it is occasionally desirable to place the logic in a REPORT-INPUT procedure. The next exhibit illustrates use of the REPORT-INPUT procedure in final report input selection. The first 10 report inputs for each code are the only ones selected for ultimate report input:

```plaintext
DEFINE COUNT S 2 P 0
DEFINE HOLD-CODE S CODE...
REPORT-INPUT. PROC
  IF CODE NE HOLD-CODE
    HOLD-CODE = CODE
    COUNT = 0
  END-IF
  IF COUNT LT 10
    COUNT = COUNT + 1
    SELECT
  END-IF
END-PROC
```

BEFORE-LINE and AFTER-LINE

A BEFORE-LINE procedure is invoked immediately before, and an AFTER-LINE procedure immediately following, the printing of each detail line. These procedures are typically used for special annotation associated with these detail lines. The next exhibit illustrates how either procedure can cause detail lines to be printed in groups of five with one blank line separating each group:

```plaintext
DEFINE COUNT S 2 P 0...
AFTER-LINE. PROC (could be BEFORE-LINE)
  IF COUNT EQ 4
    DISPLAY ' '
    COUNT = 0
  ELSE
    COUNT = COUNT + 1
  END-IF
END-PROC
```
BEFORE-BREAK

This procedure can be used in control reports to modify totals before they are printed. A typical application is to calculate averages and/or percentages for the fields totaled.

The sample program can be modified to include a BEFORE-BREAK procedure which calculates the percentage of senior employees (15 or more years of service) in each department and the average length of service for all employees by department, as illustrated below.

```
1  PARM  DEBUG(FLOW FLDCHK)
2  *
3  FILE PERSNL FB(150 1800)
4     NAME                     17  16   A
5     LAST-NAME               NAME  8   A
6   PAY-GROSS                94  4   P  2
7   DEPT                     98  3   N
8   DATE-OF-HIRE             136  6   N
9   HIRE-MM DATE-OF-HIRE     2   N
10  HIRE-DD DATE-OF-HIRE +2   2   N
11  HIRE-YY DATE-OF-HIRE +4   2   N
12  SALARY                   W  4   P  2
13   SERVICE                  W  2   P  0
14  CURR-DATE                S  6   N
15  CURR-MM CURR-DATE        2   N
16  CURR-DD CURR-DATE +2     2   N
17  CURR-YY CURR-DATE +4     2   N
18  SENIORS                   W  2   P  0
19  *
20  JOB  INPUT PERSNL
21    %GETDATE CURR-DATE
22    PERFORM SERVICE-CALC
23    IF  SERVICE  GT  14
24       SENIORS  =  1
25     ELSE
26       SENIORS  =  0
27     END-IF
28    PRINT  SENR-RPT
29    *
30    SERVICE-CALC.  PROC
31    SERVICE  =  CURR-YY - HIRE-YY
32    IF  CURR-MM  <  HIRE-MM
33       SERVICE  =  SERVICE  -  1
34     END-IF
35    IF  CURR-MM  NE  HIRE-MM
36       GOTO  QUIT-SERV-CALC
37    END-IF
38    QUIT-SERV-CALC
39    END-PROC
40    *
41  REPORT  SENR-RPT  LINESIZE  62  SUMMARY  SPREAD  NODATE  NOPAGE
42    SEQUENCE  DEPT
43    CONTROL  DEPT
44    TITLE  1  'SERVICE UPDATE REPORT - SALARIED EMPLOYEES'
45    HEADING  TALLY ('NUMBER OF' 'EMPLOYEES' 'IN DEPT')
46    HEADING  SERVICE ('AVERAGE' 'SERVICE')
47    HEADING  SENIORS ('PERCENT' 'SENIORS')
48    HEADING  SALARY ('TOTAL' 'SALARY')
```
**Report Procedures**

71  LINE DEPT TALLY SENIORS SERVICE SALARY
72  *
73  BEFORE-BREAK. PROC
75  SENIORS = SENIORS * 100 / TALLY + .5
76  SERVICE = SERVICE / TALLY
77  END-PROC
78  *

In this program, a system-defined field named TALLY is referenced in the report declaratives and in the BEFORE-BREAK procedure. TALLY contains the number of detail records that compose a control break.

**AFTER-BREAK**

An AFTER-BREAK procedure can be used to produce a special annotation on control reports. You can use the value of LEVEL (a system-defined field) to determine which control break is being processed. In the next exhibit, the total line for the second control field CTL1 receives special annotation:

```plaintext
... REPORT ...
  CONTROL CTLN CTL1
...
AFTER-BREAK. PROC
  IF LEVEL EQ 1
    DISPLAY 'TOTALS FOR DEPARTMENT' CTL1
  END-IF
END-PROC
```

**ENDPAGE**

You can use an ENDPAGE procedure to produce page footing information. It is invoked whenever end-of-page is detected. It is typically used to produce page totals or other annotations, as in the following example of page footer annotation:

```plaintext
... ENDPAGE. PROC
  DISPLAY PAGE-AMT ' IS THE PAGE TOTAL'
  DISPLAY SKIP 2 'CONFIDENTIAL - FOR EYES ONLY'
END-PROC
```

**TERMINATION**

A TERMINATION procedure is invoked at the end of the report. You can use this procedure to print report footing information, including control totals and distribution information. The next exhibit is an example of report footing:

```plaintext
... TERMINATION. PROC
  DISPLAY NEWPAGE
  DISPLAY GRAND-TOTAL ' IS THE CONTROL TOTAL'
  DISPLAY SKIP 5 'ROUTE TO: ...'
...
END-PROC
```
Chapter 9

File Processing

Introduction

Data file creation can be a very complex process. It is not within the scope of this Application Guide to provide sufficient information to enable you to create data files from scratch. Rather, this guide enables you to use CA-Easytrieve/Plus to process any existing file to read it, change records within it, add new records, or delete existing records.

If you want to create new files, you need to enlist the help of your data center.

CA-Easytrieve/Plus can process files or databases from the simplest to the most complex. File types include sequential access method (SAM), indexed sequential access method (ISAM), virtual storage access method (VSAM), virtual file manager (VFM) files, IMS/DLI, and IDMS databases. You can let CA-Easytrieve/Plus do all your file processing automatically, or you can control some or all of the operations yourself.

Control of Input/Output (I/O)

As described in Chapter 7, “Input/Output Specification,” the easiest way to control I/O is to let the system do it.

- Automatic I/O (under system control) includes the files specified for input on the JOB and SORT statements, and the files specified for output on the SORT, PRINT, and DISPLAY statements.
- Controlled I/O (under your control) includes the GET, POINT, and READ statements for input, and the PUT and WRITE statements for output.
You can use I/O control statements within a JOB activity, with or without automatic I/O, by observing the following restrictions:

- No I/O control statements are valid in REPORT procedures.
- No I/O control statements are valid for files involved in automatic input processing, except:
  - The POINT statement can be used with automatic input for VSAM and ISAM files to enable skip-sequential input processing
  - The PUT and WRITE statements can be used to update an automatic input VSAM file.

**Record Formats**

Records in your file must be in one of the following formats:

- Fixed-length
- Variable-length
- Undefined-length.

All formats must adhere to established IBM processing standards. Check with your data center if you have questions about these format standards.

CA-Easytrieve/Plus makes these assumptions about the record formats of CARD, PUNCH, and VSAM files:

- CARD and PUNCH file records are fixed-length, 80 characters long.
- VSAM file records are undefined-length.

The record lengths of variable and undefined records being output are controlled by the current contents of the RECORD-LENGTH field for that file. If the current record (the last record you either input or output) is smaller than the record you want to output, you can increase the record length by an Assignment statement which precedes the output statement. For example:

```
SALUPD: RECORD-LENGTH = 200
PUT SALUPD
...```

System-Defined Fields

CA-Easytrieve/Plus provides three special data fields for each file:

RECORD-LENGTH

Contains one of the following:

- For fixed-length records, the value specified for record length on the FILE statement.
- For variable or undefined-length records, the length of the data in the current record (does not include the space for the record-descriptor-word, it is automatically maintained by the system).

RECORD-COUNT

Contains the number of logical I/O operations performed for the file.

FILE-STATUS

Contains a code which indicates the result of the most recent I/O operation.

Error Conditions

Error conditions during file processing usually fall into one of three categories:

- File OPEN errors, usually caused by incorrect or missing JCL information. The operating system terminates processing. This type of problem should be referred to your data center.
- Invalid file reference errors, caused by statements that refer to data from a file with no currently available record (for example, after end-of-file). A CA-Easytrieve/Plus diagnostic message is issued and processing terminates.
- Improper handling of nonzero STATUS conditions returned from I/O statements. You are responsible for handing these types of errors.

Data Availability Tests

You can use several conditional expressions to test for the availability of data for file processing. These are discussed in Chapter 7, “Input/Output Specification,” and later in this chapter.

Opening and Closing Files

CA-Easytrieve/Plus automatically OPENS and CLOSEs all files.
SAM Files

CA-Easytrieve/Plus processes Sequential Access Method (SAM) files according to the following rules:

1. You cannot process the same SAM file as both an input and an output file within the same JOB activity. This is allowable for SORT activities.
2. You can create SAM files in one activity and process them in subsequent activities.
3. Only one CARD file is permitted in a CA-Easytrieve/Plus program.

Input

CA-Easytrieve/Plus permits both automatic and controlled I/O for SAM files. The sample program uses automatic I/O exclusively. The next two exhibits illustrate how to process a SAM file using each facility:

Automatic SAM Processing
*FILE PERSNL FB(150 1800)
...*
*JOB INPUT PERSNL
...

Controlled SAM Processing
FILE PAYFILE
  REC-KEY  1  3  N
  *
  JOB INPUT NULL
  GET PAYFILE
  DO WHILE (REC-KEY < 600, AND NOT EOF PAYFILE)
    PRINT PAY-RPT
    GET PAYFILE
  END-DO
  STOP
  *
  REPORT PAY-RPT ...
  ...

You can process only one of your input files as CARD input. CARD input is placed into the system input stream (SYSIN for OS/390, SYSIPT for VSE). If your operating mode is the default (syntax check, compile, and execute), your file data must follow an END statement after your program, as illustrated under the Device-type Parameters topic in Chapter 3, “Library.”
Output

You can load output files with the PUT statement, as described in Chapter 7, “Input/Output Specification.” The next exhibit illustrates this operation.

```
*  
FILE PAYFILE F(150)  
  REC-KEY 1 3 N  
  SALARY 94 4 P 2  
*  
FILE SALUPD VS CREATE  
*  
JOB INPUT NULL  
  GET PAYFILE  
  DO WHILE (REC-KEY < 600, AND NOT EOF PAYFILE)  
    SALARY = SALARY * 1.1  
    PUT SALUPD FROM PAYFILE  
    PRINT UPD-RPT  
  END-DO  
  STOP  
*  
REPORT UPD-RPT ...  
...  
```

You can specify the PUNCH attribute on the FILE statement when the Card Punch is the output device for a SAM file produced under VSE, as illustrated in the next exhibit. For OS/390, JCL defines the PUNCH output.

```
FILE CARDOUT PUNCH  
  COUNTER 12 4 N  
*  
JOB INPUT NULL  
  ...  
  COUNTER = COUNTER + 1  
  PUT CARDOUT  
...  
```

VFM Files

Virtual File Manager (VFM) is a sequential access method used for all CA-Easytrieve/Plus work file requirements. You can also use VFM files for temporary sequential processing. VFM processing is identical to SAM processing. The next exhibit illustrates a typical use of VFM:

```
*  
FILE PERSNL FB(150 1800)  
  EMP# 9 5 N  
FILE SORTPER F 150 VIRTUAL  
  UPD-EMP# 9 5 N  
*  
SORT PERSNL TO SORTPER USING EMP#  
*  
JOB INPUT SORTPER  
*  
```
SORTPER is a virtual file. You do not have to define it in the JCL since it is actually stored and retrieved by VFM from storage.

**ISAM Files**

CA-Easytrieve/Plus processes Indexed Sequential Access Method (ISAM) files as input only. You can perform sequential, skip-sequential, or random processing on these files.

**Sequential Processing**

Sequential processing can be performed under automatic or controlled I/O. The next exhibit illustrates automatic sequential file processing.

```plaintext
FILE PAYFILE IS
   SALARY 94 4 P 2
   *
JOB INPUT PAYFILE
   SALARY = SALARY * 1.1
   PRINT UPD-RPT
   *
```

**Skip-Sequential Processing**

Skip-sequential processing enables you to point to a record, then continue processing from that location. The next exhibit illustrates skip-sequential processing.

```plaintext
FILE PAYFILE IS
   REC-KEY 1 3 N
   SALARY 94 4 P 2
   *
JOB INPUT PAYFILE
   IF REC-KEY EQ 299 THRU 499
      PERFORM POINTER
      GO TO JOB
   END-IF
   SALARY = SALARY * 1.1
   PRINT UPD-RPT
   *
   POINTER. PROC
      POINT PAYFILE GE 500 STATUS
      IF EOF PAYFILE, OR PAYFILE:FILE-STATUS NOT ZERO
         STOP
      END-IF
      END-PROC
   *
```
Random Processing

Random processing enables you to choose specific records within a file for processing, regardless of their location in the file. Random processing is always performed with controlled I/O. The next exhibit illustrates random processing.

```
FILE PAYFILE IS
   EMPL# W 4 N
   NAME 5 20 A
*
FILE NEWFILE VS CREATE
*
JOB INPUT NULL
   EMPL# = 1126
   READ PAYFILE, KEY EMPL#, STATUS
   IF PAYFILE:FILE-STATUS NOT ZERO
   DISPLAY 'RECORD NOT FOUND'
   STOP
   END-IF
   IF NAME EQ 'OLDNAME'
   NAME = 'NEWNAME'
   PUT NEWFILE
   ELSE
   DISPLAY 'NAME DOES NOT MATCH'
   END-IF
   STOP
*
```

VSAM Files

CA-Easytrieve/Plus processes Virtual Storage Access Method (VSAM) files as both input and output files. You can perform the same types of processing (sequential, skip-sequential, and random processing) on VSAM files as on ISAM files. VSAM files are organized as one of the following types:

<table>
<thead>
<tr>
<th>VSAM Files</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESDS</td>
<td>Entry-sequenced data set</td>
</tr>
<tr>
<td>KSDS</td>
<td>Key-sequenced data set</td>
</tr>
<tr>
<td>RRDS</td>
<td>Relative-record data set</td>
</tr>
</tbody>
</table>

You must identify your VSAM file organization before coding your program.
**File Loading**

You can enter (load) data for the first time into a new VSAM file with the PUT statement in the JOB activity portion of your program, as illustrated in the next exhibit.

```plaintext
FILE PAYMSTR VS UPDATE
   REC-KEY  1  3  N
   SALARY  94  4  P  2
FILE SALUPD VS CREATE
* 
JOB INPUT NULL
   POINT PAYMSTR GE '300'
   GET PAYMSTR
   DO WHILE (REC-KEY < 500, AND NOT EOF PAYMSTR)
      SALARY = SALARY * 1.1
      WRITE PAYMSTR
      PUT SALUPD FROM PAYMSTR
      GET PAYMSTR
   END-DO
   STOP
* 
```

This routine updates the PAYMSTR records between 300 and 499 with a 10 percent salary increase and also loads the updated records into the newly created file SALUPD.

**Input**

CA-Easytrieve/Plus processes VSAM input files the same as ISAM files. You can perform sequential, skip-sequential, or random processing on VSAM input files.

- Sequential processing can be performed under automatic or controlled I/O. The next exhibit illustrates sequential processing under automatic control.

```plaintext
FILE PAYMSTR VS
   SALARY  94  4  P  2
* 
JOB INPUT PAYMSTR
   SALARY = SALARY * 1.1
   PRINT SAL-RPT
* 
```
Skip-sequential processing enables you to specify one or more records in the input file which are not processed. They are skipped and processing continues with the following records. The next exhibit illustrates skip-sequential processing.

FILE PAYMSTR VS
  REC-KEY  1  3  N
  SALARY-CODE  134  2  N
*
FILE NEWMSTR VS CREATE
*
JOB INPUT PAYMSTR
  IF REC-KEY EQ 100 THRU 199
    PERFORM SKIPPER
  GO TO JOB
END-IF
SALARY-CODE = SALARY-CODE + 5
PUT NEWMSTR FROM PAYMSTR
*
SKIPPER.  PROC
  POINT PAYMSTR GE '300' STATUS
  IF EOF PAYMSTR, OR PAYMSTR:FILE-STATUS NOT ZERO
    STOP
  END-IF
END-PROC
*

Random processing enables you to choose specific records within a file for processing, regardless of their location in the file. Random processing is always performed with controlled I/O. The next exhibit illustrates random processing.

FILE PAYMSTR VS UPDATE
  DEPT    W  3  N
  JOB-CATEGORY 132  2  N
*
JOB INPUT NULL
  DEPT = 914
  READ PAYMSTR, KEY DEPT, STATUS
  IF PAYMSTR:FILE-STATUS NOT ZERO
    STOP
  END-IF
  IF JOB-CATEGORY GT 25
    JOB-CATEGORY = 77
  ELSE
    DISPLAY DEPT +3 JOB-CATEGORY +3 'NOT COVERED'
  END-IF
  STOP
*
**Record Addition**

You can use the WRITE or PUT statement to add records to an established VSAM file. Either statement adds a single record to the file, but the PUT statement is more efficient if you are inserting many records into the same place in the file. To add records to a file, you must code the UPDATE parameter on the FILE statement as illustrated in the next two exhibits.

**VSAM Single Record Addition**

```
FILE PAYMSTR VS UPDATE
*
FILE NEWBODS VS
EMPL#   1   4   N
    NAME    5  20   A
*
JOB INPUT NULL
GET NEWBODS
WRITE PAYMSTR ADD FROM NEWBODS STATUS
IF PAYMSTR:FILE-STATUS EQ 8
    DISPLAY EMPL# +3 NAME +3 'DUPLICATE RECORD'
END-IF
STOP
*
```

**VSAM Mass-Sequential Record Addition**

```
FILE PAYMSTR VS UPDATE
*
FILE NEWBODS VS
*
JOB INPUT NEWBODS
PUT PAYMSTR FROM NEWBODS STATUS
IF PAYMSTR:FILE-STATUS NOT ZERO
    DISPLAY 'FILE ERROR - ' PAYMSTR:FILE-STATUS
END-IF
*
```

**Record Deletion**

You can delete individual records from a VSAM file with the WRITE statement using the DELETE parameter as illustrated in the next exhibit. The deleted record is the specified file's current input record.

```
FILE PAYMSTR VS UPDATE
EMPL#    1   5   N
*
JOB INPUT PAYMSTR
    IF EMPL# EQ 44152 THRU 44449
        WRITE PAYMSTR DELETE
    END-IF
    IF EMPL# GE 44450
        STOP
    END-IF
*
```
Record Update

You can modify and update the current record of a VSAM input file using the WRITE statement as illustrated in the next exhibit.

```plaintext
FILE PAYMSTR VS UPDATE
    EMPL# W 5 N
    NAME   6 20 A
*
JOB INPUT NULL
EMPL# = 41452
READ PAYMSTR, KEY EMPL#, STATUS
IF PAYMSTR/File-Status NOT ZERO
    DISPLAY 'NO PAYMSTR RECORD EXISTS FOR ' EMPL#
    STOP
END-IF
IF NAME EQ 'AMAN'
    NAME EQ 'NICHOLSON'
    WRITE PAYMSTR UPDATE
ELSE
    DISPLAY 'EMPLOYEE NUMBER 41452 IS ' NAME
END-IF
STOP
*
```

Synchronized File Processing

CA-Easytrieve/Plus simplifies combining the data from more than one file. It has the capacity to synchronize any number of files which can be processed sequentially. Synchronizing more than two files necessitates a high level of data processing expertise and a comprehensive understanding of file structures. If your application requires complex synchronized file processing, refer to the CA-Easytrieve/Plus Reference Guide for detailed information.
This chapter of the Application Guide describes a match/merge operation using two input files with one key each. The code for a sample program to accomplish this task is illustrated in the next exhibit. Subsequent paragraphs present detailed discussions of the rules to be followed in specifying input to synchronized file processing and techniques for determining file relationships, using the code in the next exhibit, as illustrated.

```
1 PARM DEBUG(FLOW FLDCHK)
2 *
3 FILE PERSNL FB(150 1800)
4   OLD-EMP# 9 5 N
5 *
6 FILE PERSUPD CARD
7   EMP# 1 5 N
8   RAISE-PERCENT 7 2 N
9 *
10 FILE SORTPER F 150 VIRTUAL
11   UPD-EMP# 9 5 N
12   NAME 17 8 A
13   PAY-GROSS 94 4 P 2
14   NEWSAL W 4 P 2
15 *
16 FILE NEWPERS FB(150 1800)
17 *
18 FILE ERRPRINT PRINTER
19 *
20 *
21 SORT PERSNL TO SORTPER USING OLD-EMP#  
22 *
23 JOB INPUT (SORTPER KEY(UPD-EMP#) +  
   PERSUPD KEY(EMP#) )  
24 *
25 IF MATCHED  
26   NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)  
27   PRINT NEW-RPT  
28   PAY-GROSS = NEWSAL  
29 END-IF  
30 IF SORTPER  
31   PUT NEWPERS FROM SORTPER  
32 ELSE  
33   DISPLAY ERRPRINT EMP# 'RECORD NOT MATCHED'  
34 END-IF  
35 *
36 REPORT NEW-RPT LINESIZE 80 NOPAGE NODATE 
37   SEQUENCE NAME  
38 TITLE 1 'SALARY UPDATE REPORT'  
39 TITLE 2 'EMPLOYEES WITH OVER 25 YEARS SERVICE'  
40 HEADING UPD-EMP# ('EMPL' 'NUMBER')  
41 HEADING NAME ('EMPL' 'NAME')  
42 HEADING PAY-GROSS ('OLD' 'SALARY')  
43 HEADING NEWSAL ('NEW' 'SALARY')  
44 HEADING RAISE-PERCENT ('RAISE' '%')  
45 LINE UPD-EMP# NAME PAY-GROSS NEWSAL RAISE-PERCENT  
46 *
47 END
```

01730 08
04225 09
09481 09
11473 11
11710 10
12267 12
The sample program illustrated in the above exhibit sorts the Personnel Master File PERSNL into order by employee number, then matches the sorted output file (SORTPER), against a card file (PERSUPD) containing raise calculations for specified employees. The data for the CARD input file, also in order by employee number, is coded immediately following the last CA-Easytrieve/Plus statement (END) in the program.

A new master file (NEWPERS) is created which contains the updated salary information, and a report is printed to list the names and associated data about the employees who received raises. The report is illustrated in the next exhibit.

SALARY UPDATE REPORT
EMPLOYEES WITH OVER 25 YEARS SERVICE

<table>
<thead>
<tr>
<th>EMPL NUMBER</th>
<th>EMPL NAME</th>
<th>OLD SALARY</th>
<th>NEW SALARY</th>
<th>RAISE %</th>
</tr>
</thead>
<tbody>
<tr>
<td>11473</td>
<td>BERG</td>
<td>759.20</td>
<td>842.71</td>
<td>11</td>
</tr>
<tr>
<td>04225</td>
<td>LOYAL</td>
<td>295.20</td>
<td>321.76</td>
<td>09</td>
</tr>
<tr>
<td>09481</td>
<td>OSMON</td>
<td>628.00</td>
<td>684.52</td>
<td>09</td>
</tr>
<tr>
<td>11710</td>
<td>POWELL</td>
<td>243.20</td>
<td>267.52</td>
<td>10</td>
</tr>
<tr>
<td>01730</td>
<td>SMOTH</td>
<td>315.20</td>
<td>340.41</td>
<td>08</td>
</tr>
<tr>
<td>12267</td>
<td>WIMN</td>
<td>373.60</td>
<td>418.43</td>
<td>12</td>
</tr>
</tbody>
</table>

Synchronized files are subject to the following rules:

- Both files must be sorted in ascending order by their keys. For example, in the sample synchronized file processing program illustrated earlier, the original input file PERSNL is in order by Region. For this application it is sorted to output file SORTPER, using the employee number as the key:
  
  ```
  SORT PERSNL TO SORTPER USING OLD-EMP#
  ```

- The same number of keys must be specified for each file.

- The corresponding keys for both files must have the same data class. That is, corresponding keys must both be alphabetic or both be numeric. The keys can have different lengths. Numeric keys can have different data types (N, P, U, B).
Synchronized File Processing

Input

Files for synchronized processing are specified in the library the same as any other sequential file; the only difference is how the file is specified on the JOB statement. The next exhibit illustrates the I/O specification for the sample program.

```plaintext
2 *
3 FILE PERSNL FB(150 1800)
4      OLD-EMP#  9 5  N
5 *
6 FILE PERSUPD CARD
7      EMP#  1 5  N
8      RAISE-PERCENT  7 2  N
9 *
10 FILE SORTPER F 150  VIRTUAL
11      UPD-EMP#  9 5  N
12      NAME  17 8  A
13      PAY-GROSS  94 4  P  2
14      NEWSAL  W 4  P  2
15 *
16 FILE NEWPERS FB(150 1800)
17 *
18 FILE ERRPRINT PRINTER
20 *
21 SORT PERSNL TO SORTPER USING OLD-EMP#
22 *
23 JOB INPUT (SORTPER KEY(UPD-EMP#) +
              PERSUPD KEY(EMP#) )
24 *
```

Conditional Expressions

In synchronized file processing, you need to know:

- If records are available from each file
- If a record has the same key as a record in another file
- If two or more records in a file have the same key.

This can be determined by using three types of conditional expressions: file presence, file presence series, and record relational.
File Presence Condition

This condition determines if a record of the named input file is available for processing. The format is diagrammed in the next exhibit.

IF [NOT] [EOF] file-name

The condition tests true if a record is available, such as:
- The optional EOF parameter returns a true result if the named file is at end-of-file.
- The optional NOT parameter reverses the condition test; the result is true if no record is currently available for processing or if the named file is NOT at end-of-file.

The next exhibit illustrates how this is used in the sample program.

IF SORTPER
  PUT NEWPERS FROM SORTPER
ELSE
  DISPLAY ERRPRINT EMP# ' RECORD NOT FOUND'
END-IF

File Presence Series Condition

This condition determines if the records from more than one file have the same key. The format is diagrammed in the next exhibit.

IF [NOT] MATCHED

A record from one file is considered to be available for processing if its key matches the key of a record from the other file. The result is true if the input files have matching keys. The optional NOT parameter reverses the condition test; the result is true if the keys do not match.

The next exhibit illustrates how this is used in the sample program.

IF MATCHED
  NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)
  PRINT NEW-RPT
  PAY-GROSS = NEWSAL
END-IF

When this condition (IF MATCHED) is true, a record is available from both PERSUPD and SORTPER.
Record Relational Condition

This condition tests for duplicate records within one file. The current record of the named file is compared to the previous and next records of the same file. The optional NOT parameter reverses the condition tests. The next exhibit diagrams this condition.

```
IF [NOT] DUPLICATE file-name

```

Depending on the condition parameter chosen, the tests are performed as follows:

**DUPLICATE**

The current record of the named file is compared to the previous and next records of the same file. The result is true if the current record has the same key as either of the other two records. The optional NOT parameter returns a true result if neither of the contiguous records has the same key.

**FIRST-DUP**

The current record of the named file is compared to the previous and next records of the same file. The result is true if the current record's key is different from the previous record's key but the same as the next record's key.

**LAST-DUP**

The current record of the named file is compared to the previous and next records of the same file. The result is true if the current record's key is the same as the previous record's key but different from the next record's key.

The next exhibit illustrates how these condition tests work.

```
+---+---+---+
|   | E  |   |
| A | E  | C |
+---+---+---+

First-DUP
```

```
+---+---+---+
|   | E  |   |
| D  | E  |   |
+---+---+---+

DUP

```

```
+---+---+---+
|   | E  |   |
| D  | E  |   |
+---+---+---+

DUP

```

```
+---+---+---+
|   | E  |   |
| D  | E  | C |
+---+---+---+

Last-DUP
```
Table Definition

A table is a collection of uniform data records. Tables have two parts:

- The argument uniquely identifies a table entry.
- The description is information directly associated with the argument.

Typical examples of table usage include: organization structures, accounting charts-of-accounts, state abbreviations, department code/names, and parts lists for assembly processes.

Tables are defined by FILE statements in the library section of your program. The TABLE option must be coded on the FILE statement, as discussed in Chapter 3, “Library.” This option identifies the file as the target of a SEARCH statement issued in your program.

```plaintext
FILE  file-name TABLE [        
[INSTREAM]
FILE  file-name TABLE [        
[literal ]
```

The only fields that can be defined for TABLE files are ARG (argument) and DESC (description). ARG defines the field used to search the table. DESC defines the field which contains the desired information. Data within a TABLE must be sorted in ascending order by its search argument. The maximum length for an alphanumeric ARG or DESC field is 254 bytes.

There are two types of TABLEs, instream and external. Instream tables reside within your program — they are established for use when your program is compiled. If you make changes to data in an instream table, you must recompile your program. External tables are stored on files external to your program — they are established for use during initiation of the JOB activity that contains the SEARCH statement which references them.
Instream Tables

Instream tables are specified by the INSTREAM subparameter of the TABLE option on the FILE statement. Instream tables are created by coding the table data immediately following the associated library definition statements for the table file. Table data is ended by the word ENDTABLE in the first eight positions of a record. Instream data is 80 characters per record. Table size is limited only by the amount of available memory. The next exhibit illustrates an instream table definition.

```
FILE WEEKDAY TABLE INSTREAM
   ARG    1   1   A
   DESC   3   9   A
1 SUNDAY
2 MONDAY
3 TUESDAY
4 WEDNESDAY
5 THURSDAY
6 FRIDAY
7 SATURDAY
ENDTABLE
```

External Tables

If you specify the TABLE option with no subparameter, the file is an external table whose maximum number of entries is limited by a value in the options table established at installation. Check with your data center to determine this value. If the number of entries in your external table is larger than the default value, you can code a numeric literal as the subparameter of the TABLE option to specify the maximum number of entries.

A file which meets the following criteria can be defined as an external table:

- An existing file that is in ascending order by the field used as a search argument
- A file created by having its name specified as the TO parameter of a SORT statement which is sorted into ascending order by the search argument.
SEARCH Statement

Use the SEARCH statement to access table information. Its syntax is illustrated next.

```
SEARCH file-name  WITH field-name-1    GIVING field-name-2
```

**file-name**

This is the name of the file which describes the table and its source. The file must be defined with the TABLE attribute.

**WITH field-name-1**

This parameter identifies the field that contains the search argument. Field-name-1 can be defined in working storage or in any file except a file with the TABLE attribute.

**GIVING field-name-2**

This parameter identifies the receiving field for the results of the table search. This field can be defined in working storage or in any file except a file with the TABLE attribute.

The named TABLE file is searched for an ARGument whose value is the same as the value of field-name-1. If a match is found, the content of field-name-2 is set to the value of the DESCRIPTION associated with the ARGument. The content of field-name-2 is not changed if a match for field-name-1 is not found in the named TABLE file. An IF statement with a file presence condition (see Chapter 9, “File Processing”) can be coded after the SEARCH statement to determine the success of the table search.
You can code SEARCH statements any place within a JOB activity, SORT procedure, or REPORT procedure. The next exhibit illustrates the retrieval of names of the days of the week based on numeric identification codes.

* FILE CALENDR
  DAY-OF-WEEK 12 1 A
  NAME-OF-DAY 14 20 A
  * FILE WEEKDAY TABLE INSTREAM
    ARG 1 1 A
    DESC 3 9 A
    1 SUNDAY
    2 MONDAY
    3 TUESDAY
    4 WEDNESDAY
    5 THURSDAY
    6 FRIDAY
    7 SATURDAY
    ENDTABLE
  *
  JOB INPUT CALENDR
  SEARCH WEEKDAY WITH DAY-OF-WEEK, GIVING NAME-OF-DAY
  IF WEEKDAY
    DISPLAY NAME-OF-DAY, ' IS DAY ', DAY-OF-WEEK
  ELSE
    DISPLAY '****INVALID DAY OF WEEK = ', DAY-OF-WEEK
  END-IF
  *

The next exhibit is a more extensive example that illustrates the retrieval of month name translations, based on the English name.

* FILE CALENDR
  ENGL-NAME 12 10 A
  EURO-NAME 22 40 A
  FREN-NAME 22 10 A
  ITAL-NAME 32 10 A
  GERM-NAME 42 10 A
  SPAN-NAME 52 10 A
  *
  FILE MONTH TABLE
  ARG 1 10 A
  DESC 11 40 A
  *
  JOB INPUT CALENDR
  SEARCH MONTH WITH ENGL-NAME, GIVING EURO-NAME
  IF NOT MONTH
    DISPLAY 'INVALID ENGLISH NAME = ', ENGL-NAME
    GO TO JOB
  END-IF
  PRINT MON-RPT
  *
  REPORT MON-RPT LINESIZE 80
  SEQUENCE ENGL-NAME
  TITLE 1 'WESTERN EUROPEAN MONTH NAME'
  TITLE 2 'TRANSLATION TABLE'
  HEADING ENGL-NAME ('ENGLISH' 'NAME')
  HEADING FREN-NAME ('FRENCH' 'NAME')
  HEADING ITAL-NAME ('ITALIAN' 'NAME')
  HEADING GERM-NAME ('GERMAN' 'NAME')
  HEADING SPAN-NAME ('SPANISH' 'NAME')
  LINE ENGL-NAME FREN-NAME ITAL-NAME GERM-NAME SPAN-NAME
  *
Introduction

Through the IMS/DL/I interface, CA-Easytrieve/Plus provides facilities for information retrieval from databases. To use this interface efficiently, you should have a thorough knowledge of IMS/DL/I and of the database(s) to be processed. Refer to the CA-Easytrieve/Plus Reference Guide for detailed discussions of the processing techniques needed and to your Database Administrator for specific information regarding the structure of your database.

A database is a collection of interrelated data items. The specific pieces of data, called segments, are organized in a hierarchical or tree structure. A segment is the smallest unit of data that an application program can retrieve from the database. The highest level segment is called the root segment. The root can have one or more dependent segments, which in turn can also have dependent segments. The segments immediately above and below a given segment are called parent and child segments, respectively.

Hierarchical Database Structure

```
    A
   / |
  B  C
 /   |
D    E
```
This chapter briefly describes the CA-Easytrieve/Plus statements which define database processing:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE</td>
<td>Identifies the database.</td>
</tr>
<tr>
<td>RECORD</td>
<td>Identifies the database segments available for processing.</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>Describes automatic database input.</td>
</tr>
</tbody>
</table>

Three special terms used in database processing are referenced throughout this chapter:

<table>
<thead>
<tr>
<th>Terms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Description (DBD)</td>
<td>A control block that describes the structure of the database. The DBD also defines the appearance and contents (fields or records) that make up each of the segment types in the database.</td>
</tr>
<tr>
<td>Program Communication Block (PCB)</td>
<td>Defines an application program's view of the database. An application program often needs to process only some of the segments in a database. A PCB defines which of the segments in the database the program is allowed to access.</td>
</tr>
<tr>
<td>Program Specification Block (PSB)</td>
<td>Contains the PCBs for a particular application program. A program can use one or several PCBs. There is one PSB for each application program.</td>
</tr>
</tbody>
</table>

**FILE Statement**

FILE file-name DLI (dbd-name  [literal])

The FILE statement (see Chapter 3, “Library”) identifies the database by specifying DLI as the file-type parameter and by identifying the PCB to be processed as follows:

- dbd-name is the name of the DBD.
- literal is a numeric integer that specifies the relative occurrence of the DBD within the PSB to be processed.

All field definitions coded immediately after the FILE statement relate to the PCB. The PCB data format is described in the IBM publication, *IMS/DLI Applications Programming Manual*. PCB references are normally made in association with controlled database activities (which are not covered in this Application Guide.)
RECORD Statement

RECORD statements are coded immediately after the FILE statement to identify the database segments which are to be available for processing. RECORD allocates a work space which contains the segment data during execution. Field-definition statements, coded immediately following a RECORD statement, relate to data fields within that segment. A RECORD statement must be coded for each segment of the database to be processed. The RECORD statements must be coded in the same order as in the PSB which defines the database. All segments of a database do not need to be defined. However, since incomplete paths are not supported, the parent segment of each RECORD must be coded.

The next exhibit illustrates the RECORD statement syntax.

```
RECORD  segment-name-1 literal-1  [segment-name-2] +
        [KEY (field-name, literal-2, literal-3)]
```

**segment-name-1**

Segment-name-1 is the one-to eight-character name of the segment. This name must correspond to the name of a segment specified in the DBD.

**literal-1**

Literal-1 is a numeric integer which designates the length of the segment.

**segment-name-2**

Segment-name-2 is an optional parameter which designates the parent of segment-name-1. This parameter is not coded for the root segment, but it is necessary for all other segments.

**[KEY]**

The KEY parameter is required for:

- Defining the root segment when using a tickler file.
- All segments of a DL/I database (prior to DL/I 1.6) processed in VSE (except for the lowest segment in a path).

KEY is optional for:

- The RECORD statement that defines the lowest segment in a path.
- Identifying the key field for segments of an IMS database processed in OS/390.

**field-name**

This is the one- to eight- character name used to designate the key field to the IMS/DLI database. The name must correspond to a field named in the segment in the DBD.
Literal-2 is a numeric integer that specifies the location of the key within the segment.

Literal-3 is a numeric integer that specifies the length of the key field.

RETRIEVE Statement

Code the RETRIEVE statement immediately following the JOB statement to describe automatic database input. You can code only one RETRIEVE statement per JOB. Automatic database input is processed in the same manner as non-database input. The syntax of the RETRIEVE statement is illustrated next.

RETRIEVE  file-name-1 +

[KEYFILE  file-name-2, KEYVALUE field-name] +

SELECT  (record-name +
   [ID  literal-1] +
   [LIMIT  literal-2] +
   [WHILE (condition)] +
   ...)

file-name-1

File-name-1 identifies the database being accessed. This is the name coded on the JOB INPUT and FILE statements.

[KEYFILE file-name-2, KEYVALUE field-name]

You can designate the tickler file option by coding both the KEYFILE and the KEYVALUE parameters on the RETRIEVE statement and the KEY parameter on the RECORD statement for the root segment.

- File-name-2 is the name of a file which is sequentially processed to obtain the keys of the root segments to be retrieved.
- Field-name is a data field from file-name-2 which contains the key. The key values are used to retrieve the root segments.

Automatic input is terminated when all of the keys in file-name-2 have been processed.

SELECT

The SELECT parameter identifies which segments (record-name) CA-Easytrieve/Plus retrieves.
record-name

Record-name must be the same as the segment-name coded on a RECORD statement. You can specify any number of record-names for input; however, the parent of each selected segment must also be selected.

[ID literal-1]

ID literal-1 is an optional two-byte alphabetic field which identifies retrieved paths. For example, in the Hierarchical Database Structure exhibit (shown earlier), one path might be designated AB, another as AD.
- The AB path includes two segments: A and B.
- The AD path includes three segments: A, C, and D.

The path ID designations CAN be any two-character alphabetic literals that you choose.

[LIMIT literal-2]

LIMIT literal-2 optionally controls the number of segment occurrences that are retrieved. The LIMIT applies to each path. For example, if it is known that a particular segment never occurs more than two times in a path, code LIMIT 2 for that segment. When you do not code this parameter, CA-Easytrieve/Plus retrieves all qualified occurrences of the segment.

[WHILE (condition)]

WHILE (condition) optionally pre-screens input segments. The syntax of the WHILE condition is exactly the same as the conditional expressions discussed in Chapter 6, “Decision and Branching Logic.”

As the associated segment is returned by IMS/DLI, CA-Easytrieve/Plus evaluates the WHILE condition. Segments are accepted for input only if the WHILE condition is true.

Code the record-name parameter (and optionally the ID, LIMIT, and WHILE subparameters) for every segment of the database to be processed by the JOB.
Automatic Input with RETRIEVE

The RETRIEVE statement performs a sweep of a database (the default) or is used for the tickler file control.

Sweep of a Database

Sweeping the entire database provides the default input. A GN (get next) call is issued at the root level until the database has been exhausted. LIMIT, SSA, or WHILE options, if specified, control the sweep.

Tickler File Control

Optionally, a file of root segment keys can control the extent of the database to be processed. Root segment keys are obtained one-at-a-time from the tickler file. GU (get unique) calls are issued at the root level for each key in the tickler file. GNP calls are issued to obtain all segments associated with the root.

Input Definition (Paths)

Automatic input of IMS/DLI databases uses path processing. Each database path identified by the SELECT parameter is processed in a top-to-bottom, front-to-back, and left-to-right order. A root segment is accessed first; path accessing continues downward to the left until the end of the path. As the end of each path is reached, that data is made available to the program as an input record.

CA-Easytrieve/Plus exhausts each path before proceeding to the next path. When it exhausts the last path, it retrieves the next root and processing begins again with the leftmost path.
Introduction

All CA-Easytrieve/Plus programs require a set of associated commands or statements called Job Control Language (JCL) when they are submitted to be compiled and/or executed. This set of statements defines the components and requirements of the CA-Easytrieve/Plus program to the operating system under which it runs.

JCL is an IBM language described in detail in IBM publications available in your data center. Specifically which statements are supplied is dependent on the files used in your program and which IBM operating system your installation has.

This chapter provides some general information about OS/390 JCL requirements. Examples are provided of the JCL used for the sample programs in Part I of this Application Guide. Within these examples of JCL, material in lowercase letters is dependent on your installation.

Material in uppercase letters is required.
Sample Short Report Output Program

This sample program reads one input file (PERSNL) and outputs one short report. The JCL and CA-Easytrieve/Plus code for this program is illustrated in the exhibit below. The next exhibit illustrates the output report.

```
//jobname JOB (acctng.info),your.name
//stepname EXEC PGM=EZTPA00
//STEPLIB DD DSN=your.load.library,DISP=SHR
//SYSPRINT DD SYSOUT=A
//EZTVFM DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SORTWK01 DD UNIT=SYSDA,SPACE=(4096,500,,,ROUND)
//SYSOUT DD SYSOUT=A
//PERSNL DD DSN=your.input.filename,DISP=SHR
//PANDD1 DD DSN=your.macro.library,DISP=SHR
//SYSIN DD *
PARM DEBUG(FLOW)
*
FILE PERSNL FB(150 1000)
  NAME                     17  16   A
  LAST-NAME            NAME   8   A
  PAY-GROSS                94   4   P  2
  DEPT                     98   3   N
*
JOB INPUT PERSNL
  IF DEPT = 900 THRU 911
    PRINT SHORT-RPT
  END-IF
*
REPORT SHORT-RPT LINESIZE 60 SPREAD NODATE NOPAGE
  SEQUENCE DEPT
  CONTROL DEPT
  TITLE 1 'SALARY REPORT'
  TITLE 2 'DEPARTMENTS 900 - 911'
  HEADING PAY-GROSS ('TOTAL' 'SALARY')
  HEADING LAST-NAME ('EMPLOYEE' 'NAME')
  LINE DEPT LAST-NAME PAY-GROSS
/*
Mailing Label Output Program

The label-generating program discussed in Chapter 8, “Report Processing,” reads one input file (PERSNL) and outputs a set of mailing labels. The JCL and CA-Easytrieve/Plus code for this program is illustrated in the next exhibit.

```
//jobname JOB (acctng.info),your.name
//stepname EXEC PGM=EZTPA00
//STEPLIB DD DSN=your.load.library,DISP=SHR
//SYSPRINT DD SYSOUT=A
//EZTVFM DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SORTWK01 DD UNIT=SYSDA,SPACE=(4096,500,,,ROUND)
//SYSOUT DD SYSOUT=A
//PERSNL DD DSN=your.input.filename,DISP=SHR
//PANDD1 DD DSN=your.macro.library,DISP=SHR
//SYSIN DD *
PARM DEBUG(FLOW)
*
FILE PERSNL FB(150 1800)
 NAME 17 16  A
 LAST-NAME 17 8  A
 FIRST-NAME 25 8  A
 ADDRESS 37 39  A
 STREET 37 20  A
 CITY 57 12  A
 STATE 69 2  A
 ZIP 71 5  N
 DATE-OF-HIRE 136 6  N
 HIRE-MM 136 2  N
 HIRE-DD 138 2  N
 HIRE-YY 140 2  N
 SERVICE W 2  N
```
**Synchronized File Processing Program**

The synchronized file processing program discussed in Chapter 9, “File Processing,” reads two input files (PERSNL and PERSUPD), sorts one file (PERSNL) to another file (SORTPER), outputs one printed report, displays messages to an error file (ERRPRINT), and creates a new master file (NEWPERS). The JCL and CA-Easytrieve/Plus code for this program is illustrated in the next exhibit.

```plaintext
//jobname  JOB (acctng.info),your.name
//stepname EXEC  PGM=EZTPA00
//STEPLIB   DD DSN=your.load.library,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//SYSOUT    DD SYSOUT=A
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SORTWK01  DD UNIT=SYSDA,SPACE=(4096,500,,,ROUND)
//ERRPRINT  DD SYSOUT=A
//PERSNL    DD DSN=your.old.filename,DISP=SHR
//NEWPERS   DD DSN=your.new.filename,DISP=(NEW,CATLG),UNIT=SYSDA,SPACE=(1800,(50,100),RLSE)
//PANDD1    DD DSN=your.macro.library,DISP=SHR
//SYSIN     DD *
//PARM  DEBUG(FLOW)
*   FILE PERSNL FB(150 1800)
    OLD-EMP#          9   5   N
```

```plaintext
CURR-DATE          5   6   N
CURR-MM            2   N
CURR-DD            2   N
CURR-YY            2   N
*                   
JOB INPUT PERSNL
%GETDATE CURR-DATE
PERFORM SERVICE-CALC
IF SERVICE GT 19
  PRINT MAILOUT
END-IF
*                   
SERVICE-CALC. PROC
  SERVICE = CURR-YY - HIRE-YY
  IF CURR-MM < HIRE-MM
    SERVICE = SERVICE - 1
  END-IF
  IF CURR-MM NE HIRE-MM
    GOTO QUIT-SERV-CALC
  END-IF
  IF CURR-DD < HIRE-DD
    SERVICE = SERVICE - 1
  END-IF
  QUIT-SERV-CALC
END-PROC
*                   
REPORT MAILOUT LABELS (ACROSS 2 DOWN 4 SIZE 30)
  SEQUENCE LAST-NAME
  LINE 1 FIRST-NAME -3 LAST-NAME
  LINE 2 STREET
  LINE 3 CITY -3 STATE ZIP
/*
```
FILE PERSUPD CARD
   EMP#          1      5      N
   RAISE-PERCENT  7      2      N
*
FILE SORTPER F 150 VIRTUAL
   UPD-EMP#      9      5      N
   NAME         17     8      A
   PAY-GROSS    94     4      P  2
   NEWSAL       W     4      P  2
*
FILE NEWPERS FB(150 1800)
*
FILE ERRPRINT  PRINTER
*
SORT PERSNL TO SORTPER USING OLD-EMP#
*
JOB INPUT (SORTPER KEY(UPD-EMP#) +
   PERSUPD KEY(EMP#) )
*
IF MATCHED
   NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)
   PRINT NEW-RPT
   PAY-GROSS = NEWSAL
END-IF
IF SORTPER
   PUT NEWPERS FROM SORTPER
ELSE
   DISPLAY ERRPRINT EMP# ' RECORD NOT FOUND'
END-IF
*
REPORT  NEW-RPT LINESIZE 80 NOPAGE NODATE
   SEQUENCE NAME
   TITLE 1  'SALARY UPDATE REPORT'
   TITLE 2  'EMPLOYEES WITH OVER 25 YEARS SERVICE'
   HEADING UPD-EMP# ('EMPL' 'NUMBER')
   HEADING NAME    ('EMPL' 'NAME')
   HEADING PAY-GROSS ('OLD' 'SALARY')
   HEADING NEWSAL  ('NEW' 'SALARY')
   HEADING RAISE-PERCENT ('RAISE' '%')
   LINE  UPD-EMP# NAME PAY-GROSS NEWSAL RAISE-PERCENT
*
END
01730 08
04225 09
09481 09
11473 11
11710 10
12267 12
*/

SORTPER, the sort output file, is not defined in the JCL because it is a temporary VIRTUAL file. PERSUPD, the input employee number file, is also not defined in the JCL, since it is a CARD file whose data is obtained from the records after the END statement following the CA-Easytrieve/Plus program.
Compile and Link-Edit Load Module

The next exhibit illustrates the JCL to compile and link-edit a load module for later execution.

```
//jobname  JOB (acctng.info),your.name
//stepname EXEC  PGM=EZTPA00
//SYSPRINT DD SYSOUT=A
//EZTVFM DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SYSLIN DD UNIT=SYSDA,SPACE=800,(50,50),DISP=(,PASS),
//        DSN=&&SYSLIN
//SYSIN  DD *
//PARM  LINK(TESTPGM)....
 //...EASYTRIEVE/PLUS source statements....
//LKED     EXEC  PGM=IEWL
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(6144,(50,50),,,ROUND)
//SYSLIN DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//SYSLMOD DD DSN=your.load.library,DISP=SHR
```
Introduction

All CA-Easytrieve/Plus programs require a set of associated commands or statements called Job Control Language (JCL) when they are submitted to be compiled and/or executed. This set of statements defines the components and requirements of the CA-Easytrieve/Plus program to the operating system under which it runs.

JCL is an IBM language described in detail in IBM publications available in your data center. Specifically which statements are supplied is dependent on the files used in your program and which IBM operating system your installation has.

This chapter provides some general information about VSE JCL requirements. Examples are provided of the JCL used for the sample programs in Part I of this Application Guide. Within these examples of JCL, material in lowercase letters is dependent on your installation.

Material in uppercase letters is required.
This sample program reads one input file (PERSNL) and outputs one short report. The JCL and CA-Easytrieve/Plus code for this program is illustrated in the exhibit below. The next exhibit illustrates the output report.

```
* ** JOB JNM=jobname
// JOB   jobname
// DLBL   EZTP,'your.eztp.sysclb',0,SD
// EXTENT SYS003,volser.1,0.start.lgth
// ASSGN  SYS003,nnn
// LIBDEF CL,SEARCH=EZTP,TEMP
// ASSGN  SYS001,nnn
// ASSGN  SYS006,nnn
// ASSGN  SYS008,nnn
// ASSGN  SYS010,nnn
// DLBL   SORTWK1...,0,DA
// EXTENT SYS001.volser...,start.lgth
// DLBL   PANDD1,'your.macro.library',0,SD
// EXTENT SYS006.volser...,start.lgth
// DLBL   PERSNL,'your.input.filename',0,SD
// EXTENT SYS008.volser...,start.lgth
// DLBL   EZTVFM...,0,SD
// EXTENT SYS010.volser...,start.lgth
// EXEC   EZTPA00,SIZE=200K
PARM  DEBUG(FLOW)
*
FILE PERSNL FB(150 1800)
NAME                     17  16   A
LAST-NAME            NAME   8   A
PAY-GROSS                94   4   P  2
DEPT                     98   3   N
*
JOB INPUT PERSNL
IF DEPT = 900 THRU 911
PRINT SHORT-RPT
END-IF
*
REPORT SHORT-RPT LINESIZE 60 SPREAD NODATE NOPAGE
SEQUENCE DEPT
CONTROL DEPT
TITLE 1 'SALARY REPORT'
TITLE 2 'DEPARTMENTS 900 - 911'
HEADING PAY-GROSS ('TOTAL' 'SALARY')
HEADING LAST-NAME ('EMPLOYEE' 'NAME')
LINE DEPT LAST-NAME PAY-GROSS
/*
&*
*$ $ E0J
```
### Mailing Label Output Program

The label-generating program discussed in Chapter 8, “Report Processing,” reads one input file (PERSNL) and outputs a set of mailing labels. The JCL and CA-Easytrieve/Plus code for this program is illustrated in the next exhibit.

```plaintext
* $ JOB JNM=jobname
 // JOB jobname
 // DLBL EZTP,'your.eztp.sysclb',0,SD
 // EXTENT SYS003,volser1,0,start.lgth
 // ASSGN SYS003,nnn
 // LIBDEF CL,SEARCH=EZTP,TEMP
 // ASSGN SYS001,nnn
 // ASSGN SYS006,nnn
 // ASSGN SYS008,nnn
 // ASSGN SYS010,nnn
 // DLBL SORTWK1,DA
 // EXTENT SYS001,volser...start.lgth
 // DLBL PANDD1,'your.macro.library',0,SD
 // EXTENT SYS006,volser...start.lgth
 // DLBL PERSNL,'your.input.filename',0,SD
 // EXTENT SYS008,volser...start.lgth
 // DLBL EZTVFM,0,SD
 // EXTENT SYS010,volser...start.lgth
 // EXEC EZTPA00,SIZE=200K
 PARM DEBUG(FLOW)
 *
```
FILE PERSNL FB(150 1800)
NAME 17 16 A
LAST-NAME 17 8 A
FIRST-NAME 25 8 A
ADDRESS 37 39 A
STREET 37 20 A
CITY 57 12 A
STATE 69 2 A
ZIP 71 5 N
DATE-OF-HIRE 136 6 N
HIRE-MM 136 2 N
HIRE-DD 138 2 N
HIRE-YY 140 2 N
SERVICE W 2 N
CURR-DATE S 6 N
CURR-MM CURR-DATE 2 N
CURR-DD CURR-DATE +2 2 N
CURR-YY CURR-DATE +4 2 N

* JOB INPUT PERSNL
%GETDATE CURR-DATE
PERFORM SERVICE-CALC
IF SERVICE GT 19
PRINT MAILOUT
END-IF
*
SERVICE-CALC. EZTC
SERVICE = CURR-YY - HIRE-YY
IF CURR-MM < HIRE-MM
SERVICE = SERVICE - 1
END-IF
IF CURR-MM NE HIRE-MM
GOTO QUIT-SERV-CALC
END-IF
IF CURR-DD < HIRE-DD
SERVICE = SERVICE - 1
END-IF
QUIT-SERV-CALC
END-EZTC
*
REPORT MAILOUT LABELS (ACROSS 2 DOWN 4 SIZE 30)
SEQUENCE LAST-NAME
LINE 1 FIRST-NAME -3 LAST-NAME
LINE 2 STREET
LINE 3 CITY -3 STATE ZIP
/
&
* $$ EOJ
The synchronized file processing program discussed in Chapter 9, “File Processing,” reads two input files (PERSNL and PERSUPD), sorts one file (PERSNL) to another file (SORTPER), outputs one printed report, displays messages to an error file (ERRPRINT), and creates a new master file (NEWPERS). The JCL and CA-Easytrieve/Plus code for this program is illustrated in the next exhibit.

* $ JOB JNM=jobname
  // JOB jobname
  // DLBL EZTP,'your.eztp.sysclb'.0,SD
  // EXTENT SYS003.volser.,1,0.start.lgth
  // ASSGN SYS003.nnn
  // LIBDEF CL,SEARCH=EZTP,TEMP
  // ASSGN SYS001.nnn
  // ASSGN SYS006.nnn
  // ASSGN SYS008.nnn
  // ASSGN SYS010.nnn
  // ASSGN SYS011.SYSLST
  // DLBL SORTWK1.,0,DA
  // EXTENT SYS001.volser...,start.lgth
  // DLBL PANDD1,'your.macro.library'.0,SD
  // EXTENT SYS006.volser...,start.lgth
  // DLBL PERSNL,'your.input.filename'.0,SD
  // EXTENT SYS006.volser...,start.lgth
  // DLBL NEWPERS,'your.new.filename'.0,SD
  // EXTENT SYS009.volser...,start.lgth
  // DLBL EZTVFM.,0,SD
  // EXTENT SYS010.volser...,start.lgth
  // EXEC EZTPA00.SIZE=200K
  PARM DEBUG(FLOW)
  *
  FILE PERSNL FB(150 1800)
    OLD-EMP#  9 5 N
  *
  FILE PERSUPD CARD
    EMP#  1 5 N
    RAISE-PERCENT  7 2 N
  *
  FILE SORTPER F 150 VIRTUAL
    UPD-EMP#  9 5 N
    NAME 17 8 A
    PAY-GROSS 94 4 P 2
    NEWSAL  W 4 P 2
  *
  FILE NEWPERS FB(150 1800)
  *
  FILE ERRPRINT PRINTER
  *
  SORT PERSNL TO SORTPER USING OLD-EMP#
JOB INPUT (SORTPER KEY(UPD-EMP#) + 
PERSUPD KEY(EMP#) ) 
  * 
  IF MATCHED 
      NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100) 
      PRINT NEW-RPT 
      PAY-GROSS = NEWSAL 
  END-IF 
  IF SORTPER 
      PUT NEWPERS FROM SORTPER 
  ELSE 
      DISPLAY ERRPRINT EMP# ' RECORD NOT FOUND' 
  END-IF 
  * 
REPORT NEW-RPT LINESIZE 80 NOPAGE NODATE 
SEQUENCE NAME 
TITLE 1   'SALARY UPDATE REPORT' 
TITLE 2   'EMPLOYEES WITH OVER 25 YEARS SERVICE' 
HEADING UPD-EMP#    ('EMPL' 'NUMBER') 
HEADING NAME       ('EMPL' 'NAME') 
HEADING PAY-GROSS   ('OLD' 'SALARY') 
HEADING NEWSAL      ('NEW' 'SALARY') 
HEADING RAISE-PERCENT ('RAISE' '%') 
LINE UPD-EMP# NAME PAY-GROSS NEWSAL RAISE-PERCENT 
  * 
END 
01730 08 
04225 09 
09481 09 
11473 11 
11710 10 
12267 12 
/*/ 
*/& 
* $S EOJ 

SORTPER, the sort output file, is not defined in the JCL because it is a temporary VIRTUAL file. PERSUPD, the input employee number file, is also not defined in the JCL, since it is a CARD file whose data is obtained from the records after the END statement following the CA-Easytrieve/Plus program.
Compile and Link-Edit Load Module

The next exhibit illustrates the JCL to compile and link-edit a load module for later execution.

```jcl
* $$ JOB JNM=jobname
// JOB jobname
// DLBL EZTP,'your.eztp.library',0,SD
// EXTENT SYS003,volser,1,0,start,lgth
// ASSGN SYS003,nnn
// LIBDEF PHASE,CATALOG=EZTP.sublib,TEMP
// LIBDEF PHASE,SEARCH=EZTP.sublib,TEMP
// ASSGN SYS010,...
// DLBL EZTVFM.,0,SD
// EXTENT SYS010,volser...start,lgth
// OPTION CATAL
// EXEC EZTPA00,SIZE=512K
PARM LINK(TESTPGM)
...CA-Easytrieve/Plus source statements...
/*
// EXEC LNKEDT
/&
* $$ EOJ
```

Previously Compiled and Link-Edited Programs

The next exhibit illustrates the JCL to execute a previously compiled and link-edited program named TESTPGM.

```jcl
* $$ JOB JNM=jobname
// JOB jobname
// DLBL EZTP,'your.eztp.library':0,SD
// EXTENT SYS003,volser,1,0,start,lgth
// ASSGN SYS003,nnn
// LIBDEF PHASE,SEARCH=EZTP.sublib,TEMP
// ASSGN SYS001,...
// ASSGN SYS010,...
// ASSGN SYS008,...
// DLBL SORTWK1.,0,DA
// EXTENT SYS001,volser...start,lgth
// DLBL EZTVFM.,0,SD
// EXTENT SYS010,volser...start,lgth
// DLBL INREC.,0,SD
// EXTENT SYS008,volser...start,lgth
// EXEC TESTPGM,SIZE=512K
...optional CARD input...
/*
/&
* $$ EOJ
```
Introduction

Part II of the Application Guide is a composite of sample CA-Easytrieve/Plus jobs which perform typical data processing functions. The examples are presented in two forms:

- The second form implements entire application systems. Chapter 17, “Bank System,” and Chapter 18, “Project Management System,” contain mini-applications that demonstrate the wide scope of CA-Easytrieve/Plus capabilities.

A cross-reference of the examples in Part II and CA-Easytrieve/Plus statements is provided in Appendix B. This provides an easy way to review all of the examples that use a particular feature.

Scan the examples to get a feel for the language. If you find an example similar to your needs, use the ideas and the code (if possible) to implement your solution.

Application Overview

Four distinct applications are the basis for all of the examples. They are:

- Personnel System
- Inventory System
- Bank Customer System
- Project Management System.

Each of these applications has its own master and auxiliary files. Each file structure is defined in the description of the particular system.
In general, the applications presented are not intended to be usable in a real-world environment (except possibly the Project Management System). They are intended to provide you with ideas for developing your own programs and systems.

The Personnel System and Inventory System are sample files to be used by the sample programs in later chapters. The Bank Customer and Project Management Systems are complete, working mini-systems.

**Program Formatting Standards**

The CA-Easytrieve/Plus statements in the examples are coded in a standard format. FILE, JOB, and REPORT statements are coded in column one. All other statements are indented two columns for each logical level. Vertical spacing is used between FILE definitions, JOB activities, and REPORT subactivities.

These guidelines help make the programs more readable. Similar guidelines used in the development of your programs enable the logic and structure of the programs to be easily discerned. Liberally supplying meaningful comments can make program maintenance much easier.

**Program Output Standards**

CA-Easytrieve/Plus can generate a report in nearly any format. In this guide, we limit all reports to a maximum of 80-character print lines to accommodate our page size. Wider print lines enable you considerable flexibility in this area.

Remember to route DISPLAY information from a JOB activity to a different file than your reports are routed. This prevents DISPLAY output from being interspersed with your report. Also, try to use DISPLAY from your JOB activity only for abnormal condition messages - REPORT should be used for all quality output.

It is a good practice to TITLE your reports with something meaningful and to include the current date in the title. Most installations have report format standards; CA-Easytrieve/Plus should be compatible with your standards.
Inventory Sample File

The Inventory sample file is the basis for many of the examples in the following chapters. Following is a macro listing of %INVMSTR that provides the field definitions for the Inventory file in the examples. These field definitions are not repeated in the examples. Refer to this chapter when studying examples which use the Inventory file.

MACRO

*    INVENTORY MASTER FIELD DEFINITIONS
*
PART-INFO               1  43 A
   PART-DESCRIPTION               1 35 A -
      HEADING('PART DESCRIPTION')
   PART-NUMBER                   36  8 N     MASK '999-99-999' -
      HEADING('PART ' NUMBER')
*
LOCATION-INFO          44  18 A
   LOCATION-CITY                 44  7 A   HEADING 'CITY'
   LOCATION-STATE                51  2 A   HEADING 'STATE'
   LOCATION-CODE                 53  3 P   HEADING 'CODE'
   LOCATION-BAY                  56  1 A   HEADING 'BAY'
   LOCATION-BIN                  57  3 N   HEADING 'BIN'
   LOCATION-LEVEL                60  2 N   HEADING 'LEVEL'
*
ITEM-INFO              62  29 A
   ITEM-SELLING-PRICE            62  4 P 2 -
      HEADING('SELLING ' PRICE ' (DOLLARS)')
   ITEM-REORDER-POINT            66  4 N 0 -
      HEADING('REORDER ' POINT')
   ITEM-LAST-SALE-DATE           70  6 N   MASK(D 'Z9/99/99') -
      HEADING('LAST SALE ' DATE')
   ITEM-LAST-INVENTORY-DATE      76  6 N   MASK D -
      HEADING('LAST ' INVENTORY ' DATE')
   ITEM-LAST-INVENTORY-QUANTITY  82  4 P 0 -
      HEADING('LAST ' INVENTORY ' QUANTITY')
   ITEM-MFGD-COMMODITY-GROUP     86  3 P -
      HEADING('MFGD ' COMMODITY ' GROUP')
   ITEM-WEIGHT-POUNDS            89  2 P 0 MASK 'ZZ9 #' -
      HEADING('WEIGHT ' (POUNDS))
*
LAST-PURCHASE-INFO     91 13 A
   LAST-PURCHASE-QUANTITY        91  3 P 0 -
      HEADING('LAST ' PURCHASE ' QUANTITY')
   LAST-PURCHASE-PRICE           94  4 P 2 -
      HEADING('LAST ' PURCHASE ' PRICE')
   LAST-PURCHASE-DATE            98  6 N   MASK D -
      HEADING('LAST ' PURCHASE ' DATE')
*
VENDOR-INFO            104  17 A
   VENDOR-NUMBER                104  8 N   MASK '99-99-9-999' -
      HEADING('VENDOR ' NUMBER')
   VENDOR-LOCATION-CITY         112  7 A   HEADING('VENDOR ' CITY')
   VENDOR-LOCATION-STATE        119  2 A   HEADING('VENDOR ' STATE')
*
SHIPPING-INFO         121  6 A
   SHIPPING-FOB-CODE            121  2 P   HEADING('FOB ' CODE')
   SHIPPING-CARRIER-ALPHA-CODE  123  4 A   HEADING('CARRIER ' CODE')
Personnel Sample File

The Personnel sample file is the basis for many of the examples in the following chapters. The following JOB lists the contents of the file. The field definitions are imbedded as macro &PERSNL. These field definitions are not repeated in the examples. Refer to this chapter when studying an example which uses the Personnel file.

1 *       PERSONNEL MASTER FILE LISTING
2 *       FILE PERSNL FB(150 1800)
3 %PERSNL
4 *       TEST FILE FIELD DEFINITIONS
5 *
6 REGION  1 1 N
7 BRANCH   2 2 N
8 SSN      4 5 P   MASK '999-99-9999' -
9 EMP#     9 5 N   HEADING('EMPLOYEE' 'NUMBER')
10 NAME    17 16 A   HEADING 'EMPLOYEE NAME'
11     NAME-LAST NAME     8 A   HEADING('LAST' 'NAME')
12     NAME-FIRST NAME +8  8 A   HEADING('FIRST' 'NAME')
13 ADDRESS  37 39 A
14     ADDR-STREET 37 20 A   HEADING 'STREET'
15     ADDR-CITY   57 12 A   HEADING 'CITY'
16     ADDR-STATE  69  2 A   HEADING 'STATE'
17     ADDR-ZIP    71  5 N   HEADING('ZIP' 'CODE')
18 PAY-NET   90  4 P 2   HEADING('NET' 'PAY')
19 PAY-GROSS 94  4 P 2   HEADING('GROSS' 'PAY')
20 DEPT     98  3 N
21 DATE-OF-BIRTH 103 6 N   MASK(Y 'Z9/99/99') -
22 TELEPHONE 117 10 N   HEADING('DATE' 'OF' 'BIRTH')
23     TELEPHONE '999-9999' -
24 SEX      127 1 N   HEADING('SEX' 'CODE')
25     * 1 - FEMALE
26     * 2 - MALE
27 MARITAL-STAT 128 1 A   HEADING('MARITAL' 'STATUS')
28     * M - MARRIED
29     * S - SINGLE
30 JOB-CATEGORY 132 2 N   HEADING('JOB' 'CATEGORY')
31 SALARY-CODE 134 2 N   HEADING('SALARY' 'CODE')
32 DATE-OF-HIRE 136 6 N   MASK Y -
33     HEADING('DATE' 'OF' 'HIRE')
34 *
35 JOB INPUT PERSNL
36 PRINT PERSNL-LIST
37 *
38 REPORT PERSNL-LIST SKIP 1 SPACE 1 LINESIZE 80
39 *
40 TITLE 'NEW PERSONNEL SAMPLE FILE LISTING'
41 *
42 HEADING REGION ('R' 'G' 'N')
43 HEADING BRANCH ('BRCH')
44 HEADING EMP# ('EMPL' 'NUMBER')
45 HEADING SSN ('SOCIAL SECURITY' 'NUMBER/' 'TELEPHONE')
46 HEADING PAY-GROSS ('PAY - ' 'GROSS/' 'NET')
47 HEADING SEX ('SEX/' 'M/S')
48 HEADING DEPT ('DPT/' 'J*C/ ' 'S*C')
49 HEADING DATE-OF-BIRTH ('DATE OF' 'BIRTH/' 'HIRE')
<table>
<thead>
<tr>
<th>R</th>
<th>SOCIAL SECURITY NUMBER/BRANCH</th>
<th>TELEPHONE NUMBER</th>
<th>EMPLOYEE NAME</th>
<th>PAY - GROSS/NET</th>
<th>DPT/DEPT</th>
<th>DATE OF BIRTH/SEX/M/S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01 625-30-5228 (617) 332-2762</td>
<td>12267 430 M ST 107</td>
<td>WIMN GLORIA</td>
<td>373.60 903</td>
<td>5/2/2/30 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>251.65 10</td>
<td>7/12/51 S</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>02 121-16-6413 (301) 636-8995</td>
<td>11473 3710 JENIFER ST</td>
<td>BERG NANCY</td>
<td>759.20 943</td>
<td>8/15/31 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>547.88 25</td>
<td>6/17/55 M</td>
<td>03</td>
</tr>
<tr>
<td>1</td>
<td>03 228-58-8307 (609) 444-7688</td>
<td>02688 3208 S 5TH</td>
<td>CORNING GEORGE</td>
<td>146.16 915</td>
<td>10/12/52 2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>103.43 40</td>
<td>11/08/70 S</td>
<td>06</td>
</tr>
<tr>
<td>1</td>
<td>02 256-52-8737 (301) 636-8995</td>
<td>08370 826 D STREET</td>
<td>NAGLE MARY</td>
<td>554.40 935</td>
<td>1/13/43 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>340.59 10</td>
<td>3/18/73 S</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>04 281-36-2873 (212) 451-4040</td>
<td>01963 1569 COLONIAL TERR</td>
<td>ARNOLD LINDA</td>
<td>445.50 911</td>
<td>8/29/42 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>356.87 10</td>
<td>10/19/68 S</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>03 298-34-4755 (609) 444-3094</td>
<td>11602 1305 POTOMAC ST</td>
<td>MANHART VIRGINIA</td>
<td>344.80 914</td>
<td>10/13/38 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>250.89 60</td>
<td>7/06/65 S</td>
<td>08</td>
</tr>
<tr>
<td>1</td>
<td>04 322-30-0050 (212) 451-4531</td>
<td>11931 1412 36TH ST</td>
<td>TALL ELAINE</td>
<td>492.26 917</td>
<td>12/25/40 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>355.19 40</td>
<td>10/16/62 S</td>
<td>06</td>
</tr>
<tr>
<td>1</td>
<td>01 452-52-1419 (617) 332-6701</td>
<td>02200 3616 B ST S E</td>
<td>BRANDOW LYDIA</td>
<td>804.64 918</td>
<td>9/14/47 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>554.31 10</td>
<td>7/11/72 S</td>
<td>01</td>
</tr>
<tr>
<td>1</td>
<td>04 554-70-3189 (212) 451-7382</td>
<td>11357 610 H ST SW</td>
<td>LARSON RODNEY</td>
<td>283.92 911</td>
<td>2/11/39 2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>215.47 25</td>
<td>8/20/66 S</td>
<td>03</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ BRANCH</td>
<td>TELEPHONE NUMBER</td>
<td>EMPLOYEE NAME</td>
<td>NET PAY</td>
<td>GROSS PAY</td>
<td>DEPARTMENT/ DATE OF HIRE</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------</td>
<td>-----------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1</td>
<td>04 577-20-0461 (212) 773-0799</td>
<td>11467</td>
<td>BYER JULIE</td>
<td>396.68</td>
<td>259.80</td>
<td>04/17/39 1</td>
</tr>
<tr>
<td>2</td>
<td>01 579-12-0813 (813) 796-1189</td>
<td>11376</td>
<td>HUSS PATTI</td>
<td>360.80</td>
<td>223.71</td>
<td>03/13/42 1</td>
</tr>
<tr>
<td>2</td>
<td>02 579-50-4818 (404) 832-8081</td>
<td>11710</td>
<td>POWELL CAROL</td>
<td>243.20</td>
<td>167.96</td>
<td>11/10/37 1</td>
</tr>
<tr>
<td>2</td>
<td>03 008-28-7725 (202) 715-0484</td>
<td>04234</td>
<td>MCMAHON BARBARA</td>
<td>386.40</td>
<td>283.19</td>
<td>12/08/45 2</td>
</tr>
<tr>
<td>2</td>
<td>03 120-32-5734 (202) 715-0389</td>
<td>03416</td>
<td>FORREST BILL</td>
<td>13.80</td>
<td>13.19</td>
<td>07/17/63 1</td>
</tr>
<tr>
<td>2</td>
<td>04 190-32-2181 (904) 986-0834</td>
<td>00445</td>
<td>POST JEAN</td>
<td>292.00</td>
<td>206.60</td>
<td>2/15/45 1</td>
</tr>
<tr>
<td>2</td>
<td>03 212-48-5461 (202) 715-1914</td>
<td>00577</td>
<td>PETRIK KATHY</td>
<td>226.80</td>
<td>154.70</td>
<td>3/17/41 1</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ G</td>
<td>EMPLOYER NUMBER/ N</td>
<td>EMPLOYEE NAME</td>
<td>PAY - DPT/ GROSS/ J*C/</td>
<td>DATE OF BIRTH/ S*C</td>
<td>HIRE/ M/S</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------</td>
<td>---------------------</td>
<td>----------------</td>
<td>------------------------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td>2 05</td>
<td>235-72-1049</td>
<td>01895 (919) 489-1614</td>
<td>VETTER DENISE</td>
<td>7311 KEYSTONE LANE RALEIGH NC 27591</td>
<td>279.36 914 7/25/37</td>
<td>1</td>
</tr>
<tr>
<td>2 02</td>
<td>284-36-5652</td>
<td>03571 (404) 832-1776</td>
<td>KRUSE MAX</td>
<td>2161 N PIERCE ST ATLANTA GA 30345</td>
<td>242.40 911 1/01/42</td>
<td>2</td>
</tr>
<tr>
<td>2 05</td>
<td>310-44-5370</td>
<td>04225 (919) 489-5531</td>
<td>LOYAL NED</td>
<td>17 KENNEDY STREET RALEIGH NC 27516</td>
<td>295.20 912 5/06/19</td>
<td>2</td>
</tr>
<tr>
<td>2 03</td>
<td>362-48-0393</td>
<td>02765 (202) 715-1832</td>
<td>DENNING RALPH</td>
<td>1629 16TH ST NW APT WASHINGTON DC 20005</td>
<td>135.85 919 11/12/49</td>
<td>2</td>
</tr>
<tr>
<td>3 01</td>
<td>570-10-5594</td>
<td>04132 (816) 581-1352</td>
<td>WEST KATHY</td>
<td>1728 IRVING ST N W KANSAS CITY KS 66015</td>
<td>736.00 940 4/04/33</td>
<td>1</td>
</tr>
<tr>
<td>3 02</td>
<td>577-09-1160</td>
<td>01743 (214) 941-1441</td>
<td>THOMPSON JANICE</td>
<td>7752 EMERSON RD DALLAS TX 75235</td>
<td>259.40 923 6/23/32</td>
<td>1</td>
</tr>
<tr>
<td>3 02</td>
<td>578-38-7587</td>
<td>01730 (214) 941-1585</td>
<td>SMOTH CINDY</td>
<td>4120 18TH STREET NE DALLAS TX 75219</td>
<td>315.20 911 5/21/38</td>
<td>1</td>
</tr>
<tr>
<td>3 03</td>
<td>578-54-3178</td>
<td>03936 (312) 646-0934</td>
<td>NORIDGE DEBBIE</td>
<td>4264 E CAPITOL NE CHICAGO IL 60652</td>
<td>324.00 944 3/02/36</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ BRANCH</td>
<td>EMPLOYEE NAME</td>
<td>PAY - GROSS/ NET</td>
<td>DPT/ DATE OF BIRTH/ SEX/</td>
<td>PHONE NUMBER</td>
<td>ADDRESS</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>---------</td>
</tr>
<tr>
<td>3 03</td>
<td>579-50-4170 (312) 646-1650</td>
<td>ROGERS PAT</td>
<td>329.00</td>
<td>4/19/41 M</td>
<td>(312) 646-1650</td>
<td>1625 FRANKLIN ST N, CHICAGO IL 60691</td>
</tr>
<tr>
<td>3 04</td>
<td>208-28-2315 (612) 588-1900</td>
<td>GREEN BRENDA</td>
<td>365.60</td>
<td>7/21/28 M</td>
<td>(612) 588-1900</td>
<td>2671 DOUGLAS PL S, MINNEAPOLIS MN 55319</td>
</tr>
<tr>
<td>3 01</td>
<td>231-68-9995 (816) 581-0031</td>
<td>KELLY KEITH</td>
<td>197.60</td>
<td>9/11/34 S</td>
<td>(816) 581-0031</td>
<td>211 E GLEBE RD #C, KANSAS CITY KS 66201</td>
</tr>
<tr>
<td>3 02</td>
<td>418-46-1872 (214) 941-0558</td>
<td>ISAAC RUTH</td>
<td>313.60</td>
<td>6/28/29 S</td>
<td>(214) 941-0558</td>
<td>2639 15TH ST NW, DALLAS TX 75213</td>
</tr>
<tr>
<td>3 03</td>
<td>467-56-4149 (312) 646-1891</td>
<td>STRIDE ANN</td>
<td>386.40</td>
<td>2/29/39 S</td>
<td>(312) 646-1891</td>
<td>325 C STREET SE APT, CHICAGO IL 60619</td>
</tr>
<tr>
<td>3 04</td>
<td>477-44-4948 (612) 588-8991</td>
<td>MALLOW TERRY</td>
<td>282.40</td>
<td>9/11/39 S</td>
<td>(612) 588-8991</td>
<td>2515 K STREET NW AP, MINNEAPOLIS MN 55329</td>
</tr>
<tr>
<td>3 02</td>
<td>215-36-5852 (214) 986-1981</td>
<td>LACH LORRIE</td>
<td>310.40</td>
<td>7/19/41 S</td>
<td>(214) 986-1981</td>
<td>3419 LORRING DRIVE, DALLAS TX 75218</td>
</tr>
<tr>
<td>3 03</td>
<td>228-46-5157 (312) 588-5118</td>
<td>EPERT LINDA</td>
<td>310.40</td>
<td>4/13/44 S</td>
<td>(312) 588-5118</td>
<td>1440 ROCK CREEK FOR, CHICAGO IL 60609</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ EMPLOYEE NAME</td>
<td>PAY - GROSS/ NET S/C</td>
<td>DPT/ DATE OF BIRTH/ SEX/ HIRE</td>
<td>BRANCH TELEPHONE NUMBER</td>
<td>EMPLOYEE NAME</td>
<td>PAY - GROSS/ NET S/C</td>
</tr>
<tr>
<td>----</td>
<td>--------------------------------------</td>
<td>-----------------------</td>
<td>------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>3</td>
<td>03 269-24-7428 OSMON SAMUEL</td>
<td>628.00 935 6/13/42 2</td>
<td>03 269-24-7428 OSMON SAMUEL</td>
<td>(312) 588-8995</td>
<td>CHICAGO IL 60418</td>
<td>628.00 935 6/13/42 2</td>
</tr>
<tr>
<td>3</td>
<td>02 388-18-6119 GRECO LESLIE</td>
<td>1,004.00 914 3/19/45 1</td>
<td>02 388-18-6119 GRECO LESLIE</td>
<td>(214) 399-7688</td>
<td>DALLAS TX 75227</td>
<td>1,004.00 914 3/19/45 1</td>
</tr>
<tr>
<td>3</td>
<td>04 577-16-2985 CROCI JUDY</td>
<td>376.00 914 5/22/43 1</td>
<td>04 577-16-2985 CROCI JUDY</td>
<td>(553) 444-1970</td>
<td>MINNEAPOLIS MN 55339</td>
<td>376.00 914 5/22/43 1</td>
</tr>
<tr>
<td>3</td>
<td>02 061-30-8680 REYNOLDS WILLIAM</td>
<td>174.15 911 11/03/31 2</td>
<td>02 061-30-8680 REYNOLDS WILLIAM</td>
<td>(214) 399-4840</td>
<td>DALLAS TX 75244</td>
<td>174.15 911 11/03/31 2</td>
</tr>
<tr>
<td>3</td>
<td>01 090-22-9192 PHILIPS SUE</td>
<td>251.26 940 12/09/48 1</td>
<td>01 090-22-9192 PHILIPS SUE</td>
<td>(816) 836-3084</td>
<td>KANSAS CITY KS 66083</td>
<td>251.26 940 12/09/48 1</td>
</tr>
<tr>
<td>3</td>
<td>01 118-34-8805 YOUNG JANE</td>
<td>313.60 911 7/08/43 1</td>
<td>01 118-34-8805 YOUNG JANE</td>
<td>(816) 836-4531</td>
<td>KANSAS CITY KS 66054</td>
<td>313.60 911 7/08/43 1</td>
</tr>
<tr>
<td>3</td>
<td>03 140-32-0779 MILLER JOAN</td>
<td>313.60 920 1/27/47 1</td>
<td>03 140-32-0779 MILLER JOAN</td>
<td>(312) 586-6781</td>
<td>CHICAGO IL 60643</td>
<td>313.60 920 1/27/47 1</td>
</tr>
<tr>
<td>4</td>
<td>01 216-44-7756 TALUS RUTH</td>
<td>460.00 944 10/06/33 1</td>
<td>01 216-44-7756 TALUS RUTH</td>
<td>(206) 225-3828</td>
<td>SEATTLE WA 98003</td>
<td>460.00 944 10/06/33 1</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ BRANCH</td>
<td>TELEPHONE NUMBER</td>
<td>EMPLOYEE NAME</td>
<td>PAY - DATE OF BIRTH NET</td>
<td>DPT/ DATE OF BIRTH/ SEX</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------</td>
<td>-----------------</td>
<td>---------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>484-30-8293 (714) 771-0799</td>
<td>06239</td>
<td>JOHNSON LISA</td>
<td>712.80 942 2/05/46 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>577-58-0363 (206) 225-9127</td>
<td>05482</td>
<td>WARD MARINA</td>
<td>183.75 921 9/12/23 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>579-62-1768 (213) 493-5966</td>
<td>04935</td>
<td>ZOLTAN JANET</td>
<td>125.00 924 8/18/28 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>060-26-8978 (213) 493-0979</td>
<td>10949</td>
<td>JONES ALFRED</td>
<td>804.80 940 11/13/39 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>104-20-0956 (714) 771-9876</td>
<td>09764</td>
<td>HAFER ARTHUR</td>
<td>121.95 911 8/03/40 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>448-24-6593 (415) 278-1753</td>
<td>10260</td>
<td>JUDAR PAULA</td>
<td>591.20 943 10/12/50 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>537-03-4839 (213) 725-6495</td>
<td>11211</td>
<td>WALTERS KAREN</td>
<td>424.00 901 1/18/42 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>558-44-7689 (206) 225-8456</td>
<td>10961</td>
<td>RYAN PAMELA</td>
<td>399.20 914 12/12/49 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 15

Basic Examples

Introduction

This chapter illustrates the use of CA-Easytrieve/Plus to solve a variety of basic data processing problems. The emphasis is placed on reading data files and printing reports.

The input data for these examples are the Inventory and Personnel sample files described in Chapter 14, “Applications.” The field definitions for the files are contained in the macros, which are discussed later in this chapter. The field definitions are not repeated for each example; refer to the original field definitions as required.

The output for each job is typically some form of report. A wide variety of reports are printed to give you an idea of what is possible using CA-Easytrieve/Plus. For some examples, the volume of output has been condensed.

The remainder of this chapter is composed of the examples. Each example is presented in the format described in Chapter 14, “Applications.”
Employees in Region 1

The Personnel Department has requested a list of all employees in Region 1. The list must include the employees' first and last names, their employee numbers, and the branches in which they work. The list and columns must be titled, and must be in readable format.

This is a simple job for CA-Easytrieve/Plus since the report formatting is done automatically. The Personnel file is read through automatic I/O. All records with a region code of 1 are selected for the report, which is defined simply with a TITLE statement and a LINE statement.

```
1 *
2  *   EXAMPLE 15.1
3  *
4 FILE   PERSNL   FB(150 1800)
5 %PERSNL
35 *
36 *
37 JOB
38   IF REGION = 1
39     PRINT
40   END-IF
41 *
42 REPORT   LINESIZE 70
43   TITLE     'EMPLOYEES IN REGION 1'
44   LINE      NAME-FIRST  NAME-LAST  EMP#  BRANCH

------------------------------------------------------------------
11/10/83          EMPLOYEES IN REGION 1            PAGE      1
FIRST       LAST     EMPLOYEE    BRANCH
NAME       NAME      NUMBER     
GLORIA     WIMN        12267       01
NANCY      BERG        11473       02
GEORGE     CORNING     02688       03
MARY       NAGLE       00170       02
LINDA      ARNOLD      01963       04
VIRGINIA   MANHART     11602       03
ELAINE     TALL        11931       04
LYDIA      BRANDOW     02290       01
RODNEY     LARSON      11357       04
JULIE      BYER        11467       04
```
Proposed Salary Schedules

The Personnel Department has requested an evaluation of a proposed raise for the employees of Region 4. Employees with a job category of 10 are to be given a 7 percent raise; all others are to receive a 9 percent raise. Two reports are to be generated:

- A list of employees by branch, ordered by decreasing new salary, and totaled by branch and region.
- A summary breakdown by job category within branch.

Region 4 employees are actually selected by rejecting all records with a region code other than 4. The raise percentage value is set based on the job category. The raise amount (in dollars), and the new gross salary are calculated for each selected employee.

Finally, the two desired reports are generated. In DETAIL-BY-BRANCH, notice the descending sort on PAY-GROSS.

**Note:** Also, the use of the BEFORE-BREAK procedure is necessary for calculating the total raise percent for the region and for the branch. This is a very powerful facility and is used in many of the examples.

The SUMMARY-BY-CATEGORY is a straightforward summary report.

**Note:** The sequence of each report is independent. This enables a wide variety of reports to be generated with a single pass of the input file.

```
1 *   EXAMPLE 15.2
2 *
3 *   FILE    PERSNL    FB(150 1800)
4 *   %PERSNL
5      RAISE-PERCENT  W  3 P 2     HEADING('RAISE' '(PERCENT)')
6      RAISE-DOLLARS  W  4 P 2     HEADING('RAISE' '(DOLLARS)')
7      NEW-SALARY     W  4 P 2     HEADING('PROPOSED' 'SALARY')
8 *
9   JOB
10   IF  REGION NQ 4                     . * REJECT UNDESIRED RECORDS
11     GOTO JOB
12 END-IF
13   IF  JOB-CATEGORY = 10               . * SET RAISE AMT BASED ON
14     RAISE-PERCENT = 7.00              . * JOB-CATEGORY
15   ELSE
16     RAISE-PERCENT = 9.00
17   END-IF
18 *
19   CALCULATE RAISE IN DOLLARS AND NEW GROSS PAY
20      RAISE-DOLLARS = RAISE-PERCENT * PAY-GROSS / 100 + .005
21      NEW-SALARY = PAY-GROSS + RAISE-DOLLARS
22   PRINT  DETAIL-BY-BRANCH             . * PRINT DESIRED REPORTS
23   PRINT  SUMMARY-BY-CATEGORY
24 *
25   REPORT  DETAIL-BY-BRANCH               LINESIZE 78
26   SEQUENCE  BRANCH PAY-GROSS D
27   CONTROL  BRANCH
```
### Proposed Salary Schedules

**Detail by Branch -- Descending Pay-Gross**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Name</th>
<th>Gross Pay</th>
<th>Raise (Dollars)</th>
<th>Raise (Percent)</th>
<th>Proposed Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Talus</td>
<td>460.80</td>
<td>41.47</td>
<td>9.00</td>
<td>502.27</td>
</tr>
<tr>
<td></td>
<td>Ryan</td>
<td>399.20</td>
<td>27.94</td>
<td>7.00</td>
<td>427.14</td>
</tr>
<tr>
<td></td>
<td>Ward</td>
<td>183.75</td>
<td>12.86</td>
<td>7.00</td>
<td>196.61</td>
</tr>
<tr>
<td>01</td>
<td>Johnson</td>
<td>1,043.75</td>
<td>82.27</td>
<td>7.67</td>
<td>1,126.02</td>
</tr>
<tr>
<td>02</td>
<td>Hafer</td>
<td>712.80</td>
<td>49.90</td>
<td>7.00</td>
<td>762.70</td>
</tr>
<tr>
<td>02</td>
<td>Hafner</td>
<td>121.95</td>
<td>10.98</td>
<td>9.00</td>
<td>132.93</td>
</tr>
<tr>
<td></td>
<td></td>
<td>834.75</td>
<td>60.88</td>
<td>8.00</td>
<td>895.63</td>
</tr>
<tr>
<td>03</td>
<td>Jones</td>
<td>804.80</td>
<td>72.43</td>
<td>9.00</td>
<td>877.23</td>
</tr>
<tr>
<td></td>
<td>Walters</td>
<td>424.00</td>
<td>29.68</td>
<td>7.00</td>
<td>453.68</td>
</tr>
<tr>
<td></td>
<td>Zoltan</td>
<td>125.00</td>
<td>11.25</td>
<td>9.00</td>
<td>136.25</td>
</tr>
<tr>
<td>03</td>
<td></td>
<td>1,353.80</td>
<td>113.36</td>
<td>8.33</td>
<td>1,467.16</td>
</tr>
<tr>
<td>04</td>
<td>Judar</td>
<td>591.20</td>
<td>53.21</td>
<td>9.00</td>
<td>644.41</td>
</tr>
<tr>
<td>04</td>
<td></td>
<td>591.20</td>
<td>53.21</td>
<td>9.00</td>
<td>644.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,823.50</td>
<td>309.72</td>
<td>8.11</td>
<td>4,133.22</td>
</tr>
</tbody>
</table>

**Summary by Job-Category and Branch**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Job Category</th>
<th>Gross Pay</th>
<th>Proposed Salary</th>
<th>Raise (Dollars)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>10</td>
<td>582.95</td>
<td>623.75</td>
<td>40.80</td>
</tr>
<tr>
<td>01</td>
<td>25</td>
<td>460.80</td>
<td>502.27</td>
<td>41.47</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>1,043.75</td>
<td>1,126.02</td>
<td>82.27</td>
</tr>
<tr>
<td>02</td>
<td>10</td>
<td>712.80</td>
<td>762.70</td>
<td>49.90</td>
</tr>
<tr>
<td>02</td>
<td>60</td>
<td>121.95</td>
<td>132.93</td>
<td>10.98</td>
</tr>
<tr>
<td>02</td>
<td></td>
<td>834.75</td>
<td>895.63</td>
<td>68.88</td>
</tr>
<tr>
<td>03</td>
<td>10</td>
<td>424.00</td>
<td>453.68</td>
<td>29.68</td>
</tr>
<tr>
<td>03</td>
<td>40</td>
<td>929.80</td>
<td>1,013.48</td>
<td>83.68</td>
</tr>
<tr>
<td>03</td>
<td></td>
<td>1,353.80</td>
<td>1,467.16</td>
<td>113.36</td>
</tr>
<tr>
<td>04</td>
<td>25</td>
<td>591.20</td>
<td>644.41</td>
<td>53.21</td>
</tr>
<tr>
<td>04</td>
<td></td>
<td>591.20</td>
<td>644.41</td>
<td>53.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3,823.50</td>
<td>4,133.22</td>
<td>309.72</td>
</tr>
</tbody>
</table>
Employee Letters

The Personnel Department has decided to accept the proposed salary adjustments and wants to generate letters to all employees, informing them of the salary adjustment. In addition to the letter, a mailing label must be generated. The letters and mailing labels should be ordered by Zip code to minimize mailing costs.

This is the same basic job as the previous example, but the output is different. Instead of a standard report, a letter is generated.

**Note:** The ease with which the letter is specified.

By including the parameters SKIP 1 and PAGESIZE 40, we insure only one letter per page.

The mailing labels are generated by specifying their content. The ACROSS 2 parameter enables the labels to fit on the page of this document - ACROSS 4 is normal for most label runs.

The letters could be generated two-on-a-page, if desired, by replacing PAGESIZE 40 with LABELS (ACROSS 2 DOWN 40). Labels are simply a special type of report.

```plaintext
1 * 2 *   EXAMPLE 15.3
3 * 4 FILE    PERSNL    FB(150 1800)
5 %PERSNL
35 OLD-SALARY PAY-GROSS PAY-GROSS   MASK($$$$$.99')
36 RAISE-PERCENT W 3 P 2      HEADING('RAISE' '(PERCENT)')
37 RAISE-DOLLARS W 4 P 2       HEADING('RAISE' '(DOLLARS)') -
38 NEW-SALARY     W 4 P 2      HEADING('PROPOSED' 'SALARY') -
39 * 40 * 41 JOB
42 IF  REGION NE 4                     . * REJECT UNDESIRED RECORDS
44   GOTO JOB
45 END-IF
46 IF  JOB-CATEGORY = 10               . * SET RAISE AMT BASED ON
48   RAISE-PERCENT = 7.00              . * JOB-CATEGORY
50 ELSE
51   RAISE-PERCENT = 9.00
52 END-ELSE
53 *             CALCULATE RAISE IN DOLLARS AND NEW GROSS PAY
54 RAISE-DOLLARS = RAISE-PERCENT * OLD-SALARY / 100 + .005
55 NEW-SALARY = PAY-GROSS + RAISE-DOLLARS
56 PRINT  EMPLOYEE-LETTER              . * PRINT LETTER AND
58 PRINT  MAILING-LABEL                . * MAILING LABEL
60 *
61 REPORT  EMPLOYEE-LETTER   LINESIZE 78   -
63 LINE  1 COL 1 'ABC SYSTEMS, INC.' COL 60 SYSDATE
64 LINE  3 NAME-FIRST NAME-LAST
65 LINE  4 ADDR-STREET
```
IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 7.00 % OF YOUR GROSS SALARY OF $424.00. THIS EQUATES TO $29.68, OR A NEW GROSS SALARY OF $453.68. THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT
ABC SYSTEMS, INC.                                          11/11/83

ALFRED JONES
2070 BELMONT ROAD NW
LOS ANGELES  CA 90052

DEAR ALFRED

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 % OF YOUR GROSS SALARY OF $804.80 . THIS EQUATES TO $72.43 , OR A NEW GROSS SALARY OF $877.23 .

THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

ABC SYSTEMS, INC.                                          11/11/83

JANET ZOLTAN
2026 FORT DAVIS ST S
LOS ANGELES  CA 90091

DEAR JANET

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 % OF YOUR GROSS SALARY OF $125.00 . THIS EQUATES TO $11.25 , OR A NEW GROSS SALARY OF $136.25 .

THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT
ABC SYSTEMS, INC.  

ARTHUR HAER
806 CONNECTICUT AVE
SAN DIEGO CA 92031

DEAR ARTHUR

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 % OF YOUR GROSS SALARY OF $121.95. THIS EQUATES TO $10.98, OR A NEW GROSS SALARY OF $132.93.

THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

ABC SYSTEMS, INC.  

LISA JOHNSON
806 CONNECTICUT AVE
SAN DIEGO CA 92045

DEAR LISA

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 7.00 % OF YOUR GROSS SALARY OF $712.80. THIS EQUATES TO $49.90, OR A NEW GROSS SALARY OF $762.70.

THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT
ABC SYSTEMS, INC.                                          11/11/83

PAULA    JUDAR
4333 46TH ST N W
SAN FRANCISC CA 94041

DEAR PAULA

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 % OF YOUR GROSS SALARY OF $591.20. THIS EQUATES TO $53.21, OR A NEW GROSS SALARY OF $644.41. THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

---

ABC SYSTEMS, INC.                                          11/11/83

RUTH     TALUS
9331 CAROLINE AVE
SEATTLE      WA 98083

DEAR RUTH

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS PROVIDING YOU A SALARY INCREASE EFFECTIVE ON YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 % OF YOUR GROSS SALARY OF $460.80. THIS EQUATES TO $41.47, OR A NEW GROSS SALARY OF $502.27. THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT
ABC SYSTEMS, INC.                                          11/11/83

PAMELA   RYAN
1717 R NW #301
SEATTLE      WA 98009

DEAR PAMELA

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS
PROVIDING YOU A SALARY INCREASE EFFECTIVE ON
YOUR NEXT PAY CHECK.  THE INCREASE REFLECTS YOUR
EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE
FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 7.00 %
OF YOUR GROSS SALARY OF $399.20 .  THIS EQUATES
TO $27.94 , OR A NEW GROSS SALARY OF $427.14 .
THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES
YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

11/11/83

ABC SYSTEMS, INC.                                          11/11/83

MARINA   WARD
1725 H ST NE APT 2
SEATTLE      WA 98015

DEAR MARINA

IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS
PROVIDING YOU A SALARY INCREASE EFFECTIVE ON
YOUR NEXT PAY CHECK.  THE INCREASE REFLECTS YOUR
EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE
FIELD OF FINANCIAL COMPUTER SYSTEMS.

IN YOUR PARTICULAR CASE THE INCREASE IS 7.00 %
OF YOUR GROSS SALARY OF $183.75 .  THIS EQUATES
TO $12.86 , OR A NEW GROSS SALARY OF $196.61 .
THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES
YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

KAREN      WALTERS            ALFRED     JONES
1022 5 KENSINGTON PK          2070 BELMONT ROAD NW
LOS ANGELES    CA   90030     LOS ANGELES    CA   90052
Mailing Labels

The Personnel Department has requested a mailing label run for all employees in Regions 1 and 2. These labels should be ordered by Zip code, with a break on Zip code prefix (first three digits), in order to receive a lower postage rate.

Selecting the desired employee records to be passed to the report processor for formatting into labels is simple. More complex is the control break when the Zip code prefix changes.

Note: The redefinition of the Zip code field enables sorting on the first three digits. After a break occurs, the next label begins on a new line. Additional spacing can be obtained by a BEFORE-BREAK procedure which issues a DISPLAY SKIP 6 statement.

```
1 *
2 *   EXAMPLE 15.4
3 *
4 FILE    PERSNL    FB(150 1800)
5 %PERSNL
35 ZIP-PREFIX  ADDR-ZIP  3  N          * REDEFINE FIRST 3 DIGITS OF ZIP
37 *
38 *
39 JOB
40   IF  REGION EQ 1 2                 * SELECT DESIRED RECORDS
42     PRINT  MAILING-LABEL            * PRINT MAILING LABEL
44   END-IF
45 *
46 REPORT  MAILING-LABEL    LABELS (ACROSS 3  SIZE 28)  SPACE 1
47   SEQUENCE  ADDR-ZIP                * SORT ON ZIP CODE
49   CONTROL  ZIP-PREFIX              * BREAK ON ZIP PREFIX
51   LINE  1  EMP#  REGION  BRANCH
52   LINE  3  NAME-FIRST  NAME-LAST
53   LINE  4  ADDR-STREET
54   LINE  5  ADDR-CITY  ADDR-STATE  ADDR-ZIP
```
12267 1 01 02200 1 01
GLORIA WIMN LYDIA BRANDOW
430 M ST SW 107 3616 B ST S E
BOSTON MA 02005 BOSTON MA 02011
11602 1 03 02688 1 03
VIRGINIA MANHART GEORGE CORNING
1305 POTOMAC ST N W 3208 S 5TH
TRENTON NJ 08621 TRENTON NJ 08635
01963 1 04 11357 1 04 11467 1 04
LINDA ARNOLD RODNEY LARSON JULIE BYER
1569 COLONIAL TERR A 610 H ST SW 3400 NORTH 18TH STRE
NEW YORK NY 10012 NEW YORK NY 10059 NEW YORK NY 10071
11931 1 04
ELAINE TALL
1412 36TH ST NW
NEW YORK NY 10091
02765 2 03 04234 2 03 03416 2 03
RALPH DENNING BARBARA MCMANON BILL FORREST
1629 16TH ST NW APT 1318 24TH STREET S 1545 18TH ST NW
WASHINGTON DC 20005 WASHINGTON DC 20015 WASHINGTON DC 20018
00577 2 03
KATHY PETRIK
5085 BENTON AVE
WASHINGTON DC 20032
00370 1 02 11473 1 02
MARY NAGLE NANCY BERG
826 D STREET SE 3710 JENIFER ST N W
BALTIMORE MD 21034 BALTIMORE MD 21055
04225 2 05 01895 2 05
NED LOYAL DENISE VETTER
17 KENNEDY STREET 7311 KEYSTONE LANE 4
RALEIGH NC 27516 RALEIGH NC 27591
11710 2 02 03571 2 02
CAROL POWELL MAX KRUSE
5023 AMES STREET N E 2161 N PIERCE STREET
ATLANTA GA 30316 ATLANTA GA 30345
00445 2 04
JEAN POST
1250 4TH ST SW
JACKSONVILLE FL 32052
11376 2 01
PATTI HUSS
1355 TWEKESBURY PLAC
CLEARWATER FL 33512
Tally Reports

The Personnel Department wants tallies on various fields within the personnel file. Each tally report lists the number of employees in the specified category and the percent of the total employees that number represents. The desired categories are:

- Sex
- Marital status
- Job category
- Salary code
- Gross pay in $100 increments
- City.

This job generates five separate summary reports; the first two categories are combined in the first report. The report process does most of the work. All that is done explicitly is the percent calculation in the BEFORE-BREAK procedure.

If the illustration of the coding seems overwhelming to read, follow one report at a time (the way CA-Easytrieve/Plus processes the code). The report data is collected in work files, usually one for each report. After the input file is read, the output for each report is formatted serially. There are some exceptions to this flow, but it is the norm.

As you are reading the code, notice the use of W and S fields, and how rounding is performed in the percent calculations.

Note: Also, generating a number of reports from a single pass of the file dramatically reduces the resources required without increasing the complexity of the job.

```plaintext
1 * 2 * EXAMPLE 15.5 3 * 4 FILE PERSNL FB(150 1800) 5 %PERSNL 35 SEX-CODE W 6 A HEADING 'SEX' 36 GROSS-RANGE W 3 P HEADING ('SALARY RANGE' 'HUNDRED $ INCR') 37 TOTAL-EMPLOYEES S 3 P 0 38 PERCENT W 3 P 2 HEADING('PERCENT' 'OF' 'TOTAL') 39 * 40 * 41 JOB 42 TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1 43 * 44 IF SEX EQ 1 . * SET PROPER SEX CODE 45 SEX-CODE = 'FEMALE' 46 ELSE 47 SEX-CODE = 'MALE' 48 END-IF 49 * 50 PRINT SEX-MARITAL-STAT-RPT . * PRINT REPORT 51 PRINT JOB-CATEGORY-RPT . * PRINT REPORT
```
55 *
56 PRINT SALARY-CODE-RPT . * PRINT REPORT
58 *
59 GROSS-RANGE = PAY-GROSS / 100.00 . * CALCULATE GROSS SALARY
60 GROSS-RANGE = GROSS-RANGE * 100 . * RANGE
61 PRINT GROSS-PAY-RPT . * PRINT THE REPORT
62 *
63 PRINT CITY-RPT . * PRINT THE CITY REPORT
65 *
66 REPORT SEX-MARITAL-STAT-RPT SUMMARY LINESIZE 78
67 SEQUENCE SEX-CODE MARITAL-STAT . * SORT REPORT
68 CONTROL SEX-CODE MARITAL-STAT . * BREAK SPECIFICATION
69 TITLE 1 'TALLY OF EMPLOYEES BY SEX AND MARITAL STATUS'
70 LINE 1 SEX-CODE MARITAL-STAT TALLY PERCENT
71 BEFORE-BREAK. PROC . * CALCULATE PERCENT
72 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
73 END-PROC
75 *
76 REPORT JOB-CATEGORY-RPT SUMMARY LINESIZE 78
77 SEQUENCE JOB-CATEGORY . * SORT REPORT
78 CONTROL JOB-CATEGORY . * BREAK SPECIFICATION
79 TITLE 1 'TALLY OF EMPLOYEES BY JOB CATEGORY'
80 LINE 1 JOB-CATEGORY TALLY PERCENT
81 BEFORE-BREAK. PROC . * CALCULATE PERCENT
82 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
84 END-PROC
86 *
87 REPORT SALARY-CODE-RPT SUMMARY LINESIZE 78
88 SEQUENCE SALARY-CODE . * SORT REPORT
89 CONTROL SALARY-CODE . * BREAK SPECIFICATION
90 TITLE 1 'TALLY OF EMPLOYEES BY SALARY CODE'
91 LINE 1 SALARY-CODE TALLY PERCENT
92 BEFORE-BREAK. PROC . * CALCULATE PERCENT
93 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
95 END-PROC
97 *
98 REPORT GROSS-PAY-RPT SUMMARY LINESIZE 78
99 SEQUENCE GROSS-RANGE D . * SORT REPORT
100 CONTROL GROSS-RANGE . * BREAK SPECIFICATION
101 TITLE 1 'TALLY OF EMPLOYEES BY GROSS SALARY RANGE'
102 HEADING PAY-GROSS ('AVERAGE' 'GROSS' 'SALARY')
103 LINE 1 GROSS-RANGE TALLY PERCENT PAY-GROSS
104 BEFORE-BREAK. PROC . * CALCULATE PERCENT
105 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
106 PAY-GROSS = PAY-GROSS / TALLY + .005
108 END-PROC
110 *
111 REPORT CITY-RPT SUMMARY LINESIZE 78
112 SEQUENCE ADDR-CITY . * SORT REPORT
113 CONTROL ADDR-CITY . * BREAK SPECIFICATION
114 TITLE 1 'TALLY OF EMPLOYEES BY HOME CITY'
115 LINE 1 ADDR-CITY TALLY PERCENT
116 BEFORE-BREAK. PROC . * CALCULATE PERCENT
117 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
119 END-PROC
### Tally of Employees by Sex and Marital Status

<table>
<thead>
<tr>
<th>SEX</th>
<th>MARITAL STATUS</th>
<th>TALLY</th>
<th>PERCENT OF TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEMALE</td>
<td>M</td>
<td>13</td>
<td>27.08</td>
</tr>
<tr>
<td>FEMALE</td>
<td>S</td>
<td>23</td>
<td>47.92</td>
</tr>
<tr>
<td>FEMALE</td>
<td></td>
<td>36</td>
<td>75.00</td>
</tr>
<tr>
<td>MALE</td>
<td>M</td>
<td>5</td>
<td>10.42</td>
</tr>
<tr>
<td>MALE</td>
<td>S</td>
<td>7</td>
<td>14.58</td>
</tr>
<tr>
<td>MALE</td>
<td></td>
<td>12</td>
<td>25.00</td>
</tr>
</tbody>
</table>

Total: 48 employees (100.00%)

### Tally of Employees by Job Category

<table>
<thead>
<tr>
<th>JOB CATEGORY</th>
<th>TALLY</th>
<th>PERCENT OF TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>22</td>
<td>45.83</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>25</td>
<td>8</td>
<td>16.67</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>40</td>
<td>6</td>
<td>12.50</td>
</tr>
<tr>
<td>60</td>
<td>7</td>
<td>14.58</td>
</tr>
</tbody>
</table>

Total: 48 employees (100.00%)

### Tally of Employees by Salary Code

<table>
<thead>
<tr>
<th>SALARY CODE</th>
<th>TALLY</th>
<th>PERCENT OF TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>20</td>
<td>41.67</td>
</tr>
<tr>
<td>02</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>03</td>
<td>9</td>
<td>18.75</td>
</tr>
<tr>
<td>05</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>06</td>
<td>7</td>
<td>14.58</td>
</tr>
<tr>
<td>08</td>
<td>8</td>
<td>16.67</td>
</tr>
</tbody>
</table>

Total: 48 employees (100.00%)
### TALLY OF EMPLOYEES BY GROSS SALARY RANGE

<table>
<thead>
<tr>
<th>SALARY RANGE</th>
<th>PERCENT OF TOTAL</th>
<th>AVERAGE GROSS SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>2.08</td>
<td>1,004.00</td>
</tr>
<tr>
<td>800</td>
<td>4.17</td>
<td>804.72</td>
</tr>
<tr>
<td>700</td>
<td>6.25</td>
<td>736.00</td>
</tr>
<tr>
<td>600</td>
<td>2.08</td>
<td>628.00</td>
</tr>
<tr>
<td>500</td>
<td>4.17</td>
<td>572.80</td>
</tr>
<tr>
<td>400</td>
<td>8.33</td>
<td>455.64</td>
</tr>
<tr>
<td>300</td>
<td>35.42</td>
<td>348.19</td>
</tr>
<tr>
<td>200</td>
<td>20.83</td>
<td>264.29</td>
</tr>
<tr>
<td>100</td>
<td>14.58</td>
<td>154.92</td>
</tr>
<tr>
<td></td>
<td>2.08</td>
<td>13.80</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### TALLY OF EMPLOYEES BY HOME CITY

<table>
<thead>
<tr>
<th>CITY</th>
<th>TALLY</th>
<th>PERCENT OF TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATLANTA</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>BALTIMORE</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>BOSTON</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>CHICAGO</td>
<td>6</td>
<td>12.50</td>
</tr>
<tr>
<td>CLEARWATER</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>DALLAS</td>
<td>6</td>
<td>12.50</td>
</tr>
<tr>
<td>JACKSONVILLE</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>KANSAS CITY</td>
<td>4</td>
<td>8.33</td>
</tr>
<tr>
<td>LOS ANGELES</td>
<td>3</td>
<td>6.25</td>
</tr>
<tr>
<td>MINNEAPOLIS</td>
<td>3</td>
<td>6.25</td>
</tr>
<tr>
<td>NEW YORK</td>
<td>4</td>
<td>8.33</td>
</tr>
<tr>
<td>RALEIGH</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>SAN DIEGO</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>SAN FRANCISC</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>SEATTLE</td>
<td>3</td>
<td>6.25</td>
</tr>
<tr>
<td>TRENTON</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>WASHINGTON</td>
<td>4</td>
<td>8.33</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>100.00</td>
</tr>
</tbody>
</table>
The National Federation of Business and Professional Women's Clubs is recruiting for a chapter in the Chicago area. They have requested a list of all female employees in the Chicago branch, along with their phone numbers.

This example is a simple process of selecting records based on the value in two fields, ADDR and SEX, then sequencing the report by name.

```
1 *  
2 *   EXAMPLE 15.6  
3 *  
4 FILE PERSNL  FB(150 1800)  
5 %PERSNL  
35 *  
36 *  
37 JOB                                 . * SELECT DESIRED RECORDS  
39 IF ADDR-CITY EQ 'CHICAGO' AND SEX = 1  
40 PRINT PHONE-LIST               . * PRINT PHONE LIST  
42 END-IF  
43 *  
44 REPORT PHONE-LIST         LINESIZE 78  
45   SEQUENCE  NAME-LAST NAME-FIRST    . * SORT ON NAME  
47   TITLE 1 'CHICAGO AREA WOMEN AND TELEPHONE NUMBERS'  
48   LINE 1 NAME-FIRST NAME-LAST TELEPHONE
```

<table>
<thead>
<tr>
<th>FIRST</th>
<th>LAST</th>
<th>TELEPHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>LINDA</td>
<td>EPERT</td>
<td>(312) 588-5118</td>
</tr>
<tr>
<td>JOAN</td>
<td>MILLER</td>
<td>(312) 588-6701</td>
</tr>
<tr>
<td>DEBBIE</td>
<td>NORIDGE</td>
<td>(312) 646-0934</td>
</tr>
<tr>
<td>PAT</td>
<td>ROGERS</td>
<td>(312) 646-1650</td>
</tr>
<tr>
<td>ANN</td>
<td>STRIDE</td>
<td>(312) 646-1891</td>
</tr>
</tbody>
</table>
Salary Tally Report

The Personnel Department has requested that the Salary Range Report, produced in Example 15.5, be expanded to include a bar graph of tally percent. The bar graph is generated using the MOVE statement within the BEFORE-BREAK procedure. For each two percentage points, an asterisk is plotted. If the percentage exceeds 60 percent, spaces are printed.

As illustrated in this and several of the previous examples, the BEFORE-BREAK procedure is invaluable. It permits us to modify the contents of a summary line prior to printing (a common requirement in many control reports).

```plaintext
1 * 
2 * EXAMPLE 15.7 
3 * 
4 FILE PERSNL FB(150 1800) 
5 %PERSNL 
35 GROSS-RANGE W 3 P HEADING ('SALARY RANGE' 'HUNDRED $ INCR') 
36 TOTAL-EMPLOYEES S 3 P 0 
37 PERCENT W 3 P 2 HEADING ('PERCENT' 'OF' 'TOTAL') 
38 BAR-GRAPH S 30 A HEADING ('PERCENT OF EMPLOYEES' - 'EACHASTERISK EQUALS 2%') 
39 ASTERISKS S 30 A VALUE ('******************************') 
40 ILTH S 2 P 
41 * 
42 * 
43 JOB 
44 TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1 
45 * 
46 GROSS-RANGE = PAY-GROSS / 100.00 . * CALCULATE GROSS SALARY 
48 GROSS-RANGE = GROSS-RANGE * 100 . * RANGE 
50 PRINT GROSS-PAY-RPT . * PRINT THE REPORT 
52 * 
53 REPORT GROSS-PAY-RPT SUMMARY SUMCTL DTLCOPY LINESIZE 78 
54 SEQUENCE GROSS-RANGE D . * SORT REPORT 
56 CONTROL GROSS-RANGE . * BREAK SPECIFICATION 
58 TITLE 1 'TALLY OF EMPLOYEES BY GROSS SALARY RANGE' 
59 LINE 1 GROSS-RANGE TALLY PERCENT BAR-GRAPH 
60 BEFORE-BREAK PROC . * CALCULATE PERCENT 
63 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005 
64 ILTH = ( PERCENT + 1 ) / 2 
65 IF ILTH LE 30 
66 MOVE ASTERISKS ILTH TO BAR-GRAPH 
67 ELSE 
68 MOVE SPACES TO BAR-GRAPH 
69 END-IF 
70 END-PROC 
```
The Personnel master file record has run out of room, and it is necessary to expand and reformat it.

Following is a diagram describing the operation:

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Data-1          | Data-2          | Data-3          | Data-4          | Data-5          |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 1               | 51              | 71              | 121             | 141             | 150             |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| Data-1 New-1    | Data-2 New-3    | Data-3 New-5    | Data-4 New-7    | Data-5 New-9    |
+-----------------+-----------------+-----------------+-----------------+-----------------+
| 1               | 51              | 81              | 96              | 146             |
|                 |                 |                 |                 | 156             |
|                 |                 |                 |                 | 179             |
|                 |                 |                 |                 | 181             |
|                 |                 |                 |                 | 200             |
```
The key to this example is the proper definition of the fields within each file. By using the same name for the corresponding fields in each file, one MOVE LIKE statement performs all five data moves. The MOVE statement initializes the new fields.

1 *
2 *     EXAMPLE 15.8
3 *
4 FILE  PERSIN                       . * INPUT FILE
5   DATA-1        1  50 A
6   DATA-2       51  20 A
7   DATA-3       71  50 A
8   DATA-4      121  20 A
9   DATA-5      141  10 A
10 *
11 FILE  PERSOUT FB(200 3600)         . * REFORMATTED OUTPUT FILE
12   DATA-1        1  50 A
13   NEW-1        51  10 N 0
14   DATA-2       61  20 A
15   NEW-2        81   4 P
16   NEW-3        85   5 P
17   NEW-4       98   6 N 0
18   DATA-3       96  50 A
19   NEW-5      146  10 A
20   DATA-5      156  10 A
21   NEW-6      166  13 A
22   NEW-7      179   2 B
23   DATA-4      181  20 A
24 *
25 *
26 JOB   FINISH  WRAP-UP
27   MOVE  LIKE  PERSIN TO PERSOUT    . * MOVE LIKE NAMED FIELDS
28                                      *   FROM PERSIN TO PERSOUT
29   MOVE  ZERO  TO  NEW-1 NEW-2 NEW-3 NEW-4 NEW-7
30                                      * INITIALIZE NUMERIC FIELDS
31   MOVE  SPACE TO  NEW-5 NEW-6      . * INITIALIZE ALPHA FIELDS
32   PUT  PERSOUT                     . * OUTPUT THE REFORMATTED FILE
33 *
34 WRAP-UP.  PROC
35     DISPLAY NEWPAGE 'TOTAL INPUT RECORDS = ' RECORD-COUNT(PERSIN)
36     DISPLAY SKIP 2 'TOTAL OUTPUT RECORDS = ' RECORD-COUNT(PERSOUT)
37 END-PROC

-----------------------------------------------------------------------------
TOTAL INPUT RECORDS =            48
TOTAL OUTPUT RECORDS =            48
-----------------------------------------------------------------------------
Average Regional Gross Salary

The region codes of the personnel file represent regions of the United States. In most cases it is more desirable to output a text description of the region than to print the code. The conversion is performed by the CA-Easytrieve/Plus table handling facility.

In this example, the Personnel Department has requested a report of average gross salaries for each region. CA-Easytrieve/Plus reads the input records and calculates totals for the number of employees and the gross salaries. The SEARCH statement obtains the text description of the region code, and the information is output on a report.

Note: The SEQUENCE statement specifies the region code while the CONTROL break is based on REGION-TEXT. This enables the report to be ordered on region code while still printing the region text.

Also, most of the printed values are generated in the BEFORE-BREAK procedure. The order of the first two statements in that procedure is mandatory because the second statement modifies the AVERAGE-GROSS.

```plaintext
1 * 
2 * EXAMPLE 15.9 
3 * 
4 FILE PERSNL FB(150 1800) 
5 %PERSNL 
 35 AVERAGE-GROSS W 4 P 2 HEADING ('AVERAGE' 'GROSS' 'SALARY') 
 36 TOTAL-GROSS S 6 P 2 
 37 PERCENT-GROSS W 3 P 2 HEADING ('PERCENT OF' 'COMPANY' 'GROSS') 
 38 PERCENT-TALLY W 3 P 2 - 
       HEADING ('PERCENT OF' 'COMPANY' 'EMPLOYEES') 
 39 SALARY-RATIO W 3 P 3 HEADING ('RATIO OF' '%-GROSS' '/' '%-TALLY') 
 40 TOTAL-EMPLOYEES S 3 P 0 
 41 REGION-TEXT W 10 A 
      HEADING ('COMPANY' 'REGION') 
 42 * 
 43 FILE RGNID TABLE INSTREAM . * DEFINE INSTREAM REGION TABLE 
 45 ARG 1 1 N. DESC 3 10 A. * DEFINE TABLE SPECIAL FIELD IDS 
 48 1 NORTHEAST 
 2 SOUTHEAST 
 3 CENTRAL 
 4 WEST 
 ENDTABLE 
 49 * 
 50 * 
 51 JOB 
 52 TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1 . * CALCULATE TOTAL EMPLOYEES 
 54 AVERAGE-GROSS = PAY-GROSS . * AVERAGE = GROSS FOR EACH RECD 
 56 TOTAL-GROSS = TOTAL-GROSS + PAY-GROSS 
 57 * CALCULATE TOTAL GROSS FOR COMP 
 58 * SEARCH TABLE FOR MATCHING REGION INFORMATION 
 59 SEARCH RGNID WITH REGION GIVING REGION-TEXT 
 60 * 
 61 PRINT AVG-SALARY-RPT . * PRINT THE REPORT 
 63 * 
 64 REPORT AVG-SALARY-RPT SUMMARY LINESIZE 78 
 65 SEQUENCE REGION . * SORT REPORT 
 67 CONTROL REGION-TEXT . * BREAK SPECIFICATION 
 69 TITLE 1 'AVERAGE GROSS SALARY BY REGION' 
 70 HEADING TALLY ('NUMBER' 'OF' 'EMPLOYEES')
```
Central Region Employees

The Personnel Department has requested an alphabetical list of employees in the central region. The report is to include the employees' name, social security number, department code, and department name. In addition, Personnel needs a list of the central region employees grouped by department name.

To solve this problem, we must know that each employee is assigned to a particular company department, the number of which is contained within each employee record. In addition to the number, each department has a unique department name, such as Engineering, Marketing, and so forth. A table of department numbers and the corresponding names is available in a table file named DPTCODE.

First, we select all employees in Region 3 (Central Region). For each such employee, we search the DPTCODE table for the corresponding department name. If no entry is found, we insert a dummy department name (*NO TABLE ENTRY) and issue a PRINT to an error report. Regardless whether a department name is found, we issue a PRINT statement to both the ALPHA-LIST and the RPT-BY-DEPT reports.

ALPHA-LIST is a simple list, sequenced by name.

The RPT-BY-DEPT is a control report with breaks on DEPT.

Note: The use of the HEADING statement supplies alternate report headings for the specified fields; this is the only way to change the heading for TALLY.
Also, printing is suppressed for the summary line in the MISSING-DEPT-CODE report.

1 *
2 *   EXAMPLE 15.10
3 *
4 FILE   PERSNL   FB(150 1800)
5 %PERSNL
35 *
36 DEPT-NAME   W 15  A   HEADING ('DEPARTMENT' 'NAME')
37 *
38 FILE   DPTCODE   TABLE      . * TABLE FILE DEFINITION
40   ARG  1 3 N.   DESC  5 15 A
42 *
43 *
44 JOB
45   IF  REGION NE 3                     . * SELECT REGION 3 EMPLOYEES
47     GO TO JOB                         . * SKIP ALL OTHERS
49   END-IF
50   SEARCH DPTCODE  WITH DEPT  GIVING DEPT-NAME
51                                         * GET DEPT NAME FROM TABLE
52 IF  NOT DPTCODE                     . * IF NO DEPT NAME PRESENT
54     DEPT-NAME = '*NO TABLE ENTRY'     . * INDICATE MISSING ENTRY
56   PRINT  MISSING-DEPT-CODE          . * OUTPUT ERROR REPORT
58 END-IF
59   PRINT  ALPHA-LIST                   . * PRINT ALPHA LISTING
60   PRINT  RPT-BY-DEPT                  . * OUTPUT REPORT BY DEPARTMENT
63 *
64 REPORT   ALPHA-LIST                  LINESIZE 78
65   SEQUENCE  NAME-LAST NAME-FIRST
67   LINE   NAME-LAST NAME-FIRST   SSN DEPT DEPT-NAME
68 *
69 REPORT   RPT-BY-DEPT   SUMCTL NONE  LINESIZE 78
70   SEQUENCE  DEPT-NAME NAME-LAST       . * SEQUENCE BY DEPT AND NAME
72   CONTROL  DEPT-NAME                 . * CONTROL BREAK ON DEPT
74   TITLE     'CENTRAL REGION EMPLOYEES BY DEPARTMENT'
76   HEADING   TALLY    ('NUMBER' 'OF' 'EMPLOYEES')
78   LINE   DEPT-NAME BRANCH NAME-LAST NAME-FIRST  TALLY
80 *
81 REPORT   MISSING-DEPT-CODE  SUMMARY  LINESIZE 78
83   SEQUENCE  DEPT                     . * SEQUENCE BY DEPT
85   CONTROL  FINAL NOPRINT DEPT       . * CONTROL BREAK ON DEPT
88   TITLE     'CENTRAL REGION MISSING DEPARTMENT DESCRIPTIONS'
91   HEADING   TALLY    ('NUMBER' 'OF' 'EMPLOYEES')
94   LINE   DEPT TALLY
<table>
<thead>
<tr>
<th>LAST</th>
<th>FIRST</th>
<th>SECURITY</th>
<th>DEPT</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>CROCI</td>
<td>JUDY</td>
<td>577-16-2985</td>
<td>914</td>
<td>ENGINEERING</td>
</tr>
<tr>
<td>EPERT</td>
<td>LINDA</td>
<td>228-46-5157</td>
<td>918</td>
<td>DATA PROCESSING</td>
</tr>
<tr>
<td>GRECO</td>
<td>LESLIE</td>
<td>388-18-6119</td>
<td>914</td>
<td>ENGINEERING</td>
</tr>
<tr>
<td>GREEN</td>
<td>BRENSA</td>
<td>288-28-2315</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>ISAAC</td>
<td>RUTH</td>
<td>418-46-1872</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>KELLY</td>
<td>KEITH</td>
<td>231-68-9995</td>
<td>940</td>
<td>PRINTING</td>
</tr>
<tr>
<td>LACH</td>
<td>LORRIE</td>
<td>215-36-5852</td>
<td>923</td>
<td>MAILROOM</td>
</tr>
<tr>
<td>MALLOW</td>
<td>TERRY</td>
<td>477-44-4948</td>
<td>942</td>
<td>*NO TABLE ENTRY</td>
</tr>
<tr>
<td>MILLER</td>
<td>JOAN</td>
<td>148-32-0779</td>
<td>920</td>
<td>RECEIVING</td>
</tr>
<tr>
<td>NORIDGE</td>
<td>DEBBIE</td>
<td>578-54-3178</td>
<td>944</td>
<td>*NO TABLE ENTRY</td>
</tr>
<tr>
<td>PHILPS</td>
<td>SUE</td>
<td>090-22-9192</td>
<td>940</td>
<td>PRINTING</td>
</tr>
<tr>
<td>REYNOLDS</td>
<td>WILLIAM</td>
<td>061-30-8680</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>ROGERS</td>
<td>PAT</td>
<td>579-59-4170</td>
<td>924</td>
<td>PERSONNEL</td>
</tr>
<tr>
<td>SMOTH</td>
<td>CINDY</td>
<td>578-38-7587</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>STRIDE</td>
<td>ANN</td>
<td>467-56-4149</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>THOMPSON</td>
<td>JANICE</td>
<td>577-09-1160</td>
<td>923</td>
<td>MAILROOM</td>
</tr>
<tr>
<td>WEST</td>
<td>KATHY</td>
<td>570-10-5594</td>
<td>940</td>
<td>PRINTING</td>
</tr>
<tr>
<td>YOUNG</td>
<td>JANE</td>
<td>118-34-8805</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>DEPARTMENT NAME</td>
<td>BRANCH</td>
<td>LAST NAME</td>
<td>FIRST NAME</td>
<td>NUMBER OF EMPLOYEES</td>
</tr>
<tr>
<td>----------------</td>
<td>--------</td>
<td>-----------</td>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>*NO TABLE ENTRY</td>
<td>04</td>
<td>MALLOW</td>
<td>TERRY</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>NORIDGE</td>
<td>DEBBIE</td>
<td></td>
</tr>
<tr>
<td>DATA PROCESSING</td>
<td>03</td>
<td>EPERT</td>
<td>LINDA</td>
<td>1</td>
</tr>
<tr>
<td>ENGINEERING</td>
<td>04</td>
<td>CROCI</td>
<td>JUDY</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>GRECO</td>
<td>LESLIE</td>
<td></td>
</tr>
<tr>
<td>MAILROOM</td>
<td>02</td>
<td>LACH</td>
<td>LORRIE</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>THOMPSON</td>
<td>JANICE</td>
<td></td>
</tr>
<tr>
<td>MARKETING</td>
<td>04</td>
<td>GREEN</td>
<td>BREnda</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>ISAAC</td>
<td>RUTH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>REYNOLDS</td>
<td>WILLIAM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>02</td>
<td>SMOOTH</td>
<td>CINDY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>STRIDE</td>
<td>ANN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>YOUNG</td>
<td>JANE</td>
<td></td>
</tr>
<tr>
<td>PERSONNEL</td>
<td>03</td>
<td>ROGERS</td>
<td>PAT</td>
<td>1</td>
</tr>
<tr>
<td>PRINTING</td>
<td>01</td>
<td>KELLY</td>
<td>KEITH</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>PHILPS</td>
<td>SUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>WEST</td>
<td>KATHY</td>
<td></td>
</tr>
<tr>
<td>RECEIVING</td>
<td>03</td>
<td>MILLER</td>
<td>JOAN</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>OSMON</td>
<td>SAMUEL</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MISSING DEPARTMENT CODES</th>
<th>NUMBER OF EMPLOYEES</th>
</tr>
</thead>
<tbody>
<tr>
<td>942</td>
<td>1</td>
</tr>
<tr>
<td>944</td>
<td>1</td>
</tr>
</tbody>
</table>
Inventory Report by City

An Inventory Master File is available for our use. This file contains information on a diverse inventory. The Material Procurement Department has requested an inventory report, ordered by the city in which the parts are located. Also, the groups by city need to be separated by a blank line, but no totals by city are desired.

The CA-Easytrieve/Plus job to perform this request is quite simple; all processing is performed in the report section. The NOPRINT option on the CONTROL statement is used to suppress printing the summary lines.

```
1 *
2 *   EXAMPLE 15.11
3 *
4 FILE    INVMSTR   FB(200 3000)
5 %INVMSTR
44 *
45 JOB
46   PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE
48 *
49 REPORT  INV-BY-CITY         LINESIZE 80
50   SEQUENCE  LOCATION-CITY PART-NUMBER
51   CONTROL   FINAL NOPRINT LOCATION-CITY NOPRINT
52   TITLE   1 'INVENTORY BY CITY ORDERED BY PART NUMBER'
53   LINE    1 LOCATION-CITY PART-NUMBER PART-DESCRIPTION
54   BEFORE-BREAK. PROC
56   DISPLAY                     . * ADDITIONAL SPACING BETWEEN GROUPS
58 END-PROC
```
### Expanded Inventory Report

After reviewing the previous report, the Materials Department decided they would like an expanded report which includes the quantity of each item at last inventory, the selling price, and the extended total dollar value of each item.

The items must be grouped by city and must include a total for each city and a grand total. In addition, Materials wants a summary report which lists the total dollar value of the parts located in each city and what percentage of the total inventory value is represented by the local totals.

<table>
<thead>
<tr>
<th>CITY</th>
<th>PART NUMBER</th>
<th>PART DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHICAGO</td>
<td>000-15-428</td>
<td>BOOKS, SCHOOL COPY</td>
</tr>
<tr>
<td></td>
<td>000-16-490</td>
<td>BAGS, GOLF CLUB</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-10-944</td>
<td>PANEL, SOLAR</td>
</tr>
<tr>
<td></td>
<td>000-53-100</td>
<td>REFRIGERATORS, HOUSEHOLD</td>
</tr>
<tr>
<td></td>
<td>000-79-740</td>
<td>BEDS, WOODEN</td>
</tr>
<tr>
<td></td>
<td>000-81-190</td>
<td>DESKS, STEEL</td>
</tr>
<tr>
<td></td>
<td>000-82-150</td>
<td>TABLES, PICNIC</td>
</tr>
<tr>
<td>HAMMOND</td>
<td>000-79-750</td>
<td>CARPETS, FABRIC (20' X 40')</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-15-980</td>
<td>FAUCETS, BATH TUB</td>
</tr>
<tr>
<td></td>
<td>000-51-260</td>
<td>PIPE, IRON OR STEEL (3'' X 96'')</td>
</tr>
<tr>
<td></td>
<td>000-60-680</td>
<td>BATTERIES, ELECTRIC DRY CELL</td>
</tr>
<tr>
<td></td>
<td>001-78-200</td>
<td>AIR BRAKES</td>
</tr>
<tr>
<td></td>
<td>001-79-000</td>
<td>AXLE SHAFTS</td>
</tr>
<tr>
<td></td>
<td>001-83-800</td>
<td>BRAKE DRUMS</td>
</tr>
<tr>
<td></td>
<td>001-84-900</td>
<td>CYLINDER SLEEVES</td>
</tr>
<tr>
<td></td>
<td>001-85-400</td>
<td>DRIVE SHAFTS</td>
</tr>
<tr>
<td>KANS CT</td>
<td>000-17-037</td>
<td>SIDING, ALUMINUM (24'' X 72'')</td>
</tr>
<tr>
<td>MAMMOND</td>
<td>000-19-360</td>
<td>WALLBOARD, FIBERBOARD (48'' X 96'')</td>
</tr>
<tr>
<td>MEMPHIS</td>
<td>001-84-200</td>
<td>BUMPERS</td>
</tr>
<tr>
<td></td>
<td>001-85-200</td>
<td>DOORS</td>
</tr>
<tr>
<td></td>
<td>001-86-600</td>
<td>FENDERS</td>
</tr>
<tr>
<td></td>
<td>001-88-800</td>
<td>HUBS</td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>000-11-576</td>
<td>MACHINES, CALCULATING</td>
</tr>
<tr>
<td></td>
<td>000-12-268</td>
<td>DRYERS, HAIR</td>
</tr>
<tr>
<td></td>
<td>000-62-270</td>
<td>HUMIDIFIERS, PORTABLE</td>
</tr>
<tr>
<td>ST PAUL</td>
<td>000-12-440</td>
<td>MOWERS, LAWN</td>
</tr>
<tr>
<td></td>
<td>000-13-325</td>
<td>SAWS, CHAIN</td>
</tr>
</tbody>
</table>
CA-Easytrieve/Plus produces both reports with only one pass of the Inventory Master File.

- The first report is similar to the previous example, without the parts descriptions, and with added dollar values.
- The second report requests the SUMMARY option, which prints only summary total lines - no detail lines are printed.

The percentages are calculated in the BEFORE-BREAK procedure, using the total of the extended values generated in the JOB activity.

```
* EXAMPLE 15.12
*    FILE   INVMSTR FB(200 3000)
5 %INVMSTR
*    ITEM-EXT-VALUE     W 6 P 2    HEADING('EXTENDED' 'VALUE')
46   TOTAL-EXT-VALUE    S 7 P 2
47   PERCENT            W 3 P 2    HEADING('PERCENT OF' 'TOTAL VALUE')
48 JOB
49   CALC EXTENDED ITEM VALUE AND TOTAL OF ITEM VALUES
50 *   ITEM-EXT-VALUE = ITEM-SELLING-PRICE * ITEM-LAST-INVENTORY-QUANTITY
52   TOTAL-EXT-VALUE = TOTAL-EXT-VALUE + ITEM-EXT-VALUE
53 *   PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE
56   PRINT  SMY-BY-CITY
57 *
58 REPORT   INV-BY-CITY  SPREAD      LINESIZE 80
59   SEQUENCE  LOCATION-CITY PART-NUMBER
60   CONTROL  LOCATION-CITY
61   TITLE   1 'INVENTORY BY CITY ORDERED BY PART NUMBER'
62   LINE    1 LOCATION-CITY PART-NUMBER -
       ITEM-LAST-INVENTORY-QUANTITY ITEM-SELLING-PRICE -
       ITEM-EXT-VALUE
63 *
64 REPORT   SMY-BY-CITY  SUMMARY    LINESIZE 80
65   SEQUENCE  LOCATION-CITY
66   CONTROL  LOCATION-CITY
67   TITLE   1 'INVENTORY VALUE SUMMARY BY CITY'
68   LINE    1 LOCATION-CITY ITEM-EXT-VALUE PERCENT
69 BEFORE-BREAK. PROC
71   PERCENT = ITEM-EXT-VALUE * 100 / TOTAL-EXT-VALUE + .005
72 END-PROC
```
### Expanded Inventory Report

**11/18/83 INVENTORY BY CITY ORDERED BY PART NUMBER PAGE 1**

<table>
<thead>
<tr>
<th>CITY</th>
<th>PART NUMBER</th>
<th>LAST INVENTORY</th>
<th>QUANTITY</th>
<th>SELLING PRICE (DOLLARS)</th>
<th>EXTENDED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHICAGO</td>
<td>000-15-428</td>
<td>41,353</td>
<td>12.95</td>
<td>535,521.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-16-490</td>
<td>238</td>
<td>49.95</td>
<td>11,888.10</td>
<td></td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-10-944</td>
<td>854</td>
<td>54.99</td>
<td>46,961.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-53-100</td>
<td>181</td>
<td>879.95</td>
<td>159,270.95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-79-740</td>
<td>81</td>
<td>870.00</td>
<td>70,480.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-81-190</td>
<td>35</td>
<td>389.95</td>
<td>13,648.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-82-150</td>
<td>134</td>
<td>199.09</td>
<td>26,785.26</td>
<td></td>
</tr>
<tr>
<td>E MOLIN</td>
<td></td>
<td>1,285</td>
<td>2,394.78</td>
<td>317,135.92</td>
<td></td>
</tr>
<tr>
<td>HAMMOND</td>
<td>000-70-750</td>
<td>358</td>
<td>425.00</td>
<td>152,150.00</td>
<td></td>
</tr>
<tr>
<td>HAMMOND</td>
<td></td>
<td>358</td>
<td>425.00</td>
<td>152,150.00</td>
<td></td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-15-980</td>
<td>3,150</td>
<td>14.29</td>
<td>45,013.50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-51-260</td>
<td>14,389</td>
<td>15.25</td>
<td>219,432.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-60-680</td>
<td>654</td>
<td>54.90</td>
<td>35,904.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-78-200</td>
<td>385</td>
<td>59.88</td>
<td>23,053.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-79-000</td>
<td>385</td>
<td>59.88</td>
<td>23,053.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-83-800</td>
<td>439</td>
<td>43.59</td>
<td>19,136.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-84-900</td>
<td>86</td>
<td>31.59</td>
<td>2,716.74</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-85-400</td>
<td>109</td>
<td>81.45</td>
<td>8,878.05</td>
<td></td>
</tr>
<tr>
<td>INDIANP</td>
<td></td>
<td>19,597</td>
<td>360.83</td>
<td>377,188.75</td>
<td></td>
</tr>
<tr>
<td>KANS CT</td>
<td>000-17-037</td>
<td>2,218</td>
<td>8.99</td>
<td>19,939.82</td>
<td></td>
</tr>
<tr>
<td>KANS CT</td>
<td></td>
<td>2,218</td>
<td>8.99</td>
<td>19,939.82</td>
<td></td>
</tr>
<tr>
<td>MAMMOND</td>
<td>000-19-360</td>
<td>2,810</td>
<td>18.95</td>
<td>53,249.50</td>
<td></td>
</tr>
<tr>
<td>MAMMOND</td>
<td></td>
<td>2,810</td>
<td>18.95</td>
<td>53,249.50</td>
<td></td>
</tr>
</tbody>
</table>

**11/18/83 INVENTORY BY CITY ORDERED BY PART NUMBER PAGE 2**

<table>
<thead>
<tr>
<th>CITY</th>
<th>PART NUMBER</th>
<th>LAST INVENTORY</th>
<th>QUANTITY</th>
<th>SELLING PRICE (DOLLARS)</th>
<th>EXTENDED VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMPHIS</td>
<td>001-84-200</td>
<td>653</td>
<td>99.88</td>
<td>65,221.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-85-200</td>
<td>2,210</td>
<td>195.50</td>
<td>432,055.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-86-600</td>
<td>3,403</td>
<td>159.88</td>
<td>544,071.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001-88-800</td>
<td>3,952</td>
<td>55.95</td>
<td>221,114.40</td>
<td></td>
</tr>
<tr>
<td>MEMPHIS</td>
<td></td>
<td>10,218</td>
<td>511.21</td>
<td>1,262,462.68</td>
<td></td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>000-11-576</td>
<td>88</td>
<td>119.66</td>
<td>10,539.08</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-12-268</td>
<td>805</td>
<td>38.88</td>
<td>31,298.40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000-62-270</td>
<td>245</td>
<td>98.97</td>
<td>24,247.65</td>
<td></td>
</tr>
<tr>
<td>MUSKEGN</td>
<td></td>
<td>1,138</td>
<td>257.51</td>
<td>66,076.13</td>
<td></td>
</tr>
<tr>
<td>ST PAUL</td>
<td>000-12-440</td>
<td>819</td>
<td>243.69</td>
<td>199,582.11</td>
<td></td>
</tr>
<tr>
<td>ST PAUL</td>
<td>000-13-325</td>
<td>799</td>
<td>159.66</td>
<td>127,568.34</td>
<td></td>
</tr>
<tr>
<td>ST PAUL</td>
<td></td>
<td>1,618</td>
<td>403.35</td>
<td>327,150.45</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>88,833</td>
<td>4,443.52</td>
<td>3,122,762.70</td>
<td></td>
</tr>
</tbody>
</table>
Error Correction

After reviewing the Inventory by City report in Example 15.12, shown in the sample report, an error has been detected in the Inventory Master File. The location for part number 000-19-360 is currently MAMMOND instead of the correct city HAMMOND. A CA-Easytrieve/Plus job can correct it easily.

The required job reads the existing file, finds the record in error, makes the correction, generates an audit trail to reflect the change, and outputs an updated master file. All of the records in the updated file are identical to the current file, except the record for part number 000-19-360.

```
1 *
2 *    EXAMPLE 15.13
3 *
4 FILE    INVMSTR   FB(200 3000)
5 %INVMSTR
44   UPDATE-STATUS  W 6 A
45 *
46 FILE    NEWMSTR   FB(200 3000)
47 *
48 JOB
49 IF  PART-NUMBER = 00019360      . * SCAN FOR THE RECORD IN ERROR
51     UPDATE-STATUS = 'BEFORE'      . * INDICATE BEFORE UPDATE
53     PRINT AUDIT-TRAIL             . * OUTPUT AUDIT TRAIL BEFORE UPDATE
55     LOCATION-CITY = 'HAMMOND'     . * MODIFY RECORD
57     UPDATE-STATUS = 'AFTER'       . * INDICATE AFTER UPDATE
59     PRINT AUDIT-TRAIL             . * OUTPUT AUDIT TRAIL AFTER UPDATE
61 END-IF
62 *
63 PUT  NEWMSTR  FROM INVMSTR      . * OUTPUT UPDATED FILE
65 *
66 REPORT   AUDIT-TRAIL              LINESIZE 80
67 TITLE   1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
68 LINE    1 PART-NUMBER LOCATION-CITY UPDATE-STATUS
```
Inventory Reduction

A new accountant for the company wants to reduce the inventory of truck parts (commodity group 19720) by 15 percent. She thinks that this would save a substantial amount of money (since the interest rate is so high) and, therefore, has requested a report which indicates how much could be saved.

The Inventory Reduction report is produced by the following steps:
1. Select all items in commodity group 19720.
2. Determine the maximum quantity of inventory reduction which does not reduce the stock below 120 percent of the reorder point.
3. Calculate the savings, both for parts value and monthly interest cost.
4. Print a report which provides this information, ordered by decreasing savings.

```
1 *
2 *   EXAMPLE 15.14
3 *
4 FILE  INVMSTR   FB(200 3000)
5 %INVMSTR
44 *
45   MIN-STOCK-LEVEL        W   4 P 0
46   STOCK-REDUCTION-QUANT  W   4 P 0
47   PROPOSED-STOCK-QUANT   W   4 P 0   -
        HEADING('PROPOSED' 'STOCK' 'QUANTITY')
48   STOCK-VALUE-SAVINGS    W   5 P 2  HEADING('STOCK VALUE' 'SAVINGS')
49   STOCK-INT-SAVINGS      W   5 P 2  HEADING('STOCK INTEREST' 'SAVINGS')
50 *
51 JOB
52 IF ITEM-MFGD-COMMODITY-GROUP NE 19720 . * REJECT UNWANTED RECDS
53   GOTO JOB
54 END-IF
56 MIN-STOCK-LEVEL = 1.2 * ITEM-REORDER-POINT + .5
57 IF ITEM-LAST-INVENTORY-QUANTITY LE MIN-STOCK-LEVEL
58   STOCK-REDUCTION-QUANT = 0 . * NO REDUCTION IF ALREADY AT MIN
59   PROPOSED-STOCK-QUANT = ITEM-LAST-INVENTORY-QUANTITY
60 END-IF
61 PERFORM REDUCTION-REPORT
62 GOTO JOB
63 END-IF
64 *
65 STOCK-REDUCTION-QUANT = .15 * ITEM-LAST-INVENTORY-QUANTITY
66 PROPOSED-STOCK-QUANT = -
        ITEM-LAST-INVENTORY-QUANTITY - STOCK-REDUCTION-QUANT
67 IF PROPOSED-STOCK-QUANT LT MIN-STOCK-LEVEL
68 PROPOSED-STOCK-QUANT = MIN-STOCK-LEVEL
```
### Inventory Reduction

```plaintext
69    STOCK-REDUCTION-QUANT = ITEM-LAST-INVENTORY-QUANTITY - 
      PROPOSED-STOCK-QUANT
70    END-IF
71    PERFORM REDUCTION-REPORT
72    *
73    REDUCTION-REPORT. PROC
74    STOCK-VALUE-SAVINGS = STOCK-REDUCTION-QUANT * LAST-PURCHASE-PRICE
75    STOCK-INT-SAVINGS = .015 * STOCK-VALUE-SAVINGS
76    PRINT SAVINGS-REPORT
77    END-PROC
78    *
79    REPORT SAVINGS-REPORT  SKIP 1  LINESIZE 80
80    CONTROL
81    TITLE 1 'STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720'
82    LINE 1 PART-NUMBER  ITEM-LAST-INVENTORY-QUANTITY - 
      PROPOSED-STOCK-QUANT  
        STOCK-VALUE-SAVINGS  STOCK-INT-SAVINGS
83    LINE 2 PART-DESCRIPTION

11/23/83    STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720    PAGE    1

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>LAST INVENTORY QUANTITY</th>
<th>PROPOSED STOCK QUANTITY</th>
<th>STOCK VALUE SAVINGS</th>
<th>STOCK INTEREST SAVINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-86-600</td>
<td>3,403</td>
<td>2,893</td>
<td>40,774.50</td>
<td>611.61</td>
</tr>
<tr>
<td>001-85-200</td>
<td>2,210</td>
<td>1,879</td>
<td>33,060.28</td>
<td>495.90</td>
</tr>
<tr>
<td>001-88-800</td>
<td>3,952</td>
<td>3,360</td>
<td>16,155.68</td>
<td>242.33</td>
</tr>
<tr>
<td>001-84-200</td>
<td>653</td>
<td>556</td>
<td>4,413.50</td>
<td>66.20</td>
</tr>
<tr>
<td>001-78-200</td>
<td>385</td>
<td>328</td>
<td>1,707.15</td>
<td>25.60</td>
</tr>
<tr>
<td>001-79-800</td>
<td>385</td>
<td>328</td>
<td>1,707.15</td>
<td>25.60</td>
</tr>
<tr>
<td>001-83-800</td>
<td>439</td>
<td>374</td>
<td>1,462.50</td>
<td>21.93</td>
</tr>
<tr>
<td>001-85-400</td>
<td>109</td>
<td>93</td>
<td>640.00</td>
<td>9.60</td>
</tr>
<tr>
<td>001-84-900</td>
<td>86</td>
<td>86</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>11,622</td>
<td>9,897</td>
<td>99,920.76</td>
<td>1,498.77</td>
</tr>
</tbody>
</table>
```
Inventory File Update

An inventory has been taken of the truck parts (commodity group 19720), and it is necessary to update the master file with the new quantities. We create a CA-Easytrieve/Plus job to update the appropriate records and produce an audit trail of the changes.

CA-Easytrieve/Plus provides a variety of ways to update files. One method is the technique used in “Error Correction, Example 15.13,” but this requires an IF statement for each record to be modified and is too cumbersome for a large number of records.

Another method is to use the multi-file capabilities of CA-Easytrieve/Plus, which are discussed in Chapter 16, “Advanced Techniques.”

Table Files

An excellent technique to update a moderate number of records is to use a table file. In this example, an instream table is defined. The argument equals the part number and the description contains the new quantity and date of inventory.

As data is read from the master file, a check is made against the table for a match.

- If no match is found, the record is written unmodified.
- If a match occurs, the quantity and inventory date are changed, the updated record is written, and an audit report is generated.

In addition, if the inventory for a particular item has been depleted by more than 20 percent of its original value, a management report is generated.

```plaintext
1 * 2 * EXAMPLE 15.15
3 * 4 FILE INVMSTR FB(200 3000)
5 %INVMSTR
44 *
45 TABLE-DESC  W 20 A
46 NEW-DATE TABLE-DESC 6 N 0 MASK ‘29/99/99’ -
   HEADING(‘NEW’ ‘INVENTORY’ ‘DATE’)
47 NEW-QUANT TABLE-DESC +7  5 N 0 -
   HEADING(‘NEW’ ‘INVENTORY’ ‘QUANTITY’)
48 PERCENT-DROP W 3 P 2 HEADING(‘PERCENT’ ‘DROP IN ’ ‘INVENTORY’)
49 *
50 FILE NEWMSTR FB(200 3000)
51 *
52 FILE UPDTBL TABLE INSTREAM
53 ARG 1 8 N. DESC 10 20 A
```
55 00178200 103181 00312
00179600 101581 00472
00183800 110581 00311
00184200 111581 00434
00184900 102281 00081
00185200 092781 02103
00185400 111081 00073
00186600 111981 03401
00188800 110681 04027
ENDTABLE
56 *
57 JOB
58 SEARCH UPDTBL WITH PART-NUMBER GIVING TABLE-DESC
59 *
60 IF UPDTBL . * IF MATCH FOUND
61 PRINT AUDIT-TRAIL . * OUTPUT AUDIT TRAIL
62 PERFORM EXCESS-CHECK . * CHECK FOR LARGE QUANT VARIATION
63 ITEM-LAST-INVENTORY-DATE = NEW-DATE . * UPDATE DATE AND
64 ITEM-LAST-INVENTORY-QUANTITY = NEW-QUANT . * QUANTITY
65 END-IF
66 *
67 PUT NEWMSTR FROM INVMSTR . * OUTPUT UPDATED FILE
68 *
69 EXCESS-CHECK. PROC
70 IF NEW-QUANT < .8 * ITEM-LAST-INVENTORY-QUANTITY
71 PERCENT-DROP = 100 -
72 (NEW-QUANT * 100 / ITEM-LAST-INVENTORY-QUANTITY)
73 PRINT MGMT-WARNING . * IF UNUSUAL DROP IN QUANTITY
74 END-IF . * INFORM THE MANAGEMENT
75 END-PROC
76 *
77 REPORT AUDIT-TRAIL LINESIZE 80
78 TITLE 1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
79 LINE 1 PART-NUMBER ITEM-LAST-INVENTORY-DATE -
80 ITEM-LAST-INVENTORY-QUANTITY -
81 NEW-DATE NEW-QUANT
82 *
83 REPORT MGMT-WARNING LINESIZE 80
84 TITLE 1 'INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY'
85 LINE 1 PART-NUMBER LOCATION-CITY -
86 ITEM-LAST-INVENTORY-QUANTITY -
87 NEW-QUANT PERCENT-DROP

11/20/83 INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL PAGE 1

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>LAST INVENTORY DATE</th>
<th>LAST INVENTORY QUANTITY</th>
<th>NEW INVENTORY DATE</th>
<th>NEW INVENTORY QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-84-900</td>
<td>9/30/81</td>
<td>86</td>
<td>10/22/81</td>
<td>81</td>
</tr>
<tr>
<td>001-85-200</td>
<td>8/31/81</td>
<td>2,210</td>
<td>9/27/81</td>
<td>2,103</td>
</tr>
<tr>
<td>001-85-400</td>
<td>8/31/81</td>
<td>109</td>
<td>11/10/81</td>
<td>73</td>
</tr>
<tr>
<td>001-86-600</td>
<td>10/30/81</td>
<td>3,403</td>
<td>11/19/81</td>
<td>3,401</td>
</tr>
<tr>
<td>001-88-800</td>
<td>10/30/81</td>
<td>3,952</td>
<td>11/06/81</td>
<td>4,827</td>
</tr>
<tr>
<td>001-79-000</td>
<td>9/30/81</td>
<td>385</td>
<td>10/15/81</td>
<td>434</td>
</tr>
<tr>
<td>001-83-800</td>
<td>9/30/81</td>
<td>439</td>
<td>11/05/81</td>
<td>311</td>
</tr>
<tr>
<td>001-84-200</td>
<td>9/30/81</td>
<td>653</td>
<td>11/15/81</td>
<td>472</td>
</tr>
<tr>
<td>001-78-200</td>
<td>9/30/81</td>
<td>385</td>
<td>10/31/81</td>
<td>312</td>
</tr>
</tbody>
</table>
Reorder Notification Report

The Materials Department needs a program that reorders parts automatically when quantities get below a specified level. The program should provide three reports:

- A master activity report for the materials department,
- A set of purchase orders to initiate the ordering, and
- A receiving report for each warehouse which receives the ordered goods.

An effort is being made to build up stock, so an item should be reordered when the current quantity is at, or below, 400 percent of the reorder point.

The number of items to be ordered is equal to the LAST-PURCHASE-QUANTITY. If an item is below the reorder point, the order quantity should be increased 20 percent over the last quantity. This is an update job since the last purchase date and quantity are modified and a new master is written.

As complicated as this job sounds, the basic features of CA-Easytrieve/Plus still provide for a simple program. Each record in the inventory master is read.

- If the item does not require reordering, it is output as it is to the new master file.
- If a reorder is required, the desired quantity is established, the LAST-PURCHASE data is updated, an extended total for the item is calculated, the reports are written, and the updated master file record is output.

The three reports generated from this program demonstrate the power and flexibility of CA-Easytrieve/Plus. The first report is a simple control report which lists all items ordered.

**Note:** Use the SUM statement to explicitly specify which fields to total at control breaks. It does not make sense to total the purchase quantity or estimated item price.
The second report demonstrates how a form with variable information is generated. All data which is constant on a page is defined in a long TITLE. Variable information is defined through LINE statements. Final totals are suppressed. A new page and renumbering are requested at each vendor control break.

**Note:** The use of control variables is in the title lines.

The final report is again a simple control report, but controlled on warehouse location, instead of vendor.

**Note:** Again, the use of the control variable is on the title line.

```plaintext
```
### Reorder Notification Report

#### Basic Examples 15–37

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-0-562</td>
<td>001-78-200</td>
<td>600</td>
<td>29.95</td>
<td>17,970.00</td>
</tr>
<tr>
<td></td>
<td>MILW WI</td>
<td>AIR BRAKES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00-00-9-128</td>
<td>001-84-900</td>
<td>600</td>
<td>16.29</td>
<td>9,774.00</td>
</tr>
<tr>
<td></td>
<td>BAY CIT MI</td>
<td>CYLINDER SLEEVES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-03-0-443</td>
<td>000-81-190</td>
<td>360</td>
<td>195.69</td>
<td>70,448.40</td>
</tr>
</tbody>
</table>

VENDOR-NUMBER TOTAL: 70,448.40
### 11/20/83 PURCHASE ORDER ACTIVITY BY VENDOR

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>34-89-7-210</td>
<td>000-53-100</td>
<td>2</td>
<td>450.67</td>
<td>901.34</td>
</tr>
<tr>
<td>DES MOI IA</td>
<td>REFRIGERATORS, HOUSEHOLD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VENDOR-NUMBER TOTAL: 901.34

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>54-96-3-251</td>
<td>000-11-576</td>
<td>1,008</td>
<td>59.88</td>
<td>60,359.04</td>
</tr>
<tr>
<td>HOUST TX</td>
<td>MACHINES, CALCULATING</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VENDOR-NUMBER TOTAL: 60,359.04

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>65-49-8-318</td>
<td>000-82-150</td>
<td>250</td>
<td>95.80</td>
<td>23,950.00</td>
</tr>
<tr>
<td>TUCS AZ</td>
<td>TABLES, PICNIC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VENDOR-NUMBER TOTAL: 23,950.00

FINAL TOTAL: 258,872.78

---

### 11/20/83 ABC COMPANY PURCHASE ORDER

PO# 1120810000

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-0-562</td>
<td>001-78-200</td>
<td>600</td>
<td>29.95</td>
<td>17,970.00</td>
</tr>
<tr>
<td>MILW WI</td>
<td>AIR BRAKES</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-0-562</td>
<td>001-79-000</td>
<td>600</td>
<td>29.95</td>
<td>17,970.00</td>
</tr>
<tr>
<td>MILW WI</td>
<td>AXLE SHAFTS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-0-562</td>
<td>001-85-400</td>
<td>300</td>
<td>40.00</td>
<td>12,000.00</td>
</tr>
<tr>
<td>MILW WI</td>
<td>DRIVE SHAFTS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

47,940.00
<table>
<thead>
<tr>
<th>Date</th>
<th>Company</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/20/83</td>
<td>ABC COMPANY</td>
<td>1</td>
</tr>
<tr>
<td>PURCHASE ORDER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO#</td>
<td>1120810001</td>
<td></td>
</tr>
<tr>
<td>VENDOR</td>
<td>00-00-9-128</td>
<td></td>
</tr>
<tr>
<td>BAY CIT MI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-84-900</td>
<td>600</td>
<td>16.29</td>
<td>9,774.00</td>
</tr>
</tbody>
</table>

9,774.00

<table>
<thead>
<tr>
<th>Date</th>
<th>Company</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/20/83</td>
<td>ABC COMPANY</td>
<td>1</td>
</tr>
<tr>
<td>PURCHASE ORDER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO#</td>
<td>1120810002</td>
<td></td>
</tr>
<tr>
<td>VENDOR</td>
<td>00-03-4-091</td>
<td></td>
</tr>
<tr>
<td>PHIL</td>
<td>PA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-84-200</td>
<td>1,000</td>
<td>45.50</td>
<td>45,500.00</td>
</tr>
</tbody>
</table>

45,500.00

<table>
<thead>
<tr>
<th>Date</th>
<th>Company</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/20/83</td>
<td>ABC COMPANY</td>
<td>1</td>
</tr>
<tr>
<td>PURCHASE ORDER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO#</td>
<td>1120810003</td>
<td></td>
</tr>
<tr>
<td>VENDOR</td>
<td>10-03-0-443</td>
<td></td>
</tr>
<tr>
<td>LVILLE</td>
<td>KY</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-81-190</td>
<td>360</td>
<td>195.69</td>
<td>70,448.40</td>
</tr>
</tbody>
</table>

70,448.40
<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-03-0-443</td>
<td>000-81-190</td>
<td>360</td>
</tr>
<tr>
<td>34-89-7-210</td>
<td>000-53-100</td>
<td>2</td>
</tr>
<tr>
<td>65-49-8-318</td>
<td>000-82-150</td>
<td>250</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-00-0-562</td>
<td>001-78-200</td>
<td>600</td>
</tr>
<tr>
<td>00-00-0-562</td>
<td>001-79-000</td>
<td>600</td>
</tr>
<tr>
<td>00-00-0-562</td>
<td>001-85-400</td>
<td>300</td>
</tr>
<tr>
<td>00-00-9-128</td>
<td>001-84-900</td>
<td>600</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00-03-4-091</td>
<td>001-84-200</td>
<td>1,000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VENDOR NUMBER</th>
<th>PART NUMBER</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>54-96-3-251</td>
<td>000-11-576</td>
<td>1,008</td>
</tr>
</tbody>
</table>
Introduction

This chapter provides examples of some of the advanced processing techniques available in CA-Easytrieve/Plus. These examples illustrate the use of a more complex operating system interface and CA-Easytrieve/Plus facilities.

A background in data processing is required to fully understand some of the concepts and techniques. If you do not have this background, you should find someone with data processing experience to assist you.

The examples in this chapter stress one or two particular processing techniques. Because of this, no attempt has been made to provide examples as complete as the ones in Chapter 15, “Basic Examples.” These examples do not have any situation posed, as in Chapter 15, “Basic Examples.” The preamble simply describes the processing techniques.

The topics covered in this chapter include:

- GET/PUT of sequential and VSAM files
- Random access of VSAM files
  - READ
  - WRITE (add/replace/delete)
  - POINT
  - Path processing with non-unique keys
- SORT command
  - SORT exit
- Synchronized file processing
- Advanced report features
  - Control LEVEL
  - S-fields
  - TERMINATION procedure
  - ENDPAGE procedure
  - SUMMARY file
- Processing JCL parameters
- Macro definition and processing.
Selected Control Break Processing

Sometimes it is desirable to perform processing at control breaks, based on which variable caused the break. For example, it can be useful to output final break information in a different format from the one used for intermediate breaks. This example lists employee totals by branch. The normal summary line for the final break (LEVEL = 3) is suppressed, and a DISPLAY statement is used instead.

```
1 *
2 *   EXAMPLE 16.1
3 *
4 FILE     PERSNL FB(150 1800)
5 %PERSNL
35 *
36 *
37 JOB
38 PRINT  EMPLOYEE-TALLY . * PRINT REPORT FOR ALL EMPLOYEES
40 *
41 REPORT  EMPLOYEE-TALLY SUMMARY LINESIZE 80
42 SEQUENCE  REGION  BRANCH
43 CONTROL  FINAL NOPRINT  REGION  BRANCH
44 TITLE  1 'EMPLOYEE TALLY BY REGION AND BRANCH'
45 HEADING TALLY ('NUMBER OF 'EMPLOYEES')
46 LINE  1 REGION  BRANCH  TALLY
47 *
48 BEFORE-BREAK. PROC
50 IF  LEVEL = 3               . * IF FINAL BREAK
52 DISPLAY  SKIP 3 'TOTAL COMPANY EMPLOYEES:' TALLY
53 END-IF
54 END-PROC
```

```
1/24/84            EMPLOYEE TALLY BY REGION AND BRANCH       PAGE      1

NUMBER OF
REGION  BRANCH  EMPLOYEES
1        01          2
1        02          2
1        03          2
1        04          4
1                   10
2        01          1
2        02          2
2        03          4
2        04          1
2        05          2
2                   10
3        01          4
3        02          6
3        03          6
3        04          3
3                   19
4        01          3
4        02          2
4        03          3
4        04          1
4                   9

TOTAL COMPANY EMPLOYEES:                 48
```
Summary File Processing

At times, it is desirable to order a report on a value which is the result of a summing operation for a previous report. For example, using the Inventory Master File, a previous report (Example 15.11) listed the value of the inventory at each warehouse, ordered on the warehouse's city location.

To highlight the most valuable stock locations, this report might be more useful ordered by the decreasing value of the inventory for each warehouse.

To do this with CA-Easytrieve/Plus, we use the multijob and summary file facilities. The first job generates a normal summary report by location, along with a special summary file which is used as input to the second job. For comparison purposes, the two reports are identical except for the ordering.

Remember, use this technique when a report must be ordered, based on values which are calculated across groups of input records. The process is easy using the multi-job and the summary file facilities.

```
1 *  EXAMPLE 16.2
2 *
3 FILE  INVMSTR  FB(200 3000)
4 %INVMSTR
5 *
6 TOTAL-VALUE  W  6 P 2  HEADING('TOTAL VALUE' 'OF PARTS')
7 *
8 SUMMARY FILE DEFINITION
9 *
10 THE SUMMARY FILE FROM THE INV-BY-CITY REPORT HAS THE FOLLOWING
11 GENERAL FORMAT:  
12 CONTROL FIELDS + TALLY + TOTAL FIELDS
13 *
14 FOR THIS PARTICULAR FILE IT HAS THE FOLLOWING FORMAT:  
15 *
16 LOCATION   LENGTH     FIELD
17 1          7       CONTROL FIELD - LOCATION-CITY
18 8         10       TALLY
19 18        10       TOTAL FIELD - TOTAL-VALUE
20 *
21 NOTICE IN THE BELOW FIELD DEFINITIONS HOW THE FIELDS ARE DEFINED
22 TO FIT WITHIN THE FORMAT.  SINCE THE TALLY VALUE WILL NOT EXCEED
23 SEVEN DIGITS, PARTS-IN-CITY ONLY DEFINES THAT PART.  THE SAME IS
24 TRUE FOR VALUE-IN-CITY.  THE LENGTH OF THE RECORD IS 28 BYTES AND
25 IT IS SPOOLED TO THE VIRTUAL FILE MANAGER.
26 *
27 FILE  SMYFIL  F 28  VIRTUAL
28   CITY            1  7 A
29   PARTS-IN-CITY  14  4 P 0    HEADING('NUMBER OF' 'PART TYPES')
30   VALUE-IN-CITY  22  6 P 2    HEADING('TOTAL VALUE' 'OF PARTS')
31 *
32 JOB
33 TOTAL-VALUE = LAST-PURCHASE-PRICE * ITEM-LAST-INVENTORY-QUANTITY
34 *
35 PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE
36 *
37 REPORT  INV-BY-CITY   SUMMARY  SUMFILE SMYFIL  LINESIZE 80
38 SEQUENCE  LOCATION-CITY
39 CONTROL  LOCATION-CITY
```
### INVENTORY VALUE BY CITY

<table>
<thead>
<tr>
<th>City</th>
<th>Number of Part Types</th>
<th>Total Value of Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>2</td>
<td>233,379.60</td>
</tr>
<tr>
<td>E Molin</td>
<td>5</td>
<td>160,522.97</td>
</tr>
<tr>
<td>Hammond</td>
<td>1</td>
<td>81,982.00</td>
</tr>
<tr>
<td>Indiana</td>
<td>8</td>
<td>191,825.42</td>
</tr>
<tr>
<td>Indiana</td>
<td>1</td>
<td>32,794.14</td>
</tr>
<tr>
<td>Malmö</td>
<td>1</td>
<td>25,009.00</td>
</tr>
<tr>
<td>Memphis</td>
<td>4</td>
<td>630,366.23</td>
</tr>
<tr>
<td>Malmö</td>
<td>2</td>
<td>162,160.05</td>
</tr>
<tr>
<td>St Paul</td>
<td>2</td>
<td>162,160.05</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>1,527,776.43</td>
</tr>
</tbody>
</table>

---

### VALUE OF INVENTORY IN EACH CITY BY DECREASING VALUE

<table>
<thead>
<tr>
<th>City</th>
<th>Number of Part Types</th>
<th>Total Value of Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memphis</td>
<td>4</td>
<td>630,366.23</td>
</tr>
<tr>
<td>Chicago</td>
<td>2</td>
<td>233,379.60</td>
</tr>
<tr>
<td>Indiana</td>
<td>8</td>
<td>191,825.42</td>
</tr>
<tr>
<td>St Paul</td>
<td>2</td>
<td>162,160.05</td>
</tr>
<tr>
<td>E Molin</td>
<td>5</td>
<td>160,522.97</td>
</tr>
<tr>
<td>Hammond</td>
<td>1</td>
<td>81,982.00</td>
</tr>
<tr>
<td>Malmö</td>
<td>1</td>
<td>25,009.00</td>
</tr>
<tr>
<td>Kans CT</td>
<td>1</td>
<td>9,737.02</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>1,527,776.43</td>
</tr>
</tbody>
</table>
Special Report Processing Exits

Report processing provides several special processing exits. Two of these are useful for page and report annotation.

- **ENDPAGE** enables you to perform processing when CA-Easytrieve/Plus reaches the end of a logical page. It is useful for footers or page totals. Specifying a **PAGESIZE** of 12 enables more than one of these small reports to fit onto one sheet of printer paper. The footer appears at the bottom of the page, regardless of the size of the report data.

- **TERMINATION** permits annotation at the end of the report. Typical uses are report routing information, special final total data, or hash totals.

Example 16.3 demonstrates the use of both of these facilities. An important consideration for the **TERMINATION** procedure is to limit field references to S-fields, control fields, and total fields.

```plaintext
1 * 2 * EXAMPLE 16.3 3 * 4 FILE INVMSTR FB(200 3000) 5 %INVMSTR 44 * 45 JOB 46 PRINT INV-BY-CITY . * SELECT EACH RECORD IN FILE 48 * 49 REPORT INV-BY-CITY PAGESIZE 12 LINESIZE 80 50 SEQUENCE LOCATION-CITY PART-NUMBER 51 CONTROL FINAL NOPRINT LOCATION-CITY NEWPAGE 52 TITLE 1 'INVENTORY FOR ' LOCATION-CITY ' BY PART NUMBER' 53 LINE 1 PART-NUMBER PART-DESCRIPTION 54 * 55 ENDPAGE. PROC 57 DISPLAY 'CONFIDENTIAL COMPANY INFORMATION' 58 END-PROC 59 * 60 TERMINATION. PROC 62 DISPLAY NEWPAGE 'ROUTE REPORT TO:' 63 DISPLAY SKIP 2 'R. M. HODGES' 64 DISPLAY 'MATERIALS PROCUREMENT' 65 END-PROC
```

11/24/83 INVENTORY FOR CHICAGO BY PART NUMBER PAGE 1

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>PART DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-15-428</td>
<td>BOOKS, SCHOOL COPY</td>
</tr>
<tr>
<td>000-16-490</td>
<td>BAGS, GOLF CLUB</td>
</tr>
</tbody>
</table>

CONFIDENTIAL COMPANY INFORMATION
<table>
<thead>
<tr>
<th>Part Number</th>
<th>Part Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-10-944</td>
<td>Panel, Solar</td>
</tr>
<tr>
<td>000-53-100</td>
<td>Refrigerators, Household</td>
</tr>
<tr>
<td>000-79-740</td>
<td>Beds, Wooden</td>
</tr>
<tr>
<td>000-81-190</td>
<td>Desks, Steel</td>
</tr>
<tr>
<td>000-82-150</td>
<td>Tables, Picnic</td>
</tr>
</tbody>
</table>

CONFIDENTIAL COMPANY INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Part Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-70-750</td>
<td>Carpets, Fabric (20' x 40')</td>
</tr>
</tbody>
</table>

CONFIDENTIAL COMPANY INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Part Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-15-980</td>
<td>Faucets, Bath Tub</td>
</tr>
<tr>
<td>000-51-260</td>
<td>Pipe, Iron or Steel (3&quot; x 96&quot;)</td>
</tr>
<tr>
<td>000-60-680</td>
<td>Batteries, Electric Dry Cell</td>
</tr>
<tr>
<td>001-78-200</td>
<td>Air Brakes</td>
</tr>
<tr>
<td>001-79-000</td>
<td>Axle Shafts</td>
</tr>
<tr>
<td>001-83-800</td>
<td>Brake Drums</td>
</tr>
</tbody>
</table>

CONFIDENTIAL COMPANY INFORMATION

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Part Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-84-900</td>
<td>Cylinder Sleeves</td>
</tr>
<tr>
<td>001-85-400</td>
<td>Drive Shafts</td>
</tr>
</tbody>
</table>

CONFIDENTIAL COMPANY INFORMATION
<table>
<thead>
<tr>
<th>Date</th>
<th>Inventory for</th>
<th>Location</th>
<th>Page</th>
<th>Part Number</th>
<th>Part Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/24/83</td>
<td>INVENTORY FOR</td>
<td>KANS CT</td>
<td>6</td>
<td>000-17-037</td>
<td>SIDING, ALUMINUM (24&quot; X 72&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CONFIDENTIAL COMPANY INFORMATION</td>
</tr>
<tr>
<td>11/24/83</td>
<td>INVENTORY FOR</td>
<td>MAMMONT</td>
<td>7</td>
<td>000-19-360</td>
<td>WALLBOARD, FIBERBOARD (48&quot; X 96&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CONFIDENTIAL COMPANY INFORMATION</td>
</tr>
<tr>
<td>11/24/83</td>
<td>INVENTORY FOR</td>
<td>MEMPHIS</td>
<td>8</td>
<td>001-84-200</td>
<td>BUMPERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>001-85-200</td>
<td>DOORS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>001-86-600</td>
<td>FENDERS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>001-88-800</td>
<td>HUBS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CONFIDENTIAL COMPANY INFORMATION</td>
</tr>
<tr>
<td>11/24/83</td>
<td>INVENTORY FOR</td>
<td>MUSKEGN</td>
<td>9</td>
<td>000-11-576</td>
<td>MACHINES, CALCULATING</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000-12-268</td>
<td>DRYERS, HAIR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>000-62-270</td>
<td>HUMIDIFIERS, PORTABLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CONFIDENTIAL COMPANY INFORMATION</td>
</tr>
</tbody>
</table>
Sorting Input Files

The CA-Easytrieve/Plus SORT facility is useful to reorder a file or its subset prior to processing. Normally, the SEQUENCE statement is used to order reports during report processing. However, if the sorted file is to be kept, or if more than one report needs to be output in the same order, a SORT activity is more efficient. For example, if you intend to generate five reports in the same order, the input file can be sorted once and no SEQUENCE statements are required. One sort is performed instead of five. Also, this eliminates report spooling if each print file is routed to a different logical printer. With large files, this approach dramatically reduces processing time and temporary disk work space.

In this example, we select all inventory records for items which cost more than $200 and sort them by commodity group. Then, we generate control, detail, and summary reports without SEQUENCE statements. A BEFORE sort exit performs the required record selection.

```
1 *  
2 *   EXAMPLE 16.4  
3 *  
4 FILE INVMSTR FB(200 3000)  
5   VENDOR-CITY 112 7 A  
6   PART-PRICE  62 4 P 2  
7 *  
8 FILE SRTMSTR F 200 VIRTUAL  
9 %INVMSTR  
48 *  
49 SORT INVMSTR TO SRTMSTR BEFORE SCAN-INV USING VENDOR-CITY  
50 *  
51 SCAN-INV. PROC  
53 IF PART-PRICE > 200.00 . * SELECT PART RECORDS IF  
55 SELECT . * THEY COST MORE THAN $200  
57 END-IF  
58 END-PROC  
59 *  
60 *  
```
61 JOB INPUT SRTMSTR
62 PRINT DETAIL-RPT
63 PRINT CONTROL-RPT . * PRINT ALL REPORTS
65 PRINT SUMMARY-RPT
66 *
67 REPORT DETAIL-RPT LINESIZE 80
68 TITLE 1 'DETAIL LIST OF ALL PARTS SELLING FOR MORE THAN $200'
69 LINE 1 VENDOR-LOCATION-CITY PART-NUMBER PART-DESCRIPTION
70 *
71 REPORT CONTROL-RPT LINESIZE 80
72 CONTROL VENDOR-LOCATION-CITY
73 TITLE 1 'CONTROLLED LIST OF PARTS SELLING FOR MORE THAN $200'
74 TITLE 2 'BY VENDOR LOCATION'
75 LINE 1 VENDOR-LOCATION-CITY PART-NUMBER PART-DESCRIPTION TALLY
76 *
77 REPORT SUMMARY-RPT SUMMARY LINESIZE 80
78 CONTROL VENDOR-LOCATION-CITY
79 TITLE 1 'SUMMARY LIST OF PARTS SELLING FOR MORE THAN $200'
80 TITLE 2 'BY VENDOR LOCATION'
81 LINE 1 VENDOR-LOCATION-CITY TALLY

11/25/83 DETAIL LIST OF ALL PARTS SELLING FOR MORE THAN $200 PAGE 1

VENDOR PART
CITY NUMBER PART DESCRIPTION

DES MOI 000-53-100 REFRIGERATORS, HOUSEHOLD
GR BAY 000-12-440 MOWERS, LAWN
LVILLE 000-81-190 DESKS, STEEL
NEWARK 000-79-750 CARPETS, FABRIC (20' X 40')
TUPEL 000-79-740 BEDS, WOODEN

-------------------------------------------------------------------------

11/25/83 CONTROLLED LIST OF PARTS SELLING FOR MORE THAN $200 PAGE 1
BY VENDOR LOCATION

VENDOR PART
CITY NUMBER PART DESCRIPTION TALLY

DES MOI 000-53-100 REFRIGERATORS, HOUSEHOLD 1
DES MOI
GR BAY 000-12-440 MOWERS, LAWN 1
GR BAY
LVILLE 000-81-190 DESKS, STEEL 1
LVILLE
NEWARK 000-79-750 CARPETS, FABRIC (20' X 40') 1
NEWARK
TUPEL 000-79-740 BEDS, WOODEN 1
TUPEL 5
Synchronized File Facility - File Update

In Chapter 15, “Basic Examples,” we illustrated the technique for performing an update of the Inventory Master File using a table (Example 15.15). In this example, we demonstrate the use of the synchronized file processing facility to perform this update.

The synchronized update process requires the files to be in the same order by the defined key. This requirement causes us to add two Sort activities to the job. From this example, you can see that the table file method is simpler for moderate updates.

A more complete example of synchronized file updating is in Chapter 18, “Project Management System.”

```
1 *  
2 *   EXAMPLE 16.5  
3 *  
4 FILE   INVMSTR   FB(200 3000)  
5   MSTR-PART#   36 8 N  
6 *  
7 FILE   SRTMSTR   F 200 VIRTUAL  
8 %INVMSTR  
47 *  
48   PERCENT-DROP   W 3 P 2 HEADING('PERCENT' 'DROP')  
49 *  
50 FILE   NEWMSTR   FB(200 3000)  
51 *  
52 FILE   UPDTRAN   F 80  
53   TRAN-PART#   1 8 N  
54 *  
55 FILE   SRTTRAN   F 80 VIRTUAL  
56   TRAN-PART-NBR   1 8 N  
57   TRAN-INV-DATE   10 6 N HEADING('NEW' 'INVENTORY' 'DATE') -  
58   TRAN-INV-QUAN   17 5 N 0 HEADING('NEW' 'INVENTORY' 'QUANTITY')  
59 *  
60 *  
61 SORT   INVMSTR TO SRTMSTR USING MSTR-PART#  
62 *  
63 SORT   UPDTRAN TO SRTTRAN USING TRAN-PART#  
64 *  
65 JOB   INPUT (SRTMSTR KEY PART-NUMBER -  
                      SRTTRAN KEY TRAN-PART-NBR)  
```
66 *
67 IF MATCHED . * IF MATCH FOUND
69 PRINT AUDIT-TRAIL . * OUTPUT AUDIT TRAIL
71 PERFORM EXCESS-CHECK . * CHECK FOR LARGE QUANT VARIATION
73 ITEM-LAST-INVENTORY-DATE = TRAN-INV-DATE . * UPDATE DATE AND
75 ITEM-LAST-INVENTORY-QUANTITY = TRAN-INV-QUAN . * QUANTITY
77 END-IF
78 *
79 OUTPUT NEW MASTER IF MATCHED OR MASTER AND NO TRANSACTION
80 DISPLAY ERROR MSG FOR A TRANSACTION WITH NO MATCHING MASTER
81 *
82 IF SRMSTR
83 PUT NEWSTR FROM SRMSTR . * OUTPUT UPDATED FILE
85 ELSE
86 DISPLAY 'NO MASTER FILE RECORD FOR TRANSACTION ' TRAN-PART-NBR
87 END-IF
88 *
89 EXCESS-CHECK. PROC
91 IF TRAN-INV-QUAN < .8 * ITEM-LAST-INVENTORY-QUANTITY
92 PERCENT-DROP = 100 -
   (TRAN-INV-QUAN * 100 / ITEM-LAST-INVENTORY-QUANTITY)
93 PRINT MGMT-WARNING . * IF UNUSUAL DROP IN QUANTITY
95 END-IF . * INFORM THE MANAGEMENT
97 END-PROC
98 *
99 REPORT AUDIT-TRAIL LINESIZE 80
100 TITLE 1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
101 LINE 1 PART-NUMBER ITEM-LAST-INVENTORY-DATE -
   ITEM-LAST-INVENTORY-QUANTITY -
   TRAN-INV-DATE TRAN-INV-QUAN
102 *
103 REPORT MGMT-WARNING LINESIZE 80
104 TITLE 1 'INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY'
105 LINE 1 PART-NUMBER LOCATION-CITY -
   ITEM-LAST-INVENTORY-QUANTITY -
   TRAN-INV-QUAN PERCENT-DROP

11/25/83 INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL PAGE 1

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>LAST PART NUMBER</th>
<th>LAST INVENTORY DATE</th>
<th>LAST INVENTORY QUANTITY</th>
<th>NEW INVENTORY DATE</th>
<th>NEW INVENTORY QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-78-200</td>
<td>001-78-200</td>
<td>9/30/81</td>
<td>385</td>
<td>10/31/81</td>
<td>312</td>
</tr>
<tr>
<td>001-79-000</td>
<td>001-79-000</td>
<td>9/30/81</td>
<td>385</td>
<td>10/15/81</td>
<td>434</td>
</tr>
<tr>
<td>001-83-800</td>
<td>001-83-800</td>
<td>9/30/81</td>
<td>439</td>
<td>11/05/81</td>
<td>311</td>
</tr>
<tr>
<td>001-84-200</td>
<td>001-84-200</td>
<td>9/30/81</td>
<td>653</td>
<td>11/15/81</td>
<td>472</td>
</tr>
<tr>
<td>001-84-900</td>
<td>001-84-900</td>
<td>9/30/81</td>
<td>86</td>
<td>10/22/81</td>
<td>81</td>
</tr>
<tr>
<td>001-85-200</td>
<td>001-85-200</td>
<td>8/31/81</td>
<td>2,210</td>
<td>9/27/81</td>
<td>2,103</td>
</tr>
<tr>
<td>001-85-400</td>
<td>001-85-400</td>
<td>8/31/81</td>
<td>189</td>
<td>11/10/81</td>
<td>73</td>
</tr>
<tr>
<td>001-86-600</td>
<td>001-86-600</td>
<td>10/30/81</td>
<td>3,403</td>
<td>11/19/81</td>
<td>3,401</td>
</tr>
<tr>
<td>001-88-000</td>
<td>001-88-000</td>
<td>10/30/81</td>
<td>3,952</td>
<td>11/06/81</td>
<td>4,027</td>
</tr>
</tbody>
</table>
Reformat Printed Output from IDCAMS

Frequently, it is useful to read the printed output of another program and format the information in a different manner. This example shows how a CA-Easytrieve/Plus job reads the output of an IDCAMS utility run, processes the data, and then generates a report which would not otherwise be available.

This example shows how to extract VSAM CI/CA split information for review. The Assignment statement, using the OR feature, is a way of converting the hyphens (x'60') on the report to EBCDIC zeros (x'FO').

```
1 *
2 *    EXAMPLE 16.6
3 *
4 FILE    AMSINFO   VB(125 629)  WORKAREA 125
5   P-CLEAR            1 125 A
6   P-LIST-DATA        1  70 A
7   P-CLUSTER-ID       2   9 A
8   P-CLUSTER-NAME    18  40 A
9   P-COMPONENT-ID     5   5 A
10   P-SPLIT-ID        38   7 A
11   P-SPLIT-TYPE      45   2 A
12   P-SPLIT-COUNT     57   5 N
13 *
14 *   WORKING STORAGE FIELDS
15 *
16   CLUSTER            W  40 A
17   COMPONENT          W   5 A
18   SPLIT-TYPE         W   2 A
19   SPLIT-COUNT        W   5 N 0
20 *
21 JOB
22   PRINT  INPUT-DATA
23 *
24 IF P-CLUSTER-ID = 'CLUSTER -'
25     CLUSTER = P-CLUSTER-NAME
26 END-IF
27 *
28 IF P-COMPONENT-ID = 'DATA ', 'INDEX'
29     COMPONENT = P-COMPONENT-ID
30 END-IF
31 *
32 IF P-SPLIT-ID = 'SPLITS-
33     SPLIT-TYPE = P-SPLIT-TYPE
34     SPLIT-COUNT = P-SPLIT-COUNT OR X'F0F0F0F0F0'
35 END-IF
```
36 *
37 IF SPLIT-COUNT NOT ZERO
38   PRINT SPLIT-INFO
39   SPLIT-COUNT = 0
40 END-IF
41 *
42 MOVE SPACES TO P-CLEAR  * CLEAR WORKAREA AFTER EACH RECORD
44 *
45 REPORT SPLIT-INFO SPREAD LINESIZE 80
46 SEQUENCE SPLIT-TYPE CLUSTER COMPONENT
47 CONTROL SPLIT-TYPE
48 TITLE 1 'VSAM CI/CA SPLIT INFORMATION'
49 LINE 1 SPLIT-TYPE SPLIT-COUNT COMPONENT CLUSTER
50 *
51 REPORT INPUT-DATA NOHEADING LINESIZE 80 LIMIT 50
52 TITLE 1 'TYPICAL INPUT DATA FOR RUN - FIRST 50 LINES'
53 LINE 1 P-LIST-DATA
--------------------------------------------------------------------------------
SORT (DEVICE SYSDA ALTSEQ NO MSG DEFAULT  MEMORY MAX   WORK   3)  VFM ( 16 D
--------------------------------------------------------------------------------
11/24/83                  VSAM CI/CA SPLIT INFORMATION              PAGE      1
SPLIT-TYPE   SPLIT-COUNT   COMPONENT                   CLUSTER
CI              23       DATA      ABELMAN.MACRO.FILE
4       DATA      VARVERI.MACRO.LIBRARY
CI              27
27
11/24/83 TYPICAL INPUT DATA FOR RUN - FIRST 50 LINES PAGE 1

1IDCAMS SYSTEM SERVICES TIM
0 LISTCAT CATALOG(USER53.USERCAT) ALL
1IDCAMS SYSTEM SERVICES TIM
- LISTING FROM CATALOG -- USER53.USERCAT
0 CLUSTER ------ ABELMAN.MACRO.FILE
HISTORY
OWNER-IDENT--------(NULL) CREATION----------81.237
RELEASE-------------2 EXPIRATION-------81.365
PROTECTION-PSWD-----(NULL) RACF----------------(NO)
ASSOCIATIONS
DATA----VSAMDSET.TE18B950.DFD81237.T927607B.TE18B950
INDEX----VSAMDSET.TE18D390.DFD81237.T927607B.TE18D390
0 DATA ------ VSAMDSET.TE18B950.DFD81237.T927607B.TE18B950
HISTORY
OWNER-IDENT--------(NULL) CREATION----------81.237
RELEASE-------------2 EXPIRATION-------00.000
PROTECTION-PSWD-----(NULL) RACF----------------(NO)
ASSOCIATIONS
CLUSTER--ABELMAN.MACRO.FILE
ATTRIBUTES
KEYLEN----------------30 AVGLRECL------------110 BUFS
RKP-------------------0 MAXRECL-------------110 EXCP
SHROPTNS(1,3) SPEED SUBALLOC NOERASE INDE
UNORDERED REUSE NONSPANDED
STATISTICS
REC-TOTAL-------------138 SPLITSCI--------------23 EXCP
REC-DELETED-----------984 SPLITSCA-------------0 EXTE
REC-INSERTED----------949 FREESPACEMCI--------0 SYST
REC-UPDATED-----------14 FREESPACEMCA--------0
REC-RETRIEVED-------1828 FREESPACEBYTES-----434176
ALLOCATION
SPACE-TYPE------CYLINDER HI-ALLOC-RBA------491520
SPACE-PRI---------1 HI-USED-RBA--------491520
SPACE-SEC--------1
VOLUME
VOLSER---------USER53 PHYREC-SIZE-------2848 HI-A
DEVTYPE------X'3050200A' PHYRECS/TRK-------8 HI-U
VOLFLAG------------PRIME TRACKS/CA----------30 TRAC
EXTENTS:
LOW-CCHH----X'01950000' LOW-RBA-------------0
HIGH-CCHH-----X'0195001D' HIGH-RBA----------491519
0 INDEX ------ VSAMDSET.TE18D390.DFD81237.T927607B.TE18D390
HISTORY
OWNER-IDENT--------(NULL) CREATION----------81.237
RELEASE-------------2 EXPIRATION-------00.000
PROTECTION-PSWD-----(NULL) RACF----------------(NO)
ASSOCIATIONS
CLUSTER--ABELMAN.MACRO.FILE
ATTRIBUTES
The next five examples (16.7 through 16.11) demonstrate the use of CA-Easytrieve/Plus to process VSAM files, which are in increasing use in today's data processing environment. CA-Easytrieve/Plus provides a complete, flexible facility for processing VSAM structures, including access to ESDS, RRDS, and KSDS data sets and any defined PATH. A thorough understanding of VSAM file concepts is required for effective use of CA-Easytrieve/Plus with VSAM files; such understanding is assumed for these examples.

To demonstrate the use of CA-Easytrieve/Plus with VSAM files, two data structures are built from the Personnel Master File. The first structure is an ESDS cluster, with the same format as the sequential file used in Chapter 15, “Basic Examples.” In addition, an alternate index is built across the ESDS, keyed on the employee number. A path is defined for the combination of the alternate index and the base ESDS cluster.

The second structure is a KSDS cluster, built from the Personnel Master File, keyed on the employee number. A non-unique alternate index is built across the KSDS cluster, keyed on the department number. Also, a path is defined for this alternate index and KSDS cluster combination.

Three steps are required to build the above two structures:
1. Define the base clusters through IDCAMS (Example 16.7A).
2. Load the base clusters by means of CA-Easytrieve/Plus (Example 16.7B).
3. Define and build the alternate indexes, and define the paths through IDCAMS (Example 16.7C).
Defining and Loading VSAM Data Sets with Alternate Indexes

This example shows how to define and load a VSAM data set using CA-Easytrieve/Plus. In this example, we build the two VSAM structures described previously. First, we define the base clusters through IDCAMS.

Define Base Clusters through IDCAMS

* EXAMPLE 16.7A

DELETE (RETSYS$.PERSNL.KSDS/MSTPER)
DELETE (RETSYS$.PERSNL.ESDS/MSTPER)
SET MAXCC = 0
DEFINE CLUSTER -
   (NAME(RETSYS$.PERSNL.KSDS) -
   RECORDS(50 10) VOLUMES(USER53) -
   KEYS(5  8)  OWNER(EZTP) -
   RECORDSIZE(150 150)                -
   UPDATEPW(UPDPER) MASTERPW(MSTPER)) -
   DATA (NAME(RETSYS$.PERSNL.KSDS.DATA)) -
INDEX(NAME(RETSYS$.PERSNL.KSDS.INDEX))
DEFINE CLUSTER -
   (NAME(RETSYS$.PERSNL.ESDS) -
   RECORDS(50 10) VOLUMES(USER53) -
   NONINDEXED  OWNER(EZTP) -
   RECORDSIZE(150 150)                -
   UPDATEPW(UPDPER) MASTERPW(MSTPER)) -
   DATA (NAME(RETSYS$.PERSNL.ESDS.DATA))
Load Base Clusters Through CA-Easytrieve/Plus

Next, a CA-Easytrieve/Plus job is used to load the data into the VSAM clusters from the sequential version of the Personnel Master File. A SORT is required to order the KSDS input by employee number.

```plaintext
1 * 2 * EXAMPLE 16.7B 3 * 4 FILE PERSNL FB(150 1800) 5 %PERSNL 35 * 36 FILE PERESDS VS(ES PASSWORD 'UPDPER' CREATE) 37 * 38 FILE PERKSDS VS(PASSWORD 'UPDPER' CREATE) 39 * 40 FILE TWORK F 150 VIRTUAL 41 * 42 JOB FINISH WRAP-UP 43 PUT PERESDS FROM PERSNL . * BUILD ESDS VERSION OF PERSNL 45 * 46 WRAP-UP. PROC 48 DISPLAY NEWPAGE 'TOTAL RECORDS WRITTEN TO PERESDS = ' - RECORD-COUNT(PERESDS) 49 END-PROC 50 * 51 SORT PERSNL TO TWORK USING EMP# . * SORT PERSNL INTO EMP# ORDER 53 * 54 JOB FINISH WRAP-UP 55 PUT PERKSDS FROM TWORK . * BUILD KSDS VERSION OF PERSNL 57 * 58 WRAP-UP. PROC 60 DISPLAY NEWPAGE 'TOTAL RECORDS WRITTEN TO PERKSDS = ' - RECORD-COUNT(PERKSDS) 61 END-PROC
```

TOTAL RECORDS WRITTEN TO PERESDS = 48

TOTAL RECORDS WRITTEN TO PERKSDS = 48
Defining and Building Alternate Indexes and Define Paths

Now that the base clusters are built, use IDCAMS to define and build the alternate indexes, and also to define the paths.

* EXAMPLE 16.7C

```
DEFINE ALTERNATEINDEX -
  (NAME(RETSYS$.PERSNL.ESDS.AX) -
   RELATE(RETSYS$.PERSNL.ESDS/MSTPER) -
   RECORDS(50 10) VOLUMES(USER53) -
   KEYS(5 8) MASTERPW(MSTPER) -
   OWNER(PRO) RECORDSIZE(17 34) -
   REUSE SPEED UNIQUEKEY) -
  DATA -
    (NAME(RETSYS$.PERSNL.ESDS.AX.DATA)) -
   INDEX -
    (NAME(RETSYS$.PERSNL.ESDS.AX.INDEX))
DEFINE ALTERNATEINDEX -
  (NAME(RETSYS$.PERSNL.KSDS.AX) -
   RELATE(RETSYS$.PERSNL.KSDS/MSTPER) -
   RECORDS(50 10) VOLUMES(USER53) -
   KEYS(3 97) MASTERPW(MSTPER) -
   OWNER(PRO) RECORDSIZE(28 99) -
   REUSE SPEED NONUNIQUEKEY) -
  DATA -
    (NAME(RETSYS$.PERSNL.KSDS.AX.DATA)) -
   INDEX -
    (NAME(RETSYS$.PERSNL.KSDS.AX.INDEX))
BLDINDEX -
  IDS(RETSYS$.PERSNL.ESDS/MSTPER) -
  ODS(RETSYS$.PERSNL.ESDS.AX/MSTPER)
BLDINDEX -
  IDS(RETSYS$.PERSNL.KSDS/MSTPER) -
  ODS(RETSYS$.PERSNL.KSDS.AX/MSTPER)
DEFINE PATH -
  (NAME(RETSYS$.PERSNL.ESDS.PATH) -
   PATHENTRY(RETSYS$.PERSNL.ESDS.AX))
DEFINE PATH -
  (NAME(RETSYS$.PERSNL.KSDS.PATH) -
   PATHENTRY(RETSYS$.PERSNL.KSDS.AX))
```
Updating a VSAM KSDS Cluster

This example demonstrates the random reading and updating of a VSAM KSDS cluster. The ESDS is in the original order of the Personnel Master file; it is a reproduction of the sequential version. The file is in order by region number.

In this example, we read all of the records with a region code of 1 from the ESDS. We then use the employee number as the key for the KSDS. The KSDS record is read, modified, and updated on the file. This is a typical random update operation using an input tickler file.

```
1 *
2 *   EXAMPLE 16.8
3 *
4 FILE PERESDS VS ES
5   REGION-CODE   1 1 N
6   EMP-NBR       9 5 N
7 *
8 FILE PERKSDS VS(PASSWORD 'UPDPER' UPDATE)
9 %PERSNL
39 *
40 JOB
41     IF  REGION-CODE GT 1                      . * STOP IF DONE WITH REGION CODE 1
42    END-IF
43 *
44   READ  PERKSDS   KEY EMP-NBR
45   PRINT  BEFORE-UPDATE         . * PRINT BEFORE-UPDATE INFO
46   PAY-GROSS = 1.05 * PAY-GROSS . * GIVE EVERYONE IN REGION 1 A 5% RAISE
47   WRITE PERKSDS   UPDATE       . * UPDATE THE FILE
48 *
49 REPORT  BEFORE-UPDATE     LINESIZE 80
50   SEQUENCE  EMP#
51   TITLE     1 'REGION 1 EMPLOYEES GROSS SALARIES BEFORE UPDATE'
52   LINE      1 EMP#  NAME-LAST  NAME-FIRST  PAY-GROSS
53 *
54 *
55 JOB   INPUT PERKSDS
56     IF  REGION = 1
57         PRINT  AFTER-UPDATE        . * SHOW UPDATED SALARIES
58     END-IF
59 *
60 REPORT  AFTER-UPDATE       LINESIZE 80
61   SEQUENCE  EMP#
62   TITLE     1 'REGION 1 EMPLOYEES GROSS SALARIES AFTER UPDATE'
63   LINE      1 EMP#  NAME-LAST  NAME-FIRST  PAY-GROSS
```
### REGION 1 EMPLOYEES GROSS SALARIES BEFORE UPDATE

<table>
<thead>
<tr>
<th>EMPLOYEE NUMBER</th>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>GROSS PAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00370</td>
<td>NAGLE</td>
<td>MARY</td>
<td>554.40</td>
</tr>
<tr>
<td>01963</td>
<td>ARNOLD</td>
<td>LINDA</td>
<td>445.50</td>
</tr>
<tr>
<td>02200</td>
<td>BRANDOW</td>
<td>LYDIA</td>
<td>804.64</td>
</tr>
<tr>
<td>02688</td>
<td>CORNING</td>
<td>GEORGE</td>
<td>146.16</td>
</tr>
<tr>
<td>11357</td>
<td>LARSON</td>
<td>RODNEY</td>
<td>283.92</td>
</tr>
<tr>
<td>11467</td>
<td>BYER</td>
<td>JULIE</td>
<td>396.68</td>
</tr>
<tr>
<td>11473</td>
<td>BERG</td>
<td>NANCY</td>
<td>759.20</td>
</tr>
<tr>
<td>11602</td>
<td>MANHART</td>
<td>VIRGINIA</td>
<td>344.80</td>
</tr>
<tr>
<td>11931</td>
<td>TALL</td>
<td>ELAINE</td>
<td>492.26</td>
</tr>
<tr>
<td>12267</td>
<td>WIMN</td>
<td>GLORIA</td>
<td>373.60</td>
</tr>
</tbody>
</table>

### REGION 1 EMPLOYEES GROSS SALARIES AFTER UPDATE

<table>
<thead>
<tr>
<th>EMPLOYEE NUMBER</th>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>GROSS PAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>00370</td>
<td>NAGLE</td>
<td>MARY</td>
<td>582.12</td>
</tr>
<tr>
<td>01963</td>
<td>ARNOLD</td>
<td>LINDA</td>
<td>467.77</td>
</tr>
<tr>
<td>02200</td>
<td>BRANDOW</td>
<td>LYDIA</td>
<td>844.87</td>
</tr>
<tr>
<td>02688</td>
<td>CORNING</td>
<td>GEORGE</td>
<td>153.46</td>
</tr>
<tr>
<td>11357</td>
<td>LARSON</td>
<td>RODNEY</td>
<td>298.11</td>
</tr>
<tr>
<td>11467</td>
<td>BYER</td>
<td>JULIE</td>
<td>416.51</td>
</tr>
<tr>
<td>11473</td>
<td>BERG</td>
<td>NANCY</td>
<td>797.16</td>
</tr>
<tr>
<td>11602</td>
<td>MANHART</td>
<td>VIRGINIA</td>
<td>362.84</td>
</tr>
<tr>
<td>11931</td>
<td>TALL</td>
<td>ELAINE</td>
<td>516.87</td>
</tr>
<tr>
<td>12267</td>
<td>WIMN</td>
<td>GLORIA</td>
<td>392.28</td>
</tr>
</tbody>
</table>
Sequentially Reading VSAM File through Non-unique Alternate Index

This example reads employee records looking for a specified department. The POINT command is used against the department path to the KSDS. The STATUS parameter of the POINT command detects invalid department codes. The DO WHILE construct provides a simple method of reading records sequentially until the department code changes or we reach end-of-file.

1 *
2 * EXAMPLE 16.9
3 *
4 FILE DEPTCOD F 80
5  DEPARTMENT 1 3 N
6 *
7 FILE PTHKSDS VS
8 %PERSNL
9 *
10 JOB
11 point PTHKSDS EQ DEPARTMENT STATUS
12 IF PTHKSDS:FILE-STATUS NE 0
13 DISPLAY 'NO EMPLOYEES IN DEPARTMENT' DEPARTMENT
14 GOTO JOB
15 END-IF
16 *
17 GET PTHKSDS . * GET FIRST DEPT RECORD
18 DO WHILE PTHKSDS AND DEPARTMENT = DEPT . * LOOP WHILE VALID DEPT
19 PRINT SELECTED-DEPARTMENTS . * PRINT REPORT
20 GET PTHKSDS . * GET NEXT RECORD IN DEPT
21 END-DO
22 *
23 REPORT SELECTED-DEPARTMENTS LINESIZE 80
24 CONTROL DEPT NOPRINT
25 TITLE 1 'SELECTED DEPARTMENTS VIA VSAM PATH PROCESSING'
26 LINE 1 DEPT EMP# NAME-LAST NAME-FIRST
Updating a VSAM ESDS File

This example updates the VSAM ESDS file. The file is read sequentially and each employee in Region 2 is given a 10 percent cost-of-living raise. VSAM provides the ability to update an ESDS record in place as long as the record length does not change. A STOP is issued when the region code is greater than 2 to avoid unnecessary processing.

```
1 *
2 *    EXAMPLE 16.10
3 *
4 FILE    PERESDS   VS(ES UPDATE)
5 %PERSNL
35 *
36   NEW-GROSS   W 4 P 2 HEADING('NEW' 'GROSS' 'SALARY')
37 *
38 JOB
39 IF  REGION > 2
40 STOP                    . * STOP IF PAST RECORDS OF INTEREST
42 END-IF
43 IF  REGION NE 2
44 GOTO JOB                . * IGNORE IF NOT DESIRED REGION
46 END-IF
47 *
48 NEW-GROSS = 1.1 * PAY-GROSS . * CALCULATE NEW GROSS SALARY
50 PRINT   AUDIT-REPORT . * PRINT AUDIT REPORT
52 PAY-GROSS = NEW-GROSS . * UPDATE GROSS SALARY
54 WRITE   PERESDS UPDATE . * UPDATE PERSONNEL FILE
56 *
57 REPORT   AUDIT-REPORT     LINESIZE 80
58 SEQUENCE BRANCH EMP#
59 TITLE     1 'AUDIT REPORT -- REGION 2 SALARY CHANGE ACTIVITY'
60 LINE      1 BRANCH EMP# NAME-LAST NAME-FIRST PAY-GROSS NEW-GROSS
```
Updating a VSAM ESDS File

61 *
62 * VALIDATE PREVIOUS UPDATE
63 JOB
64 IF REGION > 2
65 STOP . * STOP IF PAST RECORDS OF INTEREST
66 END-IF
67 IF REGION NE 2
68 GOTO JOB . * IGNORE IF NOT DESIRED REGION
69 END-IF
70 *
71 PRINT VALIDATION . * PRINT VALIDATION
72 *
73 REPORT VALIDATION LINESIZE 80
74 SEQUENCE BRANCH EMP#
75 TITLE 1 'UPDATE VALIDATION -- REGION 2 SALARY CHANGE'
76 LINE 1 BRANCH EMP# NAME-LAST NAME-FIRST PAY-GROSS

11/25/83 AUDIT REPORT -- REGION 2 SALARY CHANGE ACTIVITY PAGE 1

   NEW
BRANCH EMPLOYEE LAST FIRST GROSS GROSS
   NUMBER NAME NAME PAY SALARY
 01  11376  HUSS  PATTI  360.80  396.88
 02  03571  KRUSE  MAX  242.40  266.64
 02  11710  POWELL  CAROL  243.20  267.52
 03  00577  PETRIK  KATHY  220.80  242.88
 03  02765  DENNING  RALPH  135.85  149.43
 03  03416  FORREST  BILL  13.80  15.18
 03  04234  MCMAHON BARBARA  386.40  425.04
 04  00445  POST  JEAN  292.00  321.20
 05  01895  VETTER  DENISE  279.36  307.29
 05  04225  LOYAL  NED  295.20  324.72

-------------------------------------------------------------------------

11/25/83 UPDATE VALIDATION -- REGION 2 SALARY CHANGE PAGE 1

   NEW
BRANCH EMPLOYEE LAST FIRST GROSS
   NUMBER NAME NAME PAY
 01  11376  HUSS  PATTI  396.88
 02  03571  KRUSE  MAX  266.64
 02  11710  POWELL  CAROL  267.52
 03  00577  PETRIK  KATHY  242.88
 03  02765  DENNING  RALPH  149.43
 03  03416  FORREST  BILL  15.18
 03  04234  MCMAHON BARBARA  425.04
 04  00445  POST  JEAN  321.20
 05  01895  VETTER  DENISE  307.29
 05  04225  LOYAL  NED  324.72

Deleting and Adding Records of VSAM KDS File

This example shows how to delete and add records to a VSAM KDS. The employee numbers of three employees must be changed. Since the file is keyed on the employee number, each record must be deleted and then added again with the new employee number. The automatic input file contains the transaction records.
In the first job activity, each record to be deleted is read. If it is not present, an error message is issued. Otherwise, the employee number is changed, a copy is placed in a sequential work file for later addition, the original record is deleted, and an audit report is generated.

In the second job activity, the records written to the work file are read by automatic input. A READ is performed to make sure no existing record in the file has the new employee number; if so, an error message is issued. Otherwise, the employee record with the new employee number is added to the file. A READ is issued to validate the ADD; this is not done typically, but is done here to show that the ADD actually occurred. An audit report is output.

```
1 *  
2 * EXAMPLE 16.11  
3 *  
4 FILE TRANS F 80  
5 OLD-EMP# 15 N  
6 NEW-EMP# 75 N  
7 *  
8 FILE PERKSDS VS(PASSWORD 'UPDPER' UPDATE)  
9 %PERSNL  
39 *  
40 FILE PERSTMP F 150 VIRTUAL . * TEMP HOLD FILE FOR RECORDS  
42 TMP-EMP# 95 N  
43 *  
44 JOB  
45 READ PERKSDS KEY OLD-EMP# STATUS  
46 IF PERKSDS:FILE-STATUS NE 0  
47 DISPLAY 'INVALID EMPLOYEE NUMBER ' OLD-EMP#  
48 GOTO JOB . * TRY NEXT TRANSACTION  
50 END-IF  
51 *  
52 TMP-EMP# = NEW-EMP# . * SET NEW EMP# INTO RECORD  
54 PUT PERSTMP FROM PERKSDS . * SAVE RECORD FOR SECOND JOB  
56 WRITE PERKSDS DELETE . * DELETE THE OLD RECORD  
58 PRINT AUDIT-REPORT . * PRINT THE AUDIT  
60 *  
61 REPORT AUDIT-REPORT LINESIZE 80  
62 TITLE 1 'AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY'  
63 TITLE 2 'FIRST PASS -- DELETED RECORDS'  
64 LINE 1 OLD-EMP# NAME-LAST NAME-FIRST  
65 *  
66 *  
67 JOB INPUT PERSTMP  
68 READ PERKSDS KEY TMP-EMP# STATUS . * VALIDATE KEY  
70 IF PERKSDS:FILE-STATUS EQ 0  
71 DISPLAY 'EMP# ALREADY ON FILE - ' TMP-EMP#  
72 GOTO JOB  
73 END-IF  
74 WRITE PERKSDS ADD FROM PERSTMP . * WRITE NEW RECORD  
76 READ PERKSDS KEY TMP-EMP# . * VALIDATE WRITE (OPTIONAL)  
78 PRINT AUDIT-REPORT . * PRINT AUDIT REPORT  
80 *  
81 REPORT AUDIT-REPORT LINESIZE 80  
82 TITLE 1 'AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY'  
83 TITLE 2 'SECOND PASS -- ADD RECORDS'  
84 LINE 1 TEMP-EMP# NAME-LAST NAME-FIRST
```
GETDATE Macro

CA-Easytrieve/Plus provides a macro facility to ease the burden of repetitive coding. Typically, a macro might contain file and field definitions and be used as a data dictionary. This technique is demonstrated in almost every example in this guide. Another use of the macro facility is to store functions, such as field reformatting, and report definitions, as illustrated in Examples 16.12A and 16.12B (shown below).

Example 16.12A is a macro to obtain the system date and strip out the slashes. It was also used in Example 15.14 (as shown in Chapter 15, “Basic Examples”). The macro is defined with a single positional parameter -- the receiving field. All other required fields are defined within the macro.

* EXAMPLE 16.12A

MACRO USER-DATE
*
* GET THE CURRENT DATE AND PUT INTO USER FIELD LESS SLASHES
*
DEFINE GETDATE-DATE W 8 A
DEFINE GETDATE-FIRST6 GETDATE-DATE 6 N
DEFINE GETDATE-LAST5 GETDATE-DATE +3 5 A
DEFINE GETDATE-LAST6 GETDATE-DATE +2 6 A
DEFINE GETDATE-LAST3 GETDATE-DATE +5 3 A
DEFINE GETDATE-LAST2 GETDATE-DATE +6 2 A
GETDATE-DATE = SYSDATE . * MOVE ALL 8
GETDATE-LAST3 = GETDATE-LAST2 . * SHIFT LEFT OVER NEXT /
GETDATE-LAST6 = GETDATE-LAST5 . * SHIFT LEFT OVER FIRST /
&USER-DATE = GETDATE-FIRST6 . * MOVE TO USER FIELD
Example 16.12B shows how the macro is invoked. A LIST NOMACROS statement is in the job to inhibit expanding the generated macro code.

1 *
2 *  EXAMPLE 16.12B
3 *
4 *
5 * THIS EXAMPLE WILL GET THE SYSTEM DATE AND STRIP THE SLASHES FROM IT
6 * THE FORMAT IS
7 *
8 * %GETDATE SYSTEM-DATE
9 *
10 * WHERE SYSTEM-DATE IS THE FIELD TO PUT THE CONVERTED DATE
11 *
12 JOB INPUT NULL
13 DEFINE SYSTEM-DATE W 4 P
14 %GETDATE SYSTEM-DATE
15 DISPLAY NEWPAGE SYSDATE ' WAS CONVERTED TO ' SYSTEM-DATE
16 STOP

-----------------------------------------------------------------------
11/24/83 WAS CONVERTED TO 112483

CONCAT Macro

This is an example of a macro which concatenates two fields into one, with variable spacing between fields. Following is the macro definition:

*  EXAMPLE 16.13A
*
MACRO RECEIVE SPACE PART2
*
* CONCAT MACRO
* FORMAT:
*   %CONCAT RECEIVE N PART2
* IT PERFORMS AS:
*   RECEIVE = RECEIVE +SPC(N) + PART2
*
DEFINE CONCAT-HOLD W 254 A
DEFINE CONCAT-SCAN CONCAT-HOLD 1 A INDEX CONCAT-NDX
DEFINE CONCAT-LENGTH W 2 P 0
CONCAT-NDX = 253 *. START AT END
CONCAT-HOLD = &RECEIVE *. COPY PART1
DO WHILE CONCAT-SCAN EQ ' ' AND CONCAT-NDX GE 0
   CONCAT-NDX = CONCAT-NDX - 1 *. FIND 1ST #BLANK
END-DO
CONCAT-NDX = CONCAT-NDX + 1 + &SPACE *. DO SPACES FACTOR
CONCAT-LENGTH = 253 - CONCAT-NDX
MOVE &PART2 TO CONCAT-SCAN CONCAT-LENGTH *. MOVE PART 2 AFTER
&RECEIVE = CONCAT-HOLD *. GIVE BACK TO USER
The CONCAT macro is exercised by the following CA-Easytrieve/Plus job:

1 *  
2 * EXAMPLE 16.13B  
3 *  
4 *  
5 * THIS EXAMPLE WILL CONCATENATE TWO FIELDS TOGETHER  
6 * THE FORMAT OF THE MACRO IS:  
7 *  
8 *    %CONCAT PART1 N PART2  
9 *  
10 *      WHERE N IS THE NUMBER OF SPACES TO BE INSERTED BETWEEN  
11 *      THE TWO PARTS  
12 *  
13 *      THE FIRST PARAMETER (PART1) IS USED AS THE RECEIVING FIELD  
14 *      THE THIRD PARAMETER (PART2) MAY BE A LITERAL BUT  
15 *      ENSURE THAT THE PROPER SYNTAX RULES ARE FOLLOWED FOR PASSING  
16 *      QUOTES (IE TO PASS A COMMA, ' , ', ' ' WOULD BE USED)  
17 *  
18 JOB INPUT NULL  
19 DEFINE WHOLE-THING W 40 A  
20 DEFINE SECOND-PART W 40 A  
21 DEFINE PART1 W 10 A VALUE 'SEE HOW'  
22 DEFINE PART2 W 10 A VALUE 'IT PUTS'  
23 DEFINE PART3 W 10 A VALUE 'THE PARTS'  
24 DEFINE PART4 W 10 A VALUE 'TOGETH'  
25 DEFINE PART5 W 10 A VALUE 'ER'  
26 WHOLE-THING = PART1  
27 SECOND-PART = PART2  
28 DISPLAY NEWPAGE WHOLE-THING  
29 PERFORM CONCAT-SPACE  
30 DISPLAY WHOLE-THING  
31 SECOND-PART = PART3  
32 PERFORM CONCAT-SPACE  
33 DISPLAY WHOLE-THING  
34 SECOND-PART = PART4  
35 PERFORM CONCAT-SPACE  
36 DISPLAY WHOLE-THING  
37 SECOND-PART = PART5  
38 PERFORM CONCAT-NOSPACE  
39 DISPLAY WHOLE-THING  
40 STOP  
41 CONCAT-SPACE. PROC  
42 *  
43 %CONCAT WHOLE-THING 1 SECOND-PART  
44 *  
45 END-PROC  
46 *  
47 CONCAT-NOSPACE. PROC  
48 *  
49 %CONCAT WHOLE-THING 0 SECOND-PART  
50 *  
51 END-PROC  

------------------------------------------------------------------------  
SEE HOW  
SEE HOW IT PUTS  
SEE HOW IT PUTS THE PARTS  
SEE HOW IT PUTS THE PARTS TOGETHER  
SEE HOW IT PUTS THE PARTS TOGETHER
Processing JCL Parameters

This example obtains a parameter coded on a JCL EXEC statement and uses it to control the selection of records.

A START procedure receives control at the beginning of the job. This procedure calls a subprogram which moves the PARM information into a CA-Easytrieve/Plus field. A JCL parameter can be from 1 to 100 characters long. The maximum length of the field you defined (PARM-DATA in this case) must be specified in PARM-LTH. The subprogram, EZTPX01, moves up to 10 characters from the JCL PARM-DATA, in this example.

If no JCL parameter is specified on the EXEC statement, PARM-LTH includes a zero after calling EZTPX01. If more than 10 characters are specified, the PARM is truncated to 10 characters.

In this example, the expected JCL parameter is defined to be three digits; the first digit specifies the region number and the next two contain the branch number. The format and content of the parameter is validated after calling EZTPX01. A PARM-LTH of 10 was used to detect either a missing JCL parameter or one which is not the correct length. If PARM-LTH = 3 was coded, CA-Easytrieve/Plus could not detect a parameter which was longer than 3, since EZTPX01 would truncate it to three characters.

Note: The use of field redefinition defines subfields of the JCL parameter. The value of the JCL parameter is 302 for this example.

The EZTPX01 routine is distributed as part of the CA-Easytrieve/Plus system.

```
1 * 
2 *   EXAMPLE 16.14 
3 * 
4 FILE PERSNL FB(150 1800) 
5 %PERSNL 
35 * 
36 *   JCL PARM DECODE DEFINITIONS 
37 * 
38 PARM-INFO W 12 A 
39 PARM-LTH PARM-INFO 2 B 
40 PARM-DATA PARM-INFO +2 10 A 
41 * 
42 SELECT-REGION PARM-DATA 1 N 
43 SELECT-BRANCH PARM-DATA +1 2 N 
44 * 
45 * 
46 JOB START(PARM-ANALYSIS) 
47 IF REGION = SELECT-REGION AND BRANCH = SELECT-BRANCH 
48 PRINT 
49 END-IF 
50 * 
51 *   MOVE THE JCL PARM INTO THE PARM-INFO W-FIELD AND VALIDATE IT 
52 * 
53 *   THE JCL PARM MUST BE 3 DIGITS OF THE FORM RBB 
54 *   WHERE R IS THE REGION NUMBER AND BB IS THE BRANCH NUMBER 
55 * 
56 PARM-ANALYSIS. PROC 
```
PARM-LTH = 10
CALL EZTPX01 USING (PARM-REGISTER PARM-INFO)
IF PARM-LTH NE 3
  DISPLAY SKIP 3 '******* MISSING OR INVALID JCL PARM'
  STOP EXECUTE
END-IF
IF SELECT-REGION NE 1 THRU 4
  DISPLAY SKIP 3 '******* INVALID REGION NUMBER'
  STOP EXECUTE
END-IF
IF SELECT-BRANCH NE 1 THRU 5
  DISPLAY SKIP 3 '******* INVALID BRANCH NUMBER'
  STOP EXECUTE
END-IF
END-PROC
*
REPORT LINESIZE 70
TITLE 'EMPLOYEES IN REGION' -2 SELECT-REGION -
  'BRANCH' -2 SELECT-BRANCH
LINE NAME-FIRST NAME-LAST EMP#

3/22/84        EMPLOYEES IN REGION 3   BRANCH 02      PAGE      1

  FIRST     LAST     EMPLOYEE
  NAME      NAME      NUMBER

  JANICE     THOMPSON  01743
  CINDY      SMOTH     01730
  RUTH       ISAAC     12641
  LORRIE     LACH      09609
  LESLIE     GRECO     07231
  WILLIAM    REYNOLDS  05805
Introduction

The BANK system is a combination of online and batch processing which illustrates the adaptability of CA-Easytrieve/Plus to a wide range of environments, and demonstrates a variety of coding techniques. This system is a sample application for a BANK, but could be adapted to other applications with minor modifications. However, implementation of a system as sophisticated as this requires considerable knowledge of generalized program development and substantial experience in data processing. If your professional expertise lies primarily in other fields, you do need help from your data center to undertake this task.

A common need is to have ready access to data; that is, the ability to quickly locate and modify specific items. This requires the means to search a file for a certain record or set of records and, once located, to update the record. This ready access is typically called online processing. Online processing offers speed and flexibility advantages over batch processing. When you are working online you can access any record, look at it on a terminal, and perform any function based on the values in the record at that time. A batch job requires you to specify in advance not only which records to access, but what functions to perform on them.

Online access is most appropriate for accessing a small number of records and performing varied activities, such as deleting one record, changing the name on another, and adding a middle initial to a third. A batch job is more efficient to list all records from a file, or to read all records from a file and to choose certain individuals to receive a letter.

The BANK system encompasses these two types:
- Online processing to access specific records
- Batch processing to produce the reports.
Online Processing

The online portion of this system runs under TSO, CMS, or ICCF. The discussion in this chapter assumes TSO. The data is stored on a key-sequenced VSAM data set (KSDS) created by running the IDCAMS utility with the following commands:

```
DEFINE CLUSTER (NAME ('your.bank.masterfile') -
  KEYS (6 0) -
  REUSE -
  VOLUMES (volser) -
  RECORDSIZE (200 200) -
  RECORDS (50 50 )
```

This group of statements defines the file to the system and must be followed by an initialization job to prepare the file for processing, as illustrated in Example 17.1 below.

**Initialize Customer File**

```
1 *
2 * EXAMPLE 17.1
3 *
4 %BANKLIB
45 *
46   NEXT-ID    7    6     N
47 *
48 JOB INPUT NULL
49 *
50 *  THIS JOB Initializes THE CUSTOMER FILE
51 *  IT WRITES THE FIRST RECORD ON THE FILE
52 *  WITH THE NEXT AVAILABLE CUST-ID OF 1
53 *
54 CUST-ID  = 0  . * SET RECORD 0
56 NEXT-ID  = 1  . * SET NEXT RECORD TO 1
58 PUT    CUST    . * ADD THE RECORD TO THE FILE
60 STOP    . * ALL DONE
```

As previously mentioned, the master file used is a key-sequenced data set whose key is a derived field - the record number. Therefore, the first record added to the file has a key of 000001, the second 000002, and so forth.
The record number (key) of the next available record on the file is stored in field
'NEXT-ID' of record zero. This field is set to a value of 000001 by the initialization
job. As data is added to the file by subsequent processing, this field is updated.
The key field is the first six bytes of each record. The other fields are:

- PERS-TITLE (i.e. MR, MRS, MISS, MS)
- FIRST-NAME
- LAST-NAME
- MIDDLE-INITIAL
- ADDRESS LINE 1
- ADDRESS LINE 2
- CITY
- STATE
- ZIP
- LOCAL INDICATOR (YES, NO)
- CREDIT RATING (0-9)
- SAVINGS ACCOUNT (YES, NO)
- CHECKING (YES, NO)
- SAFE DEPOSIT (YES, NO)
- C AND D (YES, NO)
- ALL SAVERS (YES, NO)
- VISA (YES, NO)
- MASTER CARD (YES, NO)
- MONEY MARKET (YES, NO)
- TREASURY BILL (YES, NO)

Once the IDCAMS and initialization jobs are run, you can begin adding data to
the file.

The first step is to log onto TSO and create the Job Control Setup command list
(CLIST), as follows:

- BANK CLIST FOR TSO
  FREE F(SYSIN SYSPRINT PANDD1 CUST TERMIN EZTVFM)
  ALLOC F(PANDD1) DA('your macro library') SHR
  ALLOC F(SYSPRINT) DA(*)
  ALLOC F(TERMIN) DA(*)
  ALLOC F(CUST) DA('your bank masterfile') SHR
  ALLOC F(SYSIN) DA(*)
  ALLOC F(EZTVFM) SP(2,2) CYL
  CALL 'your program library(EZTPA00)'

The next step is to type:

EXEC BANK

The system introduces itself and prepares to accept commands. The valid
commands and their descriptions are:

ADD

This command is used to add a new record to a file. After you enter the ADD
keyword, the system displays the attributes (number, name, length, and type) of
each field of the next available record, in order by field number, and asks you to
enter a value for that field.
The field type codes are:

A = alphabetic - only letters allowed
N = numeric - only numbers allowed
X = mixed - any characters allowed
Q = (yes,no) - only Y or N allowed

When you have entered data into the last field of the new record, the system displays its record number (key) as a customer ID.

**BROWSE**

This command enables you to specify desired values in up to 20 fields of a record. After these field values are specified, the system reads the entire file and displays those records, one at a time, with fields to match your specification.

To search for a field, enter the field number. The system displays the attributes of that field and asks you to enter a value. When you finish entering the search values for one record, you can enter NEXT to begin entering search values for the next record, or END to terminate the search and begin the BROWSE activity.

Only those records that have all fields equal to the search data can be displayed. Repeat END to terminate BROWSE. The last record read is the current record that can be used by DISPLAY, DEL, or UPD.

**DEL**

This command deletes a record. After you enter the DEL keyword, the system displays the ID of the current record and asks if this is the one you want.

- If you enter YES, that record is deleted.
- If you enter NO, the system requests the ID of the desired record, and deletes the record you specify.

**DISPLAY**

This command displays the fields of the current record or any other record you specify. After you enter the DISPLAY keyword, the system displays the ID of the current record and asks if this is the one you want.

- If you answer YES, the fields of that record are displayed.
- If you answer NO, the system requests the ID of the desired record and displays the fields of the record you specify.

**ECHO**

This command causes all terminal input to be displayed on the terminal.
END

This command terminates the current activity; when used on the primary command line it terminates the session. When in BROWSE or UPDate processing, this command terminates the field value specification phase and begins the BROWSE or UPDate activity.

ESC

This command is valid at anytime and returns you to the major command entry of the session. It is useful when you are in the middle of processing a command which you do not want to continue.

NOECHO

This command inhibits the display of terminal input.

UPD

This command enables you to enter UPDate mode. If a record is active from a previous activity, its ID is displayed and the system asks if this is the one you want. If you enter NO, the system requests the ID of the desired record. Once that record is found, the system requests the field number of the field to be updated. That field's attributes are displayed and the system requests new data.

This process of specifying a field number and entering new data continues until you enter END. The system then inquires whether you are ready to update.

- If you answer NO, you return to specifying field numbers and entering new data.
- If you answer YES, the record is updated.

The CA-Easytrieve/Plus coding required for the BANK system and the sample terminal session using this system are illustrated in Example 17.2, as shown in the next two exhibits. The first page of this example provides a listing of the macro %BANKLIB, which is used throughout the BANK system.
MACRO 0 VSAMOPT 'UPDATE'
* COMMON TABLE FILE TO DEFINE ACCOUNT TYPES
* 
FILE ACCTNAME  TABLE   INSTREAM
ARG 1 2 N
DESC 4 30 A
01 SAVINGS
02 CHECKING
03 SAFE DEPOSIT
04 CERTIFICATE OF DEPOSIT
05 ALL SAVERS
06 VISA
07 MASTER CARD
08 MONEY MARKET
09 TREASURY BILL
ENDTABLE
* BANK CUSTOMER FILE DESCRIPTION
* 
FILE CUST  VS(F &VSAMOPT)  WORKAREA 200
* 
CUST-ID 1 6 N
NAME 7 25 A
PERS-TITLE 7 4 A
FIRST-NAME 11 10 A
LAST-NAME 21 15 A
MIDDLE-INITIAL 36 1 A
ADDRESS1 38 25 A
ADDRESS2 63 25 A
CITY 88 15 A
STATE 103 2 A
ZIP 105 9 A
LOCAL 114 1 A
CREDIT-RATING 115 1 A
- HEADING ('CREDIT' 'RATING')
ACCT-DATA 116 9 A
SAVINGS 116 1 N 0
CHECKING 117 1 N 0
SAFE-DEPOSIT 118 1 N 0
- HEADING ('SAF' 'DEPOSIT')
C-AND-D 119 1 N 0
ALL-SAVERS 120 1 N 0
- HEADING ('ALL' 'SAVERS')
VISA 121 1 N 0
MONEY-MARKET 123 1 N 0
- HEADING ('MONEY' 'MARKET')
MONEY-MARKET 123 1 N 0
- HEADING ('MONEY' 'MARKET')
T-BILL 124 1 N 0
ACCT-IND ACCT-DATA 1 N 0
- OCCURS 9 INDEX ACCT-NDX
* ACCT-MAX W 2 P 0 VALUE 9
CUST-KEY W 6 N
*
Bank File Program

1 *
2 * BANK EXAMPLE 17.2 BANK FILE UPDATE TERMINAL SESSION
3 *
4 %BANKLIB
45 CUST-AREA 1 200 A
46 CUST-FIELD 1 1 A INDEX CUST-NDX
47 NEXT-ID 7 6 N
48 FILE TERMIN WORKAREA 80
49 TERM-REC 1 80 A -
50 INDEX TERM-BEGIN-NDX . * INPUT TERMINAL RECORD
51 TERM-BEGIN-CH 1 1 A -
52 INDEX TERM-BEGIN-NDX . * LEADING SPACE SCAN CHAR
53 TERM-END-CH 1 1 A -
54 INDEX TERM-END-NDX . * TRAILING SPACE SCAN CHAR
55 FIRST3 TERM-REC 3 A
56 CMD-IN TERM-REC 10 A
57 KEY-IN TERM-REC 6 N . * INPUT CUST-ID
58 TERM-FIELD-NO TERM-REC 3 N . * INPUT FIELD NUMBER
59 TERM-LEN W 2 P . * LENGTH OF TERMINAL INPUT
60 NEXT-KEY W 6 N . * NEXT AVAILABLE KEY
61 *
62 * SWITCHS USED DURING PROCESSING
63 *
64 FIELD-ERR W 1 A . * NOT DONE WITH FIELD YET
65 ERROR W 1 A
66 CUST-ERR W 1 A
67 ECHO-SW W 1 A VALUE 'Y' . * INITIAL VALUE FOR ECHO
68 NUM-SW W 1 A . * INPUT HAS NUMBERS
69 ALPHA-SW W 1 A . * INPUT HAS ALPHAS
70 OTHER-SW W 1 A . * INPUT HAS NON(ALPHANUMERIC)
71 ANSWER W 1 A . * RESPONSE TO YES/NO
72 *
73 * COMMAND TABLES/INFORMATION
74 *
75 CUR-STATE W 1 N . * CURRENT STATUS
76 CMD-ID W 1 N
77 CMD-FND W 1 A . * CMD SWITCH
78 CMD-OVHD W 2 P VALUE 3 . * AMT TO ADD TO CMD-LEN
79 CMD-DEFAULT W 1 N
80 CMD-HOLD W 10 A
81 *
82 * THE FOLLOWING TABLE OF COMMANDS CONTAIN:
83 *
84 COMMAND-ID 1 BYTE
85 COMMAND-STATE 1 BYTE
86 COMMAND-LENGTH 1 BYTE
87 COMMAND-NAME 7 BYTES
88 *
89 THIS TABLE IS LOOPED THRU TO VALIDATE AND DETERMINE WHAT THE
90 USER REQUEST IS
91 A NEW COMMAND CAN BE ADDED BEFORE THE LAST COMMAND WHICH
92 MUST HAVE A CMD-LEN OF 0
93 MAINLINE CODE PERFORMS THE ROUTINE DEPENDING ON COMMAND-ID
94 *
Online Processing

100 CMD-TBL W 100 A VALUE -
   '003END+ 107D15PLAY+ 283ADD+ 303UPD+
   403DEL+ 504ECHO+ 606NOECHO+ 706BROWSE+
   073END+ 174NEXT+ 000'   . * END OF COMMAND TABLE
110 CMD-DATA CMD-TBL 20 A -
   INDEX CMD-NDX
111 CMD-NO CMD-DATA 1 N
112 CMD-STATE CMD-DATA +1 1 N
113 CMD-LEN CMD-DATA +2 1 N
114 CMD-NAME CMD-DATA +3 1 A
115 *
116 * THIS AREA IS USED TO MOVE THE CORRECT NUMBER OF BYTES FOR THE
117 * VALID COMMANDS TO PROMPT THE USER
118 *
119 DSP-LINE W 100 A
120 DSP-LIT DSP-LINE 22 A VALUE 'VALID COMMANDS ARE:ESC'
121 DSP-DATA DSP-LINE +22 78 A
122 DSP-COMMA DSP-DATA 1 A INDEX DSP-NDX
123 DSP-CMD DSP-DATA +2 10 A INDEX DSP-NDX
124 *
125 * THE NEXT AREA IS THE OBJECT OF THE FIELD TABLE LOOKUP
126 *
127 FIELD-DEFN W 72 A
128 FIELD-ID FIELD-DEFN 3 N
129 FIELD-DATA FIELD-DEFN +4 68 A
130 FIELD-NAME FIELD-DATA 20 A
131 FIELD-LOC FIELD-DATA +20 3 N
132 FIELD-LEN FIELD-DATA +24 2 N
133 FIELD-TYPE FIELD-DATA +27 1 A
134 FIELD-MAX W 3 P VALUE 20 . * MAX NUMBER OF FIELDS
135 FIXED-FIELD W 40 A . * DISPLAY AREA FOR FIELD
136 COMPARE-FIELD W 40 A . * HOLD AREA FOR FIELD
138 QUERY-MAX W 3 P VALUE 20 . * 20 QUERYS
139 QUERY-MAX +1 1 N
140 QUERY-INC W 3 P VALUE 43 . * AMT TO INC
141 QUERY-INC +1 1 N
142 QUERY-CNT W 3 P . * COUNTER FOR LOOPING
143 QUERY-CNT +1 1 N
144 QUERY-TABLE W 43 A -
   OCCURS 21 INDEX QUERY-NDX . * TABLE OF USER QUERIES
145 *
148 * PLUS ONE DUMMY
149 QUERY-FIELD QUERY-TABLE 3 N . * FIELD NO FOR QUERY
150 COMAPRE-FIELD QUERY-TABLE +3 40 A . * QUERY
151 *
154 * THE NEXT TABLE CONTAINS ALL VALID FIELDS TO BE ENTERED
155 * IT ALSO CONTAINS THE BEGINNING, LENGTH, TYPE, AND EDIT RULES
156 *
157 FILE FIELDTBL TABLE INSTREAM
158 ARG 1 3 N
159 DESC 4 68 A
160 001 TITLE 007 04 A
161 002 FIRST NAME 011 10 A
162 LAST NAME 021 15 A
163 MIDDLE INITIAL 036 01 A
164 ADDRESS LINE 1 038 25 X
165 ADDRESS LINE 2 063 25 X
166 CITY 088 15 A
167 STATE 103 02 A
168 ZIP 165 09 X
169 IS IT LOCAL 114 01 Q
170 CREDIT RATING 115 01 N
**Online Processing**

```
012 SAVINGS  116 01 Q
013 CHECKING  117 01 Q
014 SAFE DEPOSIT BOX  118 01 Q
015 C AND D  119 01 Q
016 ALL SAVERS  120 01 Q
017 VISA  121 01 Q
018 MASTER CARD  122 01 Q
019 MONEY MARKET  123 01 Q
020 T BILL  124 01 Q
ENDTABLE

161 JOB INPUT NULL START HELLO FINISH END-OF-JOB
162 *
163 * THE MAINLINE LOGIC FOLLOWS:
164 *
165 * A COMMAND IS OBTAINED
166 * THE COMMAND ID CORRESPONDING TO THE COMMAND IS RETURNED
167 * THE ROUTINE TO HANDLE THE REQUEST IS PERFORMED
168 *
169 CUR-STATE = 0                   . * SET TO FIRST LEVEL
170 CMD-DEFAULT = 0                  . * NO DEFAULT HERE
171 PERFORM GET-COMMAND
172 IF CMD-ID = 0                     . * END
173     STOP
174 END-IF
175 IF CMD-ID = 1                      . * DISPLAY
176     PERFORM SELECT-RECORD         . * POSITION
177     PERFORM DISPLAY-RECORD
178     GOTO JOB
179 END-IF
180 IF CMD-ID = 2                      . * ADD
181     PERFORM ADD-NEW-RECORD
182     GOTO JOB
183 END-IF
184 IF CMD-ID = 3                      . * UPD
185     PERFORM SELECT-RECORD         . * POSITION
186     PERFORM UPDATE-RECORD          . * GET AND ACCEPT UPDATES
187     GOTO JOB
188 END-IF
189 IF CMD-ID = 4                      . * DELETE
190     PERFORM SELECT-RECORD         . * POSITION
191     PERFORM DELETE-RECORD         . * DELETE
192     GOTO JOB
193 END-IF
194 IF CMD-ID = 5                      . * ECHO
195     ECHO-SW = 'Y'                  . * SET ECHOSW
196     GOTO JOB
197 END-IF
198 IF CMD-ID = 6                      . * NOECHO
199     ECHO-SW = 'N'                  . * SET OFF
200     GOTO JOB
201 END-IF
202 IF CMD-ID = 7                      . * BROWSE
203     PERFORM BROWSE-FILE
204     GOTO JOB
205 END-IF
206 *
207 ADD-NEW-RECORD. PROC
208 *
209 ADD A NEW CUSTOMER
210 * FOR EACH FIELD ON THE FILE GET THE VALUE FROM THE USER
211 * WHEN FINISHED, ADD TO THE FILE AND RETURN
212 *
213 CUST-KEY = 0
214 PERFORM READ-CUST
215 CUST-AREA = '
216 FIELD-ID = 1
```
Online Processing

235   DO WHILE FIELD-ID LE FIELD-MAX
236       PERFORM GET-FIELD
237       FIELD-ID = FIELD-ID + 1
238   END-DO
239   CUST-ID = NEXT-KEY
240   CUST-KEY = NEXT-KEY . * SAVE FOR CURRENT RECORD TEST
241   WRITE CUST ADD STATUS
242   PERFORM CUST-FILE-TEST
243   DISPLAY 'CUSTOMER ' CUST-ID ' ADDED'
244   NEXT-KEY = NEXT-KEY + 1
245   END-PROC

247 *
248 BROWSE-FILE. PROC
250 *
251 * THIS ROUTINE WILL ASK FOR THE FIELDS BE THE SEARCH FIELDS
252 * AND ASK FOR THE VALUES DESIRED
253 * ALL RECORDS WITH THE DESIRED VALUE IN THE SELECTED FIELDS
254 * WILL BE LISTED UNTIL END OF FILE OR 'END' REQUESTED
255 *
256   CUST-KEY = 0 . * RESET CURRENT RECORD
257   PERFORM READ-CUST . * HAVE TO GET IT
258   DISPLAY 'YOU CAN ENTER UP TO ' - QUERY-MAX ' QUERIES FOR THE SEARCH'
259   DISPLAY 'THE QUERIES WILL BE + ANDED TOGETHER'
260   QUERY-NDX = QUERY-MAX * QUERY-INC . * POINT TO AFTER LAST
261   QUERY-INC = 1 . * BACK TO BEGINNING
262   QUERY-NDX = 0 . * BACK TO BEGINNING
263   DO WHILE QUERY-CNT LE QUERY-MAX
264       DISPLAY 'ENTER FIELD NUMBER FOR SEARCH ' - 'END' WHEN DONE'
265       PERFORM GET-FIELD-NUMBER
266       IF FIRST3 = 'END'
267       THEN
268           QUERY-FIELD = 0
269           COMPARE-FIELD = ''
270           GOTO QUERY-DONE
271       ELSE
272           QUERY-FIELD = TERM-FIELD-NO
273           FIELD-ID = QUERY-FIELD
274           PERFORM GET-FIELD
275           MOVE TERM-REC TERM-LEN TO COMPARE-FIELD
276           COMPARE-QUERY = ' ' . * SET TO 999
277           QUERY-MAX = 999 . * BACK TO BEGINNING
278           QUERY-MAX = 999 . * BACK TO BEGINNING
279       END-IF
280   END-DO
281   IF QUERY-NDX GT QUERY-MAX
282       DISPLAY 'MAXIMUM OF ' - QUERY-MAX ' QUERIES REACHED'
283       QUERY-DONE
284   ELSE
285       QUERY-NDX = 0 . * SCAN ENTIRE TABLE
286       DO WHILE QUERY-NDX LE QUERY-MAX
287           DISPLAY 'MAXIMUM OF ' + QUERY-MAX ' QUERIES REACHED'
288   END-IF
289   QUERY-DONE
290   CUST-KEY = 1
291   POINT CUST GE CUST-KEY STATUS
292   PERFORM CUST-FILE-TEST
293   PERFORM NEXT-RECORD
294   CUR-STATE = CMD-ID . * SUBCOMMANDS OF BROWSE
295   CMD-DEFAULT = 1 . * NEXT IS DEFAULT
296   DO WHILE CUST . * READ ALL RECORDS
297       QUERY-NX = 0 . * SCAN ENTIRE TABLE
298       QUERY-CNT = 1 . * SCAN ENTIRE TABLE
299   DO WHILE QUERY-CNT LE QUERY-MAX AND QUERY-FIELD GT 0 . * DO UNTIL END
300       SEARCH FIELD-TBL WITH QUERY-FIELD GIVING FIELD-ATTRIBUTES
301       QUERY-NX = 0 . * RETURN TO QUERY
302       QUERY-INC = 1 . * SCAN ENTIRE TABLE
303       QUERY-CNT = 1 . * SCAN ENTIRE TABLE
304       QUERY-DONE
305   END-DO
306   CUST-NX = FIELD-LOC - 1
307   MOVE CUST-FIELD FIELD-LEN TO COMPARE-FIELD
Online Processing

310    IF COMPARE-QUERY EQ COMPARE-FIELD -
       OR COMPARE-QUERY SPACE
311       QUERY-NDX = QUERY-NDX + QUERY-INC
312       QUERY-CNT = QUERY-CNT + 1
313     ELSE
314       GOTO NOT-WANTED
315   END-IF
316   END-DO
317   DISPLAY 'CUST-ID:' CUST-ID
318   DISPLAY ' ' PERFORM DISPLAY-RECORD
319   PERFORM GET-COMMAND
320   NOT-WANTED
321     IF CMD-ID NE 0
322       PERFORM NEXT-RECORD
323     ELSE
324       GOTO BROWSE-END
325     END-IF
326     END-DO
327   BROWSE-END
328     DISPLAY 'BROWSE COMPLETE'
329 END-PROC
330 *
331 CONVERT-TO-YES-NO. PROC
334 *
335   IF FIELD-TYPE NE 'Q'
336     GOTO CONVERT-DONE
337   END-IF
338   IF CUST-FIELD EQ '0'
339     CUST-FIELD EQ 'N'
340   ELSE
341     CUST-FIELD EQ 'Y'
342   END-IF
343 CONVERT-DONE
344 END-PROC
345 *
346 CUST-FILE-TEST. PROC
348 *
349   IF CUST:FILE-STATUS NE 0
350     DISPLAY '*******FILE ERROR:CUST-----STATUS=' CUST:FILE-STATUS
351     STOP
352 END-IF
353 END-PROC
354 *
355 DELETE-RECORD. PROC
357 *
358   THIS ROUTINE WILL DELETE THE CURRENT RECORD
359   THE FILE MUST HAVE BEEN PREVIOUSLY POSITIONED
360   AT THE DESIRED RECORD
361 WRITE CUST DELETE STATUS
362 PERFORM CUST-FILE-TEST
363 CUST-KEY = 0
364 DISPLAY 'RECORD ' CUST-ID ' DELETED'
365 END-PROC
366 *
367 DISPLAY-FIELD. PROC
369 *
370   DISPLAY A FIELD IN THE CURRENT RECORD
371   THE FIELD NUMBER WILL BE THAT OF 'FIELD-ID'
372 FIELD-ERR = 'N'
373 SEARCH FIELDDBL WITH FIELD-ID GIVING FIELD-DATA
374 IF NOT FIELDDBL
FIELD-ERR = 'Y'
DISPLAY FIELD-ID ' NOT A VALID FIELD'
ELSE
CUST-NDX = FIELD-LOC - 1
PERFORM CONVERT-TO-YES-NO
MOVE CUST-FIELD FIELD-LEN TO FIXED-FIELD
DISPLAY 'FIELD:' FIELD-ID +1 FIELD-NAME +1 - FIXED-FIELD
END-IF
END-PROC
*
DISPLAY-RECORD. PROC
*
DISPLAY ALL FIELDS OF THE CURRENT RECORD
*
FIELD-ID = 1
DO WHILE FIELD-ID LE FIELD-MAX
PERFORM DISPLAY-FIELD
FIELD-ID = FIELD-ID + 1
END-DO
END-PROC
*
EDIT-ALPHA. PROC
*
DISPLAY ERROR MSG IF NOT ALPHA
*
IF NUM-SW = 'Y' OR OTHER-SW = 'Y'
FIELD-ERR = 'Y'
DISPLAY 'FIELD MUST CONTAIN ONLY LETTERS'
END-IF
END-PROC
*
EDIT-NUM. PROC
*
DISPLAY ERROR MSG IF NOT NUMERIC
*
IF ALPHA-SW = 'Y' OR OTHER-SW = 'Y'
FIELD-ERR = 'Y'
DISPLAY 'FIELD MUST CONTAIN ONLY NUMBERS'
END-IF
END-PROC
*
EDIT-YES-NO. PROC
*
IF TERM-BEGIN-CH EQ 'N'
TERM-BEGIN-CH = '0'       . * CONVERT TO 0
ELSE IF TERM-BEGIN-CH EQ 'Y'
TERM-BEGIN-CH = '1'       . * CONVERT TO 1
END-IF
IF TERM-BEGIN-CH NE '0' '1'
DISPLAY 'FIELD MUST BE ''Y'' OR ''N'' '
FIELD-ERR = 'Y'
END-IF
END-PROC
*
END-OF-JOB. PROC
*
ALL DONE, READ RECORD 0 TO GET PREVIOUS NEXT KEY
IF CURRENT NEXT KEY IS GT PREVIOUS THEN UPDATE RECORD 0
INDICATE END OF SESSION
CUST-KEY = 0
READ CUST KEY CUST-KEY
PERFORM CUST-FILE-TEST
IF NEXT-ID NE NEXT-KEY
NEXT-ID = NEXT-KEY
451     WRITE CUST UPDATE
452     PERFORM CUST-FILE-TEST
453     END-IF
454     DISPLAY 'DATA ENTRY SESSION COMPLETE'
455     END-PROC
456 *
457     FIELD-EDIT-MOVE. PROC
458 *
459 *     EDIT THE INPUT DATA, IF VALID MOVE TO OUTPUT
460 * ELSE RETURN WITH FIELD-ERR='Y'
461 * *
462 *     FIELD-ERR = 'N'
463     IF FIELD-LEN LT TERM-LEN
464     FIELD-ERR = 'Y'
465     DISPLAY 'INPUT CANNOT BE LONGER THAN:' FIELD-LEN
466     GOTO FIELD-EDIT-END
467     END-IF
468     IF FIELD-TYPE = 'A'
469     PERFORM EDIT-ALPHA
470     END-IF
471     IF FIELD-TYPE = 'N'
472     PERFORM EDIT-NUM
473     END-IF
474     IF FIELD-TYPE = 'Q'
475     PERFORM EDIT-YES-NO
476     END-IF
477     IF FIELD-ERR = 'N'
478     CUST-NDX = FIELD-LOC - 1
479     MOVE TERM-REC TERM-LEN TO CUST-FIELD FIELD-LEN
480     END-IF
481     END-PROC
482     FIELD-EDIT-END. END-PROC
484 *
485     FIND-CUST. PROC
487 *
488 *     GET A CUSTOMER ID, READ THE RECORD
489 * DO UNTIL VALID RECORD FOUND
490 * *
491     CUST-ERR = 'Y'
492     DO WHILE CUST-ERR EQ 'Y'
493     DISPLAY 'ENTER CUSTOMER ID'
494     PERFORM GET-LINE
495     IF KEY-IN NOT NUMERIC OR KEY-IN ZERO
496       DISPLAY 'INVALID CUSTOMER ID, RE-ENTER'
497     ELSE
498       CUST-KEY = KEY-IN
499       PERFORM READ-CUST
500     END-IF
501     END-DO
502     END-PROC
503 *
504     GET-COMMAND. PROC
506 *
507 *     DISPLAY AVAILABLE COMMANDS AND GET THE COMMAND FROM THE USER
508 * CONTINUE UNTIL VALID COMMAND ENTERED
509 * *
510     CMD-FND = 'N'
511     DO WHILE CMD-FND = 'N'
512     DSP-DATA = '
513     MOVE ZERO TO DSP-NDX CMD-NDX
514     DO WHILE CMD-LEN GT 0
515     IF CMD-STATE = CUR-STATE
516       DSP-COMMA = '.
517       MOVE CMD-NAME CMD-LEN TO DSP-CMD CMD-LEN
518       DSP-NDX = DSP-NDX + CMD-LEN + 2
519     END-IF
520     CMD-NDX = CMD-NDX + CMD-LEN + CMD-OVHD
521       END-DO
522       DISPLAY DSP-LINE
523       DISPLAY 'ENTER COMMAND'
524       PERFORM GET-LINE
525       IF CMD-IN SPACE AND CMD-DEFAULT GT 0
526         CMD-ID = CMD-DEFAULT
527         CMD-FND = 'Y'
528         GOTO NOT-FOUND
529       END-IF
530       CMD-NDX = 0
531       MOVE CMD-NAME CMD-LEN TO CMD-HOLD
532       DO WHILE CMD-HOLD NE CMD-IN OR CMD-STATE NE CUR-STATE -
533         OR TERM-LEN NE CMD-LEN
534         IF CMD-LEN EQ 0 . * ZERO LEN IS END OF TABLE
535         DISPLAY 'INVALID COMMAND:' CMD-IN
536         GOTO NOT-FOUND
537         ELSE
538         CMD-NDX = CMD-NDX + CMD-LEN + CMD-OVHD
539         MOVE CMD-NAME CMD-LEN TO CMD-HOLD
540       END-IF
541       END-DO
542       CMD-FND = 'Y'
543       CMD-ID = CMD-NO
544     NOT-FOUND
545     END-DO
546 END-PROC
547 *
548 GET-FIELD. PROC
549 *
550 * GET ONE FIELD FROM THE USER FIELD NUMBER IS IN 'FIELD-ID'
551 * LOOP UNTIL A VALID FIELD IS ENTERED
552 * LOOP UNTIL A VALID FIELD IS ENTERED
553 *
554       SEARCH FIELDTBL WITH FIELD-ID GIVING FIELD-DATA
555       FIELD-ERR = 'Y'
556       IF NOT FIELDTBL
557         DISPLAY 'ERROR INVALID FIELD NUMBER'
558         GOTO GET-FIELD-ERR
559       END-IF
560       DO WHILE FIELD-ERR = 'Y'
561         DISPLAY 'ENTER VALUE FOR FIELD:' FIELD-ID +1 FIELD-NAME -
562         '(' FIELD-TYPE ' ' FIELD-LEN ' CHARACTERS)'
563         PERFORM GET-LINE
564         PERFORM FIELD-EDIT-MOVE
565       END-DO
566 GET-FIELD-ERR
567 END-PROC
568 *
569 GET-FIELD-NUMBER. PROC
570 *
571 * THIS ROUTINE ASKS FOR A NUMBER OF A FIELD
572 * LOOP UNTIL VALID NUMBER ENTERED
573 *
574 GET-FIELD-LOOP
575 PERFORM GET-LINE
576 IF FIRST3 = 'END'
577     GOTO GOT-FIELD-NUMBER
578 END-IF
579 IF TERM-LEN NE 3 OR TERM-FIELD-NO NOT NUMERIC
580     DISPLAY 'NUMBER MUST BE 3 DIGITS NUMERIC'
581     GOTO GET-FIELD-LOOP
582 END-IF
583 GOT-FIELD-NUMBER
584 END-PROC
585 *
586 GET-LINE. PROC
587 *
* GET A LINE OF USER INPUT
590 * LEFT JUSTIFY, TRUNCATE SPACES ON RIGHT
591 * SET SWITCHES IF ALPHA OR NUMERIC OR OTHER DATA IS ENTERED
592 *
593 TERM-BEGIN-NDX = 0
594 MOVE SPACE TO ALPHA-SW NUM-SW OTHER-SW TERM-REC
595 DISPLAY ' '                        . * PROMPT IN SYNC
597 GET TERMIN
598 IF EOF TERMIN
599 DISPLAY 'USER REQUESTED ABORT'
600 STOP
601 END-IF
602 IF TERM-REC SPACE
603 TERM-LEN = 1
604 GOTO GOT-LINE
605 END-IF
606 IF ECHO-SW = 'Y'
607 DISPLAY TERM-REC
608 END-IF
609 * SUPPRESS LEADING BLANKS
610 DO WHILE TERM-BEGIN-NDX LS 79 -
611 TERM-BEGIN-CH = ' '       TERM-BEGIN-NDX = TERM-BEGIN-NDX + 1
612 END-DO
613                                  * HANDLE MASTER ESCAPE COMMAND
614 IF FIRST3 = 'ESC'
615 GOTO JOB
616 END-IF
617                                  * TRUNCATE TRAILING BLANKS
618 TERM-END-NDX = 79
619 DO WHILE TERM-END-NDX GT TERM-BEGIN-NDX -
620 TERM-END-CH = ' '       TERM-END-NDX = TERM-END-NDX - 1
621 END-DO
622                                  * SET NUMERIC, ALPHA, OTHER SWITCHES
623 TERM-LEN = TERM-END-NDX - TERM-BEGIN-NDX + 1
624 DO WHILE TERM-END-CH GE TERM-BEGIN-NDX
625 IF TERM-END-CH NUMERIC
626   NUM-SW = 'Y'
627 END-IF
628 IF TERM-END-CH ALPHABETIC
629   ALPHA-SW = 'Y'
630 END-IF
631 IF TERM-END-CH NOT NUMERIC AND TERM-END-CH NOT ALPHABETIC
632   OTHER-SW = 'Y'
633 END-IF
634 TERM-END-NDX = TERM-END-NDX - 1
635 END-DO
636 GOT-LINE
637 END-PROC
638 *
639 GET-YES-NO. PROC
640 *
641 * GET A YES OR NO ANSWER
642 *
643 ANSWER = ' '
644 DO WHILE ANSWER EQ ''
645 PERFORM GET-LINE
646 IF TERM-REC EQ 'Y' 'N' 'NO' 'YES'
647 MOVE TERM-REC 1 TO ANSWER
648 ELSE
649 DISPLAY 'VALID ANSWER IS YES OR NO'
650 DISPLAY 'ENTER ANSWER'
651 END-IF
652 END-DO
653 END-PROC
HELI. PROC

START PROCEDURE, INTRODUCE YOURSELF, READ RECORD 0 TO GET NEXT FREE KEY

TEST FOR EXISTANCE OF NEXT RECORD, IF IT IS THERE THEN SOME ERROR OCCURED DURING THE LAST DATA ENTRY SESSION THEN FIND NEXT AVAILABLE SLOT

DISPLAY NEWPAGE 'HELLO, WELCOME TO BRILLIG BANKS DATA ENTRY SYSTEM'

CUST-KEY = 0

READ CUST KEY CUST-KEY STATUS

PERFORM CUST-FILE-TEST

NEXT-KEY = NEXT-ID

POINT CUST GE NEXT-KEY . * TEST FOR ERROR

IF NOT EOF CUST

DISPLAY 'INITIALIZATION ERROR, READING TILL FREE RECORD'

END-IF

DO WHILE NOT EOF CUST

DISPLAY 'CUSTOMER ID:' NEXT-KEY ' FOUND, SKIPPING TO NEXT'

POINT CUST GE NEXT-KEY . * LAST UPDATE SESSION

END-DO

END-PROC

NEXT-RECORD. PROC

GET CUST STATUS

PERFORM CUST-FILE-TEST

IF CUST

CUST-KEY = CUST-ID

ELSE

CUST-KEY = 0

END-IF

END-PROC

READ-CUST. PROC

READ ONE RECORD, KEY:CUST-KEY

READ CUST KEY CUST-KEY STATUS

IF CUST

CUST-ERR = 'N'

ELSE

DISPLAY 'ERROR, CUSTOMER NOT FOUND:' CUST-KEY

END-IF

END-PROC

SELECT-RECORD. PROC

ANSWER = 'N'

IF CUST-KEY GT 0

DISPLAY 'CURRENT CUSTOMER IS:' CUST-KEY

DISPLAY 'IS THIS THE DESIRED RECORD(Y/N)?'

PERFORM GET-YES-NO

END-IF

IF ANSWER = 'Y'

PERFORM READ-CUST

ELSE

PERFORM FIND-CUST

END-IF

END-PROC

UPDATE-RECORD. PROC

THIS RECORD WILL UPDATE THE FIELDS OF THE CURRENT RECORD THE USER SELECTS, BY FIELD NUMBER, THE FIELD TO UPDATE
729 * THEN ENTERS THE NEW DATA
730 * THIS CONTINUES UNTIL 'END' IS ENCOUNTERED
731 *
732 UPDATE-LOOP
733 DISPLAY 'ENTER FIELD NUMBER TO ALTER DATA, (IE 001)'
734 DISPLAY 'ENTER ''END'' WHEN READY TO UPDATE'
735 PERFORM GET-FIELD-NUMBER
736 IF FIRST3 NE 'END'
737 FIELD-ID = TERM-FIELD-NO
738 PERFORM GET-FIELD
739 GOTO UPDATE-LOOP
740 END-IF
741 DISPLAY 'READY TO UPDATE RECORD(Y/N)'
742 PERFORM GET-YES-NO
743 IF ANSWER = 'N'
744 GOTO UPDATE-LOOP
745 END-IF
746 WRITE CUST UPDATE STATUS
747 PERFORM CUST-FILE-TEST
748 DISPLAY 'RECORD ' CUST-ID ' SUCCESSFULLY UPDATED'
749 END-PROC

HELLO, WELCOME TO BRILLIG BANKS DATA ENTRY SYSTEM
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ADD
ENTER VALUE FOR FIELD:001 TITLE (A,04 CHARACTERS)
MR
ENTER VALUE FOR FIELD:002 FIRST NAME (A,10 CHARACTERS)
TOM
ENTER VALUE FOR FIELD:003 LAST NAME (A,15 CHARACTERS)
LEONARD
ENTER VALUE FOR FIELD:004 MIDDLE INITIAL (A,01 CHARACTERS)
ENTER VALUE FOR FIELD:005 ADDRESS LINE 1 (X,25 CHARACTERS)
1781 ORANGE PLACE
ENTER VALUE FOR FIELD:006 ADDRESS LINE 2 (X,25 CHARACTERS)
ENTER VALUE FOR FIELD:007 CITY (A,15 CHARACTERS)
CALCULAS
ENTER VALUE FOR FIELD:008 STATE (A,02 CHARACTERS)
CA
ENTER VALUE FOR FIELD:009 ZIP (X,09 CHARACTERS)
31095
ENTER VALUE FOR FIELD:010 IS IT LOCAL (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:011 CREDIT RATING (N,01 CHARACTERS)
S
ENTER VALUE FOR FIELD:012 SAVINGS (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:013 CHECKING (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:014  SAFE DEPOSIT BOX  (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:015  C AND D  (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:016  ALL SAVERS  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:017  VISA  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:018  MASTER CARD  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:019  MONEY MARKET  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:020  T BILL  (Q,01 CHARACTERS)
N
CUSTOMER 000001 ADDED
VALID COMMANDS ARE: ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND
ADD
ENTER VALUE FOR FIELD:001  TITLE  (A,04 CHARACTERS)
MISS
ENTER VALUE FOR FIELD:002  FIRST NAME  (A,10 CHARACTERS)
JANE
ENTER VALUE FOR FIELD:003  LAST NAME  (A,15 CHARACTERS)
NEARY
ENTER VALUE FOR FIELD:004  MIDDLE INITIAL  (A,01 CHARACTERS)
I
ENTER VALUE FOR FIELD:005  ADDRESS LINE 1  (X,25 CHARACTERS)
887 DETOUR PLACE
ENTER VALUE FOR FIELD:006  ADDRESS LINE 2  (X,25 CHARACTERS)
ENTER VALUE FOR FIELD:007  CITY  (A,15 CHARACTERS)
MARLBOROUGH
ENTER VALUE FOR FIELD:008  STATE  (A,02 CHARACTERS)
VA
ENTER VALUE FOR FIELD:009  ZIP  (X,09 CHARACTERS)
22211
ENTER VALUE FOR FIELD:010  IS IT LOCAL  (Q,01 CHARACTERS)
N
ENTER VALUE FOR FIELD:011  CREDIT RATING  (N,01 CHARACTERS)
8
ENTER VALUE FOR FIELD:012  SAVINGS  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:013  CHECKING  (Q,01 CHARACTERS)
Y
ENTER VALUE FOR FIELD:014  SAFE DEPOSIT BOX  (Q,01 CHARACTERS)  Y
ENTER VALUE FOR FIELD:015  C AND D             (Q,01 CHARACTERS)  N
ENTER VALUE FOR FIELD:016  ALL SAVERS          (Q,01 CHARACTERS)  Y
ENTER VALUE FOR FIELD:017  VISA                (Q,01 CHARACTERS)  N
ENTER VALUE FOR FIELD:018  MASTER CARD         (Q,01 CHARACTERS)  N
ENTER VALUE FOR FIELD:019  MONEY MARKET        (Q,01 CHARACTERS)  N
ENTER VALUE FOR FIELD:020  T BILL              (Q,01 CHARACTERS)  Y

CUSTOMER 000002 ADDED
VALID COMMANDS ARE: ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND
BROWSE
YOU CAN ENTER UP TO 20 QUERIES FOR THE SEARCH
THE QUERIES WILL BE ‘ANDED’ TOGETHER
ENTER FIELD NUMBER FOR SEARCH ‘END’ WHEN DONE

002
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)
TOM
ENTER FIELD NUMBER FOR SEARCH ‘END’ WHEN DONE

001
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)
MR
ENTER FIELD NUMBER FOR SEARCH ‘END’ WHEN DONE

END

CUST-ID:000001

FIELD:001  TITLE               MR
FIELD:002  FIRST NAME          TOM
FIELD:003  LAST NAME           LEONARD
FIELD:004  MIDDLE INITIAL      
FIELD:005  ADDRESS LINE 1      1781 ORANGE PLACE
FIELD:006  ADDRESS LINE 2      
FIELD:007  CITY                CALCULAS
FIELD:008  STATE               CA
FIELD:009  ZIP                 31895
FIELD:010  IS IT LOCAL         N
FIELD:011  CREDIT RATING       5
FIELD:012  SAVINGS             N
FIELD:013  CHECKING            N
FIELD:014  SAFE DEPOSIT BOX    N
FIELD:015  C AND D             N
FIELD:016  ALL SAVERS          Y
FIELD:017  VISA                Y
FIELD:018  MASTER CARD         Y
FIELD:019  MONEY MARKET        Y
FIELD:020  T BILL              N
VALID COMMANDS ARE: ESC, END, NEXT
ENTER COMMAND

END
BROWSE COMPLETE
VALID COMMANDS ARE: ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ADD
ENTER VALUE FOR FIELD: 001 TITLE (A, 04 CHARACTERS)
MR
ENTER VALUE FOR FIELD: 002 FIRST NAME (A, 10 CHARACTERS)
KIP
ENTER VALUE FOR FIELD: 003 LAST NAME (A, 15 CHARACTERS)
LING
ENTER VALUE FOR FIELD: 004 MIDDLE INITIAL (A, 01 CHARACTERS)
ENTER VALUE FOR FIELD: 005 ADDRESS LINE 1 (X, 25 CHARACTERS)
227 BEETLE LN
ENTER VALUE FOR FIELD: 006 ADDRESS LINE 2 (X, 25 CHARACTERS)
ENTER VALUE FOR FIELD: 007 CITY (A, 15 CHARACTERS)
PALMER
ENTER VALUE FOR FIELD: 008 STATE (A, 02 CHARACTERS)
MA
ENTER VALUE FOR FIELD: 009 ZIP (X, 09 CHARACTERS)
01072
ENTER VALUE FOR FIELD: 010 IS IT LOCAL (Q, 01 CHARACTERS)
N
ENTER VALUE FOR FIELD: 011 CREDIT RATING (N, 01 CHARACTERS)
1
ENTER VALUE FOR FIELD: 012 SAVINGS (Q, 01 CHARACTERS)
Y
ENTER VALUE FOR FIELD: 013 CHECKING (Q, 01 CHARACTERS)
Y
ENTER VALUE FOR FIELD: 014 SAFE DEPOSIT BOX (Q, 01 CHARACTERS)
N
ENTER VALUE FOR FIELD: 015 C AND D (Q, 01 CHARACTERS)
N
ENTER VALUE FOR FIELD: 016 ALL SAVERS (Q, 01 CHARACTERS)
Y
ENTER VALUE FOR FIELD: 017 VISA (Q, 01 CHARACTERS)
N
ENTER VALUE FOR FIELD: 018 MASTER CARD (Q, 01 CHARACTERS)
N
ENTER VALUE FOR FIELD: 019 MONEY MARKET (Q, 01 CHARACTERS)
N
CUSTOMER 000003 ADDED
VALID COMMANDS ARE: ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

DISPLAY
CURRENT CUSTOMER IS: 000003
IS THIS THE DESIRED RECORD (Y/N)?

Y

FIELD: 001 TITLE MR
FIELD: 002 FIRST NAME KIP
FIELD: 003 LAST NAME LING
FIELD: 004 MIDDLE INITIAL
FIELD: 005 ADDRESS LINE 1 227 BEETLE LN
FIELD: 006 ADDRESS LINE 2
FIELD: 007 CITY PALMER
FIELD: 008 STATE MA
FIELD: 009 ZIP 01072
FIELD: 010 IS IT LOCAL N
FIELD: 011 CREDIT RATING 1
FIELD: 012 SAVINGS Y
FIELD: 013 CHECKING Y
FIELD: 014 SAFE DEPOSIT BOX N
FIELD: 015 C AND D N
FIELD: 016 ALL SAVERS Y
FIELD: 017 VISA N
FIELD: 018 MASTER CARD N
FIELD: 019 MONEY MARKET N
FIELD: 020 T BILL Y

VALID COMMANDS ARE: ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

UPD
CURRENT CUSTOMER IS: 000003
IS THIS THE DESIRED RECORD (Y/N)?

Y

ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

001
ENTER VALUE FOR FIELD: 001 TITLE

MS

ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

002
ENTER VALUE FOR FIELD: 002 FIRST NAME

LOIS

ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

END
READY TO UPDATE RECORD (Y/N)?

Y

RECORD 000003 SUCCESSFULLY UPDATED
Batch Processing

The efficient batch processing capabilities of CA-Easytrieve/Plus enable you to use the data you have entered into the file for many purposes. Several examples are presented here.

Detail Report

One practical use for this file is to output a list of all bank customers with a list of their individual accounts. Example 17.3, as shown in the next exhibit, illustrates the CA-Easytrieve/Plus coding and the resulting report. In this example, the START parameter of the JOB statement tells CA-Easytrieve/Plus to execute the specified procedure (HELLO) after opening all the files, but before reading the first record. The HELLO procedure positions the file to the next record after record zero, which contains no data, only the key of the next available record in this file. This enables all subsequent program logic to operate only on records containing valid data.

Each record is searched for a 1 in the fields which contain the account names - field numbers 012 through 020. If it is found, that account name is stored into a working storage field named ACCT-WORK, which eventually is printed on the report along with the corresponding customer name. In the REPORT group, the CONTROL command provides a total of the number of accounts for each customer. The NOPRINT option suppresses printing the total line for the FIRST-NAME control break.
* EXAMPLE 17.3 DETAIL LISTING OF CUSTOMERS

%BANKLIB
ACCT-ID W 2 N
ACCT-WORK W 30 A

JOB START HELLO

ACCT-NDX = 0
DO WHILE ACCT-NDX LT ACCT-MAX
IF ACCT-IND = 1
ACCT-ID = ACCT-NDX + 1
SEARCH ACCTNAME WITH ACCT-ID GIVING ACCT-WORK
PRINT ACCT-DETAIL
END-IF
ACCT-NDX = ACCT-NDX + 1
END-DO
HELLO. PROC
CUST-KEY = 1
POINT CUST GE CUST-KEY
END-PROC
REPORT ACCT-DETAIL LINESIZE 80 SPACE 1
SEQUENCE LAST-NAME
CONTROL LAST-NAME FIRST-NAME NOPRINT
TITLE 'LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS'
HEADING ACCT-WORK ('ACCOUNT' 'NAME')
LINE LAST-NAME FIRST-NAME ACCT-WORK TALLY

ACCOUNT
LAST-NAME    FIRST-NAME              NAME              TALLY
DEMO            TOM        SAFE DEPOSIT
                CERTIFICATE OF DEPOSIT
                ALL SAVERS
                VISA
                MASTER CARD
                MONEY MARKET
DEMO
GOLFER          GOOD       MONEY MARKET
                TREASURY BILL
                MASTER CARD
                SAVINGS
                CHECKING
                SAFE DEPOSIT
                CERTIFICATE OF DEPOSIT
                ALL SAVERS
                VISA
GOLFER
HANDHOLDER      HANNA      SAVINGS
                CHECKING
                SAFE DEPOSIT
                CERTIFICATE OF DEPOSIT
                ALL SAVERS
                VISA
                MASTER CARD
                MONEY MARKET
HANDHOLDER
<table>
<thead>
<tr>
<th>Last-Name</th>
<th>First-Name</th>
<th>Account Name</th>
<th>Tally</th>
</tr>
</thead>
<tbody>
<tr>
<td>KIPPER</td>
<td>JON</td>
<td>SAVINGS</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL SAVERS</td>
<td></td>
</tr>
<tr>
<td>KIPPER</td>
<td>ROD</td>
<td>SAFE DEPOSIT</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>LENGTHY</td>
<td>ROGER</td>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CARD</td>
<td></td>
</tr>
<tr>
<td>LENGTHY</td>
<td></td>
<td>SAFE DEPOSIT</td>
<td>4</td>
</tr>
<tr>
<td>LING</td>
<td>LOIS</td>
<td>TREASURY BILL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVINGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL SAVERS</td>
<td></td>
</tr>
<tr>
<td>LONELY</td>
<td>ALICE</td>
<td>SAVINGS</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL SAVERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CARD</td>
<td></td>
</tr>
<tr>
<td>LAST-NAME</td>
<td>FIRST-NAME</td>
<td>ACCOUNT NAME</td>
<td>TALLY</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>LONG</td>
<td>HARRY</td>
<td>SAVINGS</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>LONG</td>
<td></td>
<td>SAFE DEPOSIT</td>
<td>3</td>
</tr>
<tr>
<td>MIDDLEMAN</td>
<td>WILLIAM</td>
<td>ALL SAVERS</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>MIDDLEMAN</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>NEARY</td>
<td>JANE</td>
<td>ALL SAVERS</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TREASURY BILL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVINGS</td>
<td></td>
</tr>
<tr>
<td>NICE</td>
<td>ANNE</td>
<td>MASTER CARD</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVINGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TREASURY BILL</td>
<td></td>
</tr>
<tr>
<td>NICE</td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td>RONALD</td>
<td>SAVINGS</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL SAVERS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TREASURY BILL</td>
<td></td>
</tr>
<tr>
<td>RIGHT</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>LAST-NAME</td>
<td>FIRST-NAME</td>
<td>ACCOUNT NAME</td>
<td>TALLY</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>ROY</td>
<td>ROLAND</td>
<td>ALL SAVERS MASTER CARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAVINGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECKING VISA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td>6</td>
</tr>
<tr>
<td>ROY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHORE</td>
<td>KAREN</td>
<td>ALL SAVERS SAVINGS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MASTER CARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET</td>
<td>4</td>
</tr>
<tr>
<td>SHORE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHORT</td>
<td>DAVE</td>
<td>CHECKING SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ALL SAVERS VISA MASTER CARD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONEY MARKET SAVINGS</td>
<td></td>
</tr>
<tr>
<td>SHORT</td>
<td>MARY</td>
<td>MASTER CARD MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VISA CERTIFICATE OF DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>SHORT</td>
<td></td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>SHWONT</td>
<td>WANDA</td>
<td>SAVINGS VISA MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CERTIFICATE OF DEPOSIT ALL SAVERS MASTER CARD</td>
<td></td>
</tr>
<tr>
<td>SHWONT</td>
<td></td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
Mass Mailing

Another use for the file developed from the previous DETAIL Report is the generation of a letter to all customers who do not have an ALL-SAVERS account, notifying them of the potential savings this account can provide. As shown in the next exhibit, Example 17.4 illustrates the coding required, and the letter and the labels.

The PERFORM statement executes the CONCAT-NAME procedure which links the title (MS, MR, or whatever) with the name. The IF statement tests the value of field ALL SAVERS as not-equal 1. If this tests true, the customer does not have an ALL-SAVERS account and the statements beneath the IF are executed. The mailing label and the letter are printed.

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>FIRST-NAME</th>
<th>ACCOUNT NAME</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEWART</td>
<td>ROD</td>
<td>VISA MASTER CARD CERTIFICATE OF DEPOSIT SAFE DEPOSIT MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td>STEWART</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>TRUMPET</td>
<td>JIM</td>
<td>SAVINGS CHECKING TREASURY BILL ALL SAVERS VISA MASTER CARD MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td>TRUMPET</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>VAN GOGH</td>
<td>VINCENT</td>
<td>TREASURY BILL VISA SAVINGS CHECKING</td>
<td></td>
</tr>
<tr>
<td>VAN GOGH</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>WILCHE</td>
<td>NANCY</td>
<td>MASTER CARD VISA CERTIFICATE OF DEPOSIT MONEY MARKET SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>WILCHE</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>YAZBASEY</td>
<td>CARL</td>
<td>SAVINGS MONEY MARKET CERTIFICATE OF DEPOSIT ALL SAVERS VISA CHECKING</td>
<td></td>
</tr>
<tr>
<td>YAZBASEY</td>
<td></td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>

```
12/02/83 LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>FIRST-NAME</th>
<th>ACCOUNT NAME</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEWART</td>
<td>ROD</td>
<td>VISA MASTER CARD CERTIFICATE OF DEPOSIT SAFE DEPOSIT MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td>STEWART</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>TRUMPET</td>
<td>JIM</td>
<td>SAVINGS CHECKING TREASURY BILL ALL SAVERS VISA MASTER CARD MONEY MARKET</td>
<td></td>
</tr>
<tr>
<td>TRUMPET</td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>VAN GOGH</td>
<td>VINCENT</td>
<td>TREASURY BILL VISA SAVINGS CHECKING</td>
<td></td>
</tr>
<tr>
<td>VAN GOGH</td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>WILCHE</td>
<td>NANCY</td>
<td>MASTER CARD VISA CERTIFICATE OF DEPOSIT MONEY MARKET SAFE DEPOSIT</td>
<td></td>
</tr>
<tr>
<td>WILCHE</td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>YAZBASEY</td>
<td>CARL</td>
<td>SAVINGS MONEY MARKET CERTIFICATE OF DEPOSIT ALL SAVERS VISA CHECKING</td>
<td></td>
</tr>
<tr>
<td>YAZBASEY</td>
<td></td>
<td></td>
<td>6</td>
</tr>
</tbody>
</table>
```
1 PARM LIST(NOPARM NOFILE)
2 *
3 * BANK EXAMPLE 17.4 MAILING EXAMPLE
4 *
5 %BANKLIB
46 FILE LBLSOUT PRINTER
47 FILE LETTERS PRINTER
48 WHOLE-NAME W 30 A
49 JOB      START HELLO
50 PERFORM CONCAT-NAME
51 IF ALL-SAVERS NE 1
52 PRINT MAILING-LABELS
53 PERFORM CONCAT-COMMA
54 PRINT ALL-SAVERS-LETTER
55 END-IF
56 CONCAT-NAME. PROC
58 WHOLE-NAME = PERS-TITLE
59 %CONCAT WHOLE-NAME 1 FIRST-NAME
66 %CONCAT WHOLE-NAME 1 MIDDLE-INITIAL
113 %CONCAT WHOLE-NAME 1 LAST-NAME
140 END-PROC
141 *
142 CONCAT-COMMA. PROC
144 %CONCAT WHOLE-NAME 0 ''','''
171 END-PROC
172 *
173 HELLO. PROC
175 CUST-KEY = 1
176 POINT CUST GE CUST-KEY
177 END-PROC
178 *
179 REPORT ALL-SAVERS-LETTER NOHEADING NOADJUST -
180   LINESIZE 80 PRINTER LETTERS
184 SEQUENCE ZIP
185 CONTROL CUST-ID NEWPAGE
188 LINE 1 'DEAR' WHOLE-NAME
189 LINE 3 +5 'The new ALL SAVERS CERTIFICATES are now available.'
190 LINE 4 +5 'They offer several advantages over other investments.'
191 LINE 5 +5 'We have set up a special telephone number to answer'
192 LINE 6 +5 'your questions regarding this new and exciting ONE TIME '
193 LINE 7 +5 'chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE'
194 LINE 8 +5 'during normal working hours and we will be happy to help'
195 LINE 9 +5 'you.'
196 LINE 11 +30 'SINCERELY,'
197 LINE 13 +30 'George Million'
198 LINE 14 +30 'Vice President'
199 LINE 15 +30 'Fourth National Bank of Virginia'
194 *
195 REPORT MAILING-LABELS PRINTER LBLSOUT LABELS (ACROSS 2)
196 SEQUENCE ZIP
197 LINE 1 WHOLE-NAME
198 LINE 2 ADDRESS1
199 LINE 3 ADDRESS2
200 LINE 4 CITY -2 STATE -2 ZIP
DEAR MISS MARY SHORT,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MR ROGER N LENGTHY,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MR ROD N LENGTHY,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia
DEAR MRS HARRY K LONG,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MRS NANCY WILCHE,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MS ANNE NICE,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia
DEAR MR VINCENT I VAN GOGH,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

------------------------------------------------------------------------

DEAR MR ROD STEWART,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

MISS MARY SHORT               MR ROD N LENGTHY
100 THIS PLACE                111 BOTTLE LN
APT 200B                      WILBRAHAM       NC 01072
BELCHERTOWN                   121 TOTTLE LN
                           WILBRAHAM       FL 01072
MR ROGER N LENGTHY            MRS HARRY K LONG
121 TOTTLE LN                 2000 CALCUTTA ST
WILBRAHAM                     SPRINGFIELD     VA 22152
MRS NANCY WILCHE              MS ANNE NICE
1006 ROUND CIRCLE             171 LEE HIGHWAY
NORTHAMPTON                   LEESBURG        VA 22672
MR VINCENT I VAN GOGH         MR ROD STEWART
1 ERIE AVE                    1 MAGGIE LANE
APT 1001                      OAK BROOK       IL 30072
                           CLARENTON       CA 50072
Summary Report

As shown in the next exhibit, Example 17.5 produces a summary of the customer accounts by location; in this case by state. Summary reports can have a variety of uses and they can be sequenced on any field.

This example also provides totals by state, through the CONTROL command. This summary report does not list customers by name, but simply counts the total number of accounts for each account type.

17/02/83

<table>
<thead>
<tr>
<th>STATE</th>
<th>SAVINGS</th>
<th>CHECKING</th>
<th>SAFE-DEPOSIT</th>
<th>C-AND-D</th>
<th>ALL-SAVERS</th>
<th>VISA</th>
<th>MASTER-CARD</th>
<th>MONEY-MARKET</th>
<th>T-BILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>DC</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>FL</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>IL</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MA</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>MD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MI</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NC</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NH</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NY</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>WA</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17 15 15 16 16 19 17 18 7
Chapter 18

Project Management System

Introduction

This chapter illustrates how CA-Easytrieve/Plus might be used for a classical data processing application: a Project Management System. The structure is typical of many data processing systems, and is composed of:

1. The Project Master file, defined in Example 18.1.
2. The File Maintenance Program, illustrated in Example 18.1. The output from the file update process is illustrated in Example 18.2.
3. Various report programs, illustrated in Examples 17.3, 17.4, and 17.5. The reports are generated using data from the updated Master File.

There are several reasons why Project Management was chosen as an example. However, the primary consideration is that it enables us to present a complete base to a working system. Although this system is somewhat simple compared to many Project Management Systems, it is usable for many applications.

Master File Layout

The master file is a physical sequential file containing fixed length records. There are two record types, the project record and the task record. A project definition is composed of a project record and optionally, one or more task records, as illustrated below:

```
project record for project 1
  task record for task 1 of project 1
  task record for task 2 of project 1
  task record for task 3 of project 1

project record for project 2
  task record for task 1 of project 2
  task record for task 2 of project 2

project record for project 3
  task record for task 1 of project 3
  task record for task 2 of project 3
```
The formats of the project and task records are the same - only the data is different. This combination enables us to represent many projects on one file, with each project having its own set of tasks, as illustrated below.

MACRO FILE-PARMS

* PROJECT MANAGEMENT SYSTEM -- MASTER FILE DEFINITION

FILE PRJIN FB (110 4400) &FILE-PARMS

* PROJ-REC 1 110 A
  PROJ-NO 1 5 A -
  HEADING ('PROJECT' 'NUMBER')
  TASK-NO 6 5 A -
  HEADING ('TASK' 'NUMBER')
  NAME 11 25 A
  MANAGER-ID 36 5 A -
  HEADING ('MANAGER' 'ID')

* STATISTICS 41 39 A

* ESTIMATED-DATA 41 12 A
  EST-MAN-HRS 41 4 P 1 -
  HEADING ('EST' 'HOURS')
  EST-START-DATE 45 4 P -
  HEADING ('EST' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
  EST-END-DATE 49 4 P -
  HEADING ('EST' 'END' 'DATE') MASK(BWZ 'Z99/99/99')

* ACTUAL-DATA 53 12 A
  ACT-MAN-HRS-SO-FAR 53 4 P 1 -
  HEADING ('ACTUAL' 'HOURS')
  ACT-START-DATE 57 4 P -
  HEADING ('ACTUAL' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
  ACT-END-DATE 61 4 P -
  HEADING ('ACTUAL' 'END' 'DATE') MASK(BWZ 'Z99/99/99')

* LAST-TRANSACTION-DATE 65 4 P -
  HEADING ('LAST' 'TRANS' 'DATE') MASK(BWZ 'Z99/99/99')
  LAST-ACTIVITY-HRS 69 4 P 1
  PCT-COMPLETE 73 2 P -
  HEADING ('PERCENT' 'COMPLETE')

PREDECESSOR-TABLE 75 7 A OCCURS 5 -
INDEX PRE-NDX
  PRE-NO 75 5 A
  PRE-PCT 80 2 P
  PRE-NO1 82 5 A
  PRE-PCT2 87 2 P
  PRE-NO3 89 5 A
  PRE-PCT3 94 2 P
  PRE-NO4 96 5 A
  PRE-PCT4 101 2 P
  PRE-NO5 103 5 A
  PRE-PCT5 108 2 P
Programs

There are two classes of programs which operate on the Project Master -- file maintenance and report generation.

File Maintenance

The file maintenance program is a classical update program, diagrammed below:

Transactions are matched against the Base Master File. The requested operation is performed (if valid), the Updated Master File is produced, and three reports are generated which indicate the success of the run:

1. Listing of Edited Transactions
2. Transaction Error Report
3. Change Register Report

These reports are illustrated in Examples 17.4 and 17.5.
Transaction Record Format

The transaction file consists of transactions to a particular project or task. The table below describes the format of the transaction record.

<table>
<thead>
<tr>
<th>Name</th>
<th>Position</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANS-ID</td>
<td>1-2</td>
<td>Determines what to do (add, delete, and so forth)</td>
</tr>
<tr>
<td>TRANS-PROJ</td>
<td>3-7</td>
<td>Project ID of related master</td>
</tr>
<tr>
<td>TRANS-TASK</td>
<td>8-12</td>
<td>Task ID of master (if blank, performs transaction against project record)</td>
</tr>
<tr>
<td>TRANS-FIELD</td>
<td>13-17</td>
<td>Field ID of the field to be accessed</td>
</tr>
<tr>
<td>TRANS-DATA</td>
<td>18-?</td>
<td>Actual data to add/update master fields</td>
</tr>
</tbody>
</table>

Transaction Codes

As you can see, each transaction refers not only to a particular project/task, but also to a particular field of that record. To add a new project, one transaction is needed for each field to be added. The table below lists the valid transactions.

<table>
<thead>
<tr>
<th>Transaction Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP,AT</td>
<td>ADD transaction; AP=add project, AT=add task</td>
</tr>
<tr>
<td>DP,DT</td>
<td>DELETE transaction for project, or task</td>
</tr>
<tr>
<td>CP,CT</td>
<td>Change transaction for project, or task</td>
</tr>
<tr>
<td>IT</td>
<td>Increment transaction to add to task field</td>
</tr>
</tbody>
</table>
Field Types

The system sorts the transactions in order of DELETES, ADDS, CHANGES, and INCREMENTS for a given project/task. Thus, it is possible to delete a project, then ADD it again all in one run. Data in the TRANS-DATA field (positions 18-?) is edited to correspond to the field it pertains to. The table below lists the different types of fields.

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Description</th>
<th>Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATES</td>
<td>date fields</td>
<td>18-23</td>
</tr>
<tr>
<td>PCT</td>
<td>percent fields</td>
<td>18-20</td>
</tr>
<tr>
<td>ID fields</td>
<td>project/task numbers, etc.</td>
<td>18-22</td>
</tr>
<tr>
<td>NAME</td>
<td>project/task name</td>
<td>18-42</td>
</tr>
<tr>
<td>HOURS*</td>
<td>hour fields</td>
<td>18-24 (1 decimal)</td>
</tr>
</tbody>
</table>

*To produce 123,456.7, enter 1234567 into positions 18-24.

Examples 18.1 and 18.2 illustrate two functions; building a new Master File, then updating it.

1. As shown earlier in the Transaction Record Format table, Example 18.1 lists the CA-Easytrieve/Plus code which both creates the new Master File, and subsequently updates it. The function performed by running the CA-Easytrieve/Plus program depends upon the items contained in the Transaction file, TRANSIN, which are identified by the codes illustrated earlier in the Transaction Codes table. To create the new Master File, the program is first run with TRANSIN containing only Project (AP) and Add Task (AT) items. The input file is listed at the end of the run in the Listing of Edited Transactions described in the Change Register Report at the end of this chapter. There is no Transaction Error Report generated for this run.

2. As shown earlier in the Transaction Codes table, Example 18.2 illustrates the output reports after the program is run a second time with new items in TRANSIN, which include change task transactions (CT), increment transactions (IT), add project (AP) and add task (AT) transactions. The new input data is listed in the Listing of Edited Transactions and described in the Change Register Report later in this chapter. A duplicate transaction is listed on the Transaction Error Report at the end of this chapter.
Project Status - EXAMPLE 18.1

```plaintext
1 * PROJECT MANAGEMENT SYSTEM EXAMPLE 18.1: ADD DATA TO FILE
2 *
3 PARM DEBUG(STATE FLOW FLDCHK) LIST(NOFILE NOPARM)
4 %PROJLIB
5 *
6 PROJECT MANAGEMENT SYSTEM -- MASTER FILE DEFINITION
7 *
8 FILE PRJIN  FB (110 4400)
9 *
10 PROJ-REC 1 110 A
11 PROJ-TASK 1 10 A
12 PROJ-NO 1 5 A -
13  HEADING ('PROJECT' 'NUMBER')
14 TASK-NO 6 5 A -
15  HEADING ('TASK' 'NUMBER')
16 NAME 11 25 A -
17  HEADING ('MANAGER' 'ID')
18 *
19 STATISTICS 41 39 A
20 *
21 ESTIMATED-DATA 41 12 A
22 EST-MAN-HRS 41 4 P 1 -
23  HEADING ('EST' 'HOURS')
24 EST-START-DATE 45 4 P -
25  HEADING ('EST' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
26 EST-END-DATE 49 4 P -
27  HEADING ('EST' 'END' 'DATE') MASK(BWZ 'Z99/99/99')
28 *
29 ACTUAL-DATA 53 12 A
30 ACT-MAN-HRS-SO-FAR 53 4 P 1 -
31  HEADING ('ACTUAL' 'HOURS')
32 ACT-START-DATE 57 4 P -
33  HEADING ('ACTUAL' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
34 ACT-END-DATE 61 4 P -
35  HEADING ('ACTUAL' 'END' 'DATE') MASK(BWZ 'Z99/99/99')
36 *
37 LAST-TRANSACTION-DATE 65 4 P -
38  HEADING ('LAST' 'TRANS' 'DATE') MASK(BWZ 'Z99/99/99')
39 LAST-ACTIVITY-HRS 69 4 P 1
40 PCT-COMPLETE 73 2 P -
41  HEADING ('PERCENT' 'COMPLETE')
42 PREDECESSOR-TABLE 75 7 A OCCURS 5 -
43  INDEX PRE-NDX
44 *
45 PRE-NO PREDECESSOR-TABLE 5 A
46 PRE-PCT PREDECESSOR-TABLE 2 P
47 PRE-N01 75 5 A
48 PRE-PCT1 80 2 P
49 PRE-N02 82 5 A
50 PRE-PCT2 87 2 P
51 PRE-N03 89 5 A
52 PRE-PCT3 94 2 P
53 PRE-N04 96 5 A
54 PRE-PCT4 101 2 P
55 PRE-N05 103 5 A
56 PRE-PCT5 108 2 P
57 *
58 FILE TRANSWK VIRTUAL RETAIN F 88
59 TRANS-REC 1 78 A
60 TRANS-KEY 1 17 A
61 TRANS-ID 1 2 A
62 TRANS-ID-CMD 1 1 A
```
Programs

53    TRANS-ID-OP     2   1   A
54    TRANS-PROJ-TASK 3   10  A
55    TRANS-PROJ      3   5   A
56    TRANS-TASK      8   5   A
57    TRANS-FIELD     13  5   A
58    TRANS-DATA      18  25  A
59    TRANS-DATE      18  6   N
60    TRANS-L7-D1    TRANS-DATA 7 N 1
61    TRANS-L7-D0    TRANS-DATA 7 N 0
62    TRANS-L3-D0    TRANS-DATA 3 N 0
63    TRANS-L7       TRANS-DATA 7 N
64    TRANS-L3       TRANS-DATA 3 N
65    TRANS-SCAN     TRANS-DATA 1 A -
         INDEX TRANS-NDX
66    TRANS-ID-CNv    79  2   A
67    *
68    *
69    *
70    ADD-PROJ-ID    W    2   A    VALUE 'AP'
71    ADD-CMD      ADD-PROJ-ID 1   A
72    PROJ-OP      ADD-PROJ-ID +1 1   A
73    ADD-TASK-ID  W    2   A    VALUE 'AT'
74    TASK-OP      ADD-TASK-ID 1   A
75    CHANGE-PROJ-ID W    2   A    VALUE 'CP'
76    CHANGE-CMD   CHANGE-PROJ-ID 1   A
77    CHANGE-TASK-ID W    2   A    VALUE 'CT'
78    DELETE-PROJ-ID W    2   A    VALUE 'DP'
79    DELETE-CMD   DELETE-PROJ-ID 1   A
80    DELETE-TASK-ID W    2   A    VALUE 'DT'
81    INC-TASK-ID  W    2   A    VALUE 'IT'
82    INC-CMD      INC-TASK-ID 1   A
83    MSG          W    55   A
84    CHANGE-MSG   W    15   A
85    TRANS-ERR-FLAG W    1   A
86    FIELD-DATA   W    25   A
87    MASTER-WAITING W    1   A
88    LAST-PROJ    W    PROJ-NO
89    LAST-TASK    W    PROJ-NO
90    DELETED-PROJ W    PROJ-NO
91    DELETED-TASK W    PROJ-NO
92    LAST-TRANS-KEY W    TRANS-KEY
93    LAST-TRANS-ID W    TRANS-ID
94    LAST-TRANS-CMD LAST-TRANS-ID 1   A
95    LAST-TRANS-OP LAST-TRANS-ID +1 1   A
96    TRANS-ERR-COUNT W    3   P    MASK('ZZZZ9')
97    SYSTEM-DATE   W    4   P
98    *
99    *
100   *
101   TRANS-ATTRIBUTES W    40   A
102   TRANS-TBL-CNv   TRANS-ATTRIBUTES 2   A
103   TRANS-Msg      TRANS-ATTRIBUTES +3 20  A
104   *
105   *
106   *
107   FIELD-ATTRIBUTES W    63   A
108   FIELD-NAME FIELD-ATTRIBUTES 25   A
109   FIELD-OFFSET FIELD-ATTRIBUTES +25 3   N 0
110   FIELD-LEN FIELD-ATTRIBUTES +29 2   N 0
111   FIELD-TYPE FIELD-ATTRIBUTES +32 1   A
112   FIELD-DEC FIELD-ATTRIBUTES +34 1   N
113   FIELD-SAVE FIELD-ATTRIBUTES +36 3   N   . * XREF TO LOCATION
114   *
115   *
116   FIELD-ID  W    5   A
117    *
118    * THE FOLLOWING TABLE CONTAINS ALL FIELDS ON THE MASTER FILE
119 * REFERENCED BY THE FIELD NAME ID.
120 * ALL IMPORTANT INFORMATION ABOUT THE FIELD IS STORED IN THE TABLE
121 * ENTRY, WHEN A NEW FIELD IS ADDED TO THE RECORD IT MUST BE
122 * ASSIGNED AN ID AND PUT IN THE TABLE
123 *
124 FILE FIELDTBL  TABLE INSTREAM
125   ARG          2        5        A
126   DESC         8        63       A
127   AENDT ACTUAL END DATE           61 04 D
128   AMAN ACTUAL HOURS SO FAR       53 04 P 1 069
129   ASTDT ACTUAL START DATE        57 04 D
130   EENDT EST END DATE             49 04 D 1
131   EMAN EST MAN HOURS             41 04 P 1
132   ESTDT EST START DATE           45 04 D
133   LACHR LAST ACTIVITY HRS        69 04 P 1
134   LTRDT LAST TRANSACTION DATE    65 04 D
135   MGRID MANAGER ID               36 05 X
136   NAME  NAME                      11 25 X
137   PCTCP PCT COMPLETE             73 02 P 0
138   PREN1 PREDECESSOR 1             75 05 X
139   PREN2 PREDECESSOR 2             82 05 X
140   PREN3 PREDECESSOR 3             89 05 X
141   PREN4 PREDECESSOR 4             96 05 X
142   PREN5 PREDECESSOR 5             103 05 X
143   PREP1 PRE PCT1                  80 02 P 0
144   PREP2 PRE PCT2                  87 02 P 0
145   PREP3 PRE PCT3                  94 02 P 0
146   PREP4 PRE PCT4                 101 02 P 0
147   PREP5 PRE PCT5                 108 02 P 0
148   PROJ PROJECT NUMBER            01 05 X
149   TASK TASK NUMBER               06 05 X
150 ENDTABLE
151 FILE TRANSTBL TABLE INSTREAM
152   ARG          2       2      A
153   DESC         5       40     A
154   AP 03 ADDED
155   AT 04 ADDED
156   CP 05 CHANGED
157   CT 06 CHANGED
158   DP 01 PROJECT DELETED
159   DT 02 TASK DELETED
160   IT 07 INCREMENTED
161 ENDTABLE
162 FILE TRANREG  PRINTER
163 FILE PRJLST   PRINTER
164 FILE UPDRPT   PRINTER
165 FILE TRANERR  PRINTER
166 FILE TRANSIN
167   COPY TRANSWK
168 FILE PRJOUT  FB(110 4400)
169   PROJ-DATA     1      10      A  -
170   INDEX PROJ-NDX
171 FILE PRJIN
172 *
173 JOB INPUT TRANSIN FINISH GOODBYE
174 *
175 * THIS FIRST JOB EDITS THE TRANSACTIONS AND CREATES AN OUTPUT
176 * FILE OF ALL VALID TRANSACTIONS
177 * INVALID TRANSACTIONS ARE LISTED ON THE ERROR REPORT
178 * AND SKIPPED
179 * VALID TRANSACTIONS HAVE THE TRANS ID CONVERTED TO A VALUE
180 * FROM 01 TO 07. THIS IS USED TO SORT THE TRANSACTIONS IN
SEARCH TRANSTBL WITH TRANSIN:TRANS-ID GIVING TRANS-ATTRIBUTES

IF NOT TRANSTBL
  MSG = 'INVALID TRANSACTION CODE'
  PERFORM TRANS-ERR
END-IF

IF TRANSIN:TRANS-ID-OP = PROJ-OP -
AND TRANSIN:TRANS-TASK NOT SPACES
  MSG = 'TASK NUMBER MUST BE SPACES FOR PROJECT TRANSACTION'
  PERFORM TRANS-ERR
END-IF

IF TRANSIN:TRANS-ID-CMD NE DELETE-CMD . * ONLY DELETES HAVE
  PERFORM FIELD-EDIT . * NO FIELD DATA
END-IF

IF TRANS-ERR-FLAG NE 'Y'
  TRANSIN:TRANS-ID-CNV = TRANS-TBL-CNV
  PUT TRANSWK FROM TRANSIN
ELSE
  TRANS-ERR-FLAG = ''
END-IF

FIELD-EDIT. PROC

SEARCH FIELDTBL WITH TRANSIN:TRANS-FIELD GIVING FIELD-ATTRIBUTES

IF NOT FIELDTBL
  MSG = 'INVALID FIELD ID'
  GOTO FIELD-ERR
END-IF

IF FIELD-TYPE = 'A' -
AND TRANS-DATA NOT ALPHABETIC
  MSG = 'DATA MUST BE ALPHABETIC'
  GOTO FIELD-ERR
END-IF

IF FIELD-TYPE = 'D' -
AND TRANS-DATE NOT NUMERIC
  MSG = 'DATE MUST BE NUMERIC'
  GOTO FIELD-ERR
END-IF

IF TRANSIN:TRANS-ID-CMD = INC-CMD -
AND FIELD-TYPE NE 'N' -
AND FIELD-TYPE NE 'P'
  MSG = 'FIELD CANNOT BE INCREMENTED'
  GOTO FIELD-ERR
ELSE
  GOTO FIELD-OK
END-IF

FIELD-OK
216   PERFORM TRANS-ERR
217   FIELD-OK
218   END-PROC
219 *
220 GOODBYE. PROC
222 *
223 * FINAL CHECK FOR MAX NUMBER OF ERRORS
224 *
225 DISPLAY TRANERR TRANS-ERR-COUNT ' ERRORS FOUND'
226 IF TRANS-ERR-COUNT GT 20
227   RETURN-CODE = TRANS-ERR-COUNT
228   STOP EXECUTE
229   END-IF
230 END-PROC
231 *
232 TRANS-ERR. PROC
234    TRANS-ERR-COUNT = TRANS-ERR-COUNT + 1
235    PRINT TRANS-ERR
236    TRANS-ERR-FLAG = 'Y'
237 END-PROC
238 *
239 REPORT TRANS-ERR PRINTER TRANERR LINESIZE 80
240 TITLE 'TRANSACTION ERROR REPORT 1'
241 LINE 1 TRANSIN:TRANS-REC
242 LINE 2 MSG
243 *
244 * THIS JOB SortS THE EDITED TRANSACTION FILE CREATED IN THE PREVIOUS
245 * JOB.
246 *
247 *
248 SORT TRANSWK TO TRANSWK -
249 USING (TRANS-PROJ TRANS-TASK TRANS-ID-CNV)
249 *
250 JOB INPUT (PRJIN KEY (PROJ-NO TASK-NO) -
251 TRANSWK KEY (TRANS-PROJ TRANS-TASK)) -
252 START HELLO
253 *
254 * MAINLINE CODE
255 * DEPENDING ON WHAT RECORDS ARE AVAILABLE
256 * PERFORM THE ROUTINE RESPONSIBLE
257 *
258 IF TRANSWK
259     PRINT TRANS-REG
260     PERFORM EDIT-TRANS
261 END-IF
262 *
263 IF PRJIN AND NOT TRANSWK
264   PERFORM MASTER-WITHOUT-TRANS
265   GOTO JOB
266 END-IF
267 *
268 IF PRJIN AND TRANSWK
269   PERFORM MASTER-WITH-TRANS
270   GOTO JOB
271 END-IF
272 *
273 IF TRANSWK AND NOT PRJIN
274   PERFORM TRANS-WITHOUT-MASTER
275   GOTO JOB
276 END-IF
277 *
278 EDIT-TRANS. PROC
279 *
280 * CHECK FOR DUPLICATE TRANACTIONS
281 * IF LAST-TRANS-KEY = TRANS-KEY
282   MSG = 'DUPLICATE TRANSACTION DROPPED'
283   PRINT ERROR-REPORT
284   GOTO JOB
285   ELSE
286     LAST-TRANS-KEY = TRANS-KEY
287     END-IF
288   END-PROC
289   *
290   FIELD-INC. PROC
291   *
292   * INCREMENT SELECTED FIELD WITH TRANSACTION DATA
293   * SAVE INCREMENT DATA IF FIELD-SAVE IS SPECIFIED
294   *
295   * DEPENDING ON THE LENGTH AND DECIMAL PLACES FOR THE FIELD
296   * ADD THE TRANSACTION DATA TO THE CORRECT OUTPUT FIELD
297   * IF AN INVALID TYPE IS IN THE TABLE THEN ABORT
298   *
299   * IF ANY NEW PACKED FIELD LENGTHS/DECIMAL PLACES ARE ADDED
300   * THIS SECTION OF THE PROGRAM MUST BE UPDATED
301   *
302   IF FIELD-TYPE NE 'P'
303     GOTO INC-ERROR
304   END-IF
305   IF FIELD-LEN = 4  AND FIELD-DEC = 1
306     PROJ-PACKED-L4-D1 = PROJ-PACKED-L4-D1 + TRANS-L7-D1
307     GOTO INC-TEST-FOR-SAVE
308   END-IF
309   IF FIELD-LEN = 4  AND FIELD-DEC = 0
310     PROJ-PACKED-L4-D0 = PROJ-PACKED-L4-D0 + TRANS-L7-D0
311     GOTO INC-TEST-FOR-SAVE
312   END-IF
313   IF FIELD-LEN = 2  AND FIELD-DEC = 0
314     PROJ-PACKED-L2-D0 = PROJ-PACKED-L2-D0 + TRANS-L3-D0
315     GOTO INC-TEST-FOR-SAVE
316   END-IF
317   INC-ERROR
318   DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
319     FIELD TYPE--CALL PROGRAMMING SUPPORT'
320   RETURN-CODE = 16
321   STOP EXECUTE
322   INC-TEST-FOR-SAVE
323   IF FIELD-SAVE NOT SPACES
324     PRJOUT:PROJ-NDX = FIELD-SAVE - 1
325     PERFORM FIELD-UPD
326   END-IF
327   END-PROC
328   *
329   FIELD-UPD. PROC
330   *
331   * UPDATE OF ADD DATA TO SELECTED FIELD
332   *
333   IF FIELD-TYPE = 'A' 'X'
334     MOVE TRANS-DATA FIELD-LEN TO PRJOUT:PROJ-DATA FIELD-LEN
335     GOTO UPD-END
336   END-IF
337   IF FIELD-TYPE = 'D'
338     PROJ-DATE = TRANS-DATE
339     GOTO UPD-END
340   END-IF
341   IF FIELD-TYPE = 'P'
342     PERFORM MOVE-PACKED-FIELD
343     GOTO UPD-END
344   END-IF
345   DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
346     FIELD TYPE--CALL PROGRAMMING SUPPORT'
347   RETURN-CODE = 16
STOP EXECUTE
UPD-END
END-PROC
HELLO. PROC
GETDATE SYSTEM-DATE . * GET SYSTEM DATE FOR DATE STAMP
* • GET THE CURRENT DATE AND PUT INTO USER FIELD LESS SLASHES
* • DEFINE GETDATE-DATE W 8 A
* • DEFINE GETDATE-FIRST6 GETDATE-DATE 6 N
* • DEFINE GETDATE-LAST5 GETDATE-DATE +3 5 A
* • DEFINE GETDATE-LAST6 GETDATE-DATE +2 6 A
* • DEFINE GETDATE-LAST3 GETDATE-DATE +5 3 A
* • DEFINE GETDATE-LAST2 GETDATE-DATE +6 2 A
* • GETDATE-DATE = SYSDATE . * MOVE ALL 8
* • GETDATE-LAST3 = GETDATE-LAST2 . * SHIFT LEFT OVER NEXT /
* • GETDATE-LAST6 = GETDATE-LAST5 . * SHIFT LEFT OVER FIRST /
• SYSTEM-DATE = GETDATE-FIRST6 . * MOVE TO USER FIELD
END-PROC
INIT-NEW-RECORD. PROC
* • INITIALIZE RECORD TO ZEROS AND SPACES
• • PROJ-REC(PRJOUT) = ' '
• • PROJ-TASK(PRJOUT) = TRANS-PROJ-TASK
• • MOVE ZERO TO PRJOUT:ACT-MAN-HRS-SO-FAR -
• • PRJOUT:ACT-END-DATE -
• • PRJOUT:ACT-START-DATE -
• • PRJOUT:EST-MAN-HRS -
• • PRJOUT:EST-END-DATE -
• • PRJOUT:EST-START-DATE -
• • PRJOUT:LAST-TRANSACTION-DATE -
• • PRJOUT:LAST-ACTIVITY-HRS -
• • PRJOUT:PCT-COMPLETE -
• • PRJOUT:PRE-PCT1 -
• • PRJOUT:PRE-PCT2 -
• • PRJOUT:PRE-PCT3 -
• • PRJOUT:PRE-PCT4 -
• • PRJOUT:PRE-PCT5 -
MASTER-WAITING EQ 'Y'
END-PROC
MASTER-DELETE. PROC
• • MASTER-WAITING = ' ' . * GET RID OF ACTIVE MASTER
• • LAST-TRANS-ID = TRANS-ID
• • SEARCH TRANSTBL WITH LAST-TRANS-ID -
• • GIVING TRANS-ATTRIBUTES
• • FIELD-DATA = ' '
• • CHANGE-MSG = TRANS-MSG
• • FIELD-ID = ' '
• • PRJOUT:PROJ-TASK = PROJ-TASK
• • PRINT CHANGES-REPORT
• • DELETED-PROJ = TRANS-PROJ . * SAVE WHAT WAS DELETED
• • DELETED-TASK = TRANS-TASK . * FOR FUTURE ADDS/CHANGES
• • PRJOUT:PROJ-TASK = ' ' . * NULL OUT HERE
END-PROC
MASTER-WITH-TRANS. PROC
• • IF FIRST-DUP TRANSWK OR NOT DUPLICATE TRANSWK
• • PRJOUT:PROJ-REC = PRJIN:PROJ-REC
• • MASTER-WAITING = 'Y'
END-IF
413   IF TRANS-ID-CMD = DELETE-CMD
414       PERFORM MASTER-DELETE
415       GOTO MAST-TRANS-DELETE
416   END-IF
417 *
418   IF TRANS-ID-CMD = ADD-CMD . * MAY BE ERROR
419       PERFORM VALIDATE-ADD-WITH-MASTER . * GO CHECK
420       GOTO MAST-TRANS-DONE
421   END-IF
422 *
423   IF MASTER-WAITING NE 'Y' . * MUST HAVE BEEN DELETED
424       MSG = 'CHANGES NOT VALID TO DELETED RECORD'
425       PRINT ERROR-REPORT
426       GOTO MAST-TRANS-DONE
427   END-IF
428 *
429   IF MASTER-WAITING NE 'Y' . * MUST HAVE BEEN DELETED
430       MSG = 'CHANGES NOT VALID TO DELETED RECORD'
431       PRINT ERROR-REPORT
432       GOTO MAST-TRANS-DONE
433   END-IF
434 MAST-TRANS-DONE
435   PERFORM TEST-FOR-MASTER-WRITE . * PUT IF LAST TRANS
436 MAST-TRANS-DELETE
437 END-PROC
438 *
439 MASTER-WITHOUT-TRANS. PROC
440 *
441   IF PROJ-NO = DELETED-PROJ -
442       AND DELETED-TASK = ' ' . * PRIOR PROJECT DELETE
443       CHANGE-MSG = 'TASK DELETED'
444       FIELD-DATA = ' '.
445       FIELD-ID = ' 
446       PRINT CHANGES-REPORT
447   ELSE
448   END-IF
449   END-PROC
450 *
451 MOVE-PACKED-FIELD. PROC
452 *
453 * DEPENDING ON THE LENGTH AND DECIMAL PLACES FOR THE FIELD
454 * MOVE THE TRANSACTION DATA TO THE CORRECT OUTPUT FIELD
455 * IF AN INVALID TYPE IS IN THE TABLE THEN ABORT
456 *
457 * IF ANY NEW PACKED FIELD LENGTHS/DECIMAL PLACES ARE ADDED
458 * THIS SECTION OF THE PROGRAM MUST BE UPDATED
459 *
460 IF FIELD-LEN = 4 AND FIELD-DEC = 1
461     PROJ-PACKED-L4-D1 = TRANS-L7-D1
462     GOTO PACKED-MOVED
463 END-IF
464 *
465 IF FIELD-LEN = 4 AND FIELD-DEC = 0
466     PROJ-PACKED-L4-D0 = TRANS-L7-D0
467     GOTO PACKED-MOVED
468 END-IF
469 *
470 IF FIELD-LEN = 2 AND FIELD-DEC = 0
471     PROJ-PACKED-L2-D0 = TRANS-L3-D0
472     GOTO PACKED-MOVED
473 END-IF
474 *
475 DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
476 FIELD TYPE--CALL PROGRAMMING SUPPORT'
477 RETURN-CODE = 16
478 STOP EXECUTE
479 PACKED-MOVED
480 END-PROC
481 *
482 PUT-MASTER. PROC
483 *
LAST-PROJ = PRJOUT:PROJ-NO
LAST-TASK = PRJOUT:TASK-NO
PUT PRJOUT
PRJOUT:PROJ-REC = ' '
MASTER-WAITING = ' '
END-PROC

TRANS-WITHOUT-MASTER. PROC

IF TRANS-ID = ADD-TASK-ID AND TRANS-PROJ NE LAST-PROJ
  MSG = 'PROJECT NOT FOUND FOR NEW TASK'
  PRINT ERROR-REPORT
  GOTO TRANS-DONE
END-IF

IF TRANS-ID-CMD NE ADD-CMD AND TRANS-PROJ-TASK NE PRJOUT:PROJ-TASK
  MSG = 'TRANSACTION DOES NOT HAVE A MATCHING MASTER'
  PRINT ERROR-REPORT
  GOTO TRANS-DONE
END-IF

IF MASTER-WAITING NE 'Y'
  PERFORM INIT-NEW-RECORD
END-IF

PERFORM UPDATE-RECORD
TRANS-DONE
PERFORM TEST-FOR-MASTER-WRITE
END-PROC

UPDATE-RECORD. PROC

SEARCH FIELDTBL WITH TRANS-FIELD GIVING FIELD-ATTRIBUTES
PRJOUT:PROJ-NDX = FIELD-OFFSET - 1
IF TRANS-ID-CMD = INC-CMD
  PERFORM FIELD-INC
ELSE
  PERFORM FIELD-UPD
END-IF
SEARCH TRANSTBL WITH TRANS-ID GIVING TRANS-ATTRIBUTES
CHANGE-MSG = TRANS-MSG
FIELD-DATA = TRANS-DATA
FIELD-ID = TRANS-FIELD
PRINT CHANGES-REPORT
END-PROC

TEST-FOR-MASTER-WRITE. PROC

TEST IF A MASTER IS WAITING TO WRITTEN AND THIS IS THE LAST TRANSACTION AGAINST IT
IF MASTER-WAITING NE 'Y'
  GOTO NO-MASTER
END-IF
IF LAST-DUP TRANSWK OR NOT DUPLICATE TRANSWK
  PRJOUT:LAST-TRANSACTION-DATE = SYSTEM-DATE
  PERFORM PUT-MASTER
END-IF
NO-MASTER
END-PROC

VALIDATE-ADD-WITH-MASTER. PROC
553 *
554 * THIS ROUTINE VERIFIES THAT THE ADD TRANSACTION FOR A MATCHING
555 * MASTER WAS PRECEDED BY A DELETE TRANSACTION
556 *
557 IF DELETED-PROJ NE PROJ-NO
558   GOTO ADD-TRANS-ERR
559 END-IF
560 *
561 IF DELETED-TASK NE TRANS-TASK AND DELETED-TASK NE ''
562   GOTO ADD-TRANS-ERR
563 END-IF
564 IF MASTER-WAITING NE 'Y'
565   PERFORM INIT-NEW-RECORD
566 END-IF
567 PERFORM UPDATE-RECORD
568 GOTO ADD-TRANS-OK
569 ADD-TRANS-ERR
570 MSG = 'ADD TRANS INVALID FOR EXISTING PROJ/TASK'
571 PRINT ERROR-REPORT
572 ADD-TRANS-OK
573 END-PROC
574 *
575 REPORT CHANGES-REPORT PRINTER UPDRPT LINESIZE 80 SPACE 1
576 CONTROL PRJOUT:PROJ-NO PRJOUT:TASK-NO
577 TITLE 'CHANGE REGISTER'
578 LINE 1 PRJOUT:PROJ-NO PRJOUT:TASK-NO CHANGE-MSG FIELD-ID FIELD-DATA TALLY
579 *
580 REPORT ERROR-REPORT PRINTER TRANERR SKIP 1 LINESIZE 80
581 TITLE 'TRANSACTION ERROR REPORT 2'
582 LINE 1 TRANS-REC
583 LINE 2 MSG
584 *
585 REPORT TRANS-REG PRINTER TRANREG LINESIZE 80
586 CONTROL
587 TITLE 'LISTING OF EDITED TRANSACTIONS'
588 LINE TRANS-REC
Output Reports

12/03/83  LISTING OF EDITED TRANSACTIONS  PAGE 1

TRANS-REC
APINVSM  EENDT030581
APINVSM  ESTDT030181
APINVSM  MGRIDM0002
APINVSM  NAME INVENTORY VSAM CONVERT
ATINVSM0001ASTDT030181
ATINVSM0001EENDT030281
ATINVSM0001EMAN 00000000
ATINVSM0001ESTDT030181
ATINVSM0001MGRIDM0001
ATINVSM0001NAME DETERMINE SPACE NEEDS
ATINVSM0001PCTCP010
ATINVSM0001AMAN 00000020
ATINVSM0001EENDT030381
ATINVSM0001EMAN 00000000
ATINVSM0001ESTDT030281
ATINVSM0001MGRIDM0002
ATINVSM0001NAME MAKE JCL CHANGES
ATINVSM0001EENDT030381
ATINVSM0001EMAN 00000040
ATINVSM0001ESTDT030381
ATINVSM0001MGRIDM0003
ATINVSM0001NAME DEFINE VSAM SPACE
ATINVSM0004EENDT030481
ATINVSM0004EMAN 00000000
ATINVSM0004ESTDT030481
ATINVSM0004MGRIDM0002
ATINVSM0004NAME TEST CHANGES
ATINVSM0005EENDT030581
ATINVSM0005EMAN 00000000
ATINVSM0005ESTDT030481
ATINVSM0005MGRIDM0002
ATINVSM0005NAME PUT IN PRODUCTION
APONLST  EENDT062881
APONLST  ESTDT060181
APONLST  MGRIDM2001
APONLST  NAME ONLINE CONVERT STUDY
ATONLSTT0001EENDT061081
ATONLSTT0001EMAN 00004000
ATONLSTT0001ESTDT060181
12/03/83  LISTING OF EDITED TRANSACTIONS  PAGE  2

TRANS-REC
ATONLSTT0001MGRIDM0001
ATONLSTT0001NAME DETERMINE NEEDS
ATONLSTT0002EENDT062181
ATONLSTT0002EMAN 0000800
ATONLSTT0002ESTDT061181
ATONLSTT0002MGRIDM0002
ATONLSTT0002NAME ONLINE MONITOR ANALYSIS
ATONLSTT0003EENDT062481
ATONLSTT0003EMAN 0000150
ATONLSTT0003ESTDT062281
ATONLSTT0003MGRIDM0003
ATONLSTT0003NAME DET. MANPOWER AVAIL.
ATONLSTT0004EENDT062881
ATONLSTT0004EMAN 0000150
ATONLSTT0004ESTDT062581
ATONLSTT0004MGRIDM0003
ATONLSTT0004NAME PREPARE REPORT
APTAXCH     EENDT021481
APTAXCH     ESTDT020181
APTAXCH     MGRIDM0001
APTAXCH     NAME PAYROLL TAX CHANGE
ATTAXCHT0001EENDT020481
ATTAXCHT0001EMAN 0000100
ATTAXCHT0001ESTDT020281
ATTAXCHT0001NAME PAYROLL TAX CHANGE
ATTAXCHT0001MGRIDM0001
ATTAXCHT0001NAME ANALYZE CHANGES
ATTAXCHT0002EENDT020681
ATTAXCHT0002EMAN 0000100
ATTAXCHT0002ESTDT020581
ATTAXCHT0002MGRIDM0002
ATTAXCHT0002NAME DETERMINE PROGRAM CHANGES
ATTAXCHT0002PREN1T0001
ATTAXCHT0003EENDT021081
ATTAXCHT0003EMAN 0000200
ATTAXCHT0003ESTDT020681
ATTAXCHT0003MGRIDM0002
ATTAXCHT0003NAME MAKE PROGRAM CHANGES
ATTAXCHT0003PREN1T0002
ATTAXCHT0004EENDT021381
ATTAXCHT0004EMAN 0000100
ATTAXCHT0004ESTDT021081
ATTAXCHT0004MGRIDM0002

12/03/83  LISTING OF EDITED TRANSACTIONS  PAGE  3

TRANS-REC
ATTAXCHT0004NAME TEST CHANGES
ATTAXCHT0004PREN1T0003
ATTAXCHT0005EENDT021481
ATTAXCHT0005EMAN 0000800
ATTAXCHT0005ESTDT021481
ATTAXCHT0005MGRIDM0002
ATTAXCHT0005NAME PUT IN PRODUCTION
ATTAXCHT0005PREN1T0004
<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>CHANGE-MSG</th>
<th>FIELD-ID</th>
<th>FIELD-DATA</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVSM</td>
<td></td>
<td>ADDED</td>
<td>EENDT</td>
<td>030581</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030181</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>INVENTORY VSAM CONVERT</td>
<td>4</td>
</tr>
<tr>
<td>INVSM T0001</td>
<td></td>
<td>ADDED</td>
<td>ASTDT</td>
<td>030181</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EENDT</td>
<td>030281</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000080</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030181</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DETERMINE SPACE NEEDS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>PCTCP</td>
<td>010</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AMAN</td>
<td>00000020</td>
<td>8</td>
</tr>
<tr>
<td>INVSM T0002</td>
<td></td>
<td>ADDED</td>
<td>EENDT</td>
<td>030381</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000080</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030281</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>MAKE JCL CHANGES</td>
<td>5</td>
</tr>
<tr>
<td>INVSM T0003</td>
<td></td>
<td>ADDED</td>
<td>EENDT</td>
<td>030381</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000040</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030381</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DEFINE VSAM SPACE</td>
<td>5</td>
</tr>
<tr>
<td>INVSM T0004</td>
<td></td>
<td>ADDED</td>
<td>EENDT</td>
<td>030481</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030481</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>TEST CHANGES</td>
<td>5</td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>INVSM</td>
<td>T0005</td>
<td>ADDED</td>
<td>EENDT</td>
<td>030581</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000040</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030481</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>PUT IN PRODUCTION</td>
<td></td>
</tr>
<tr>
<td>INVSM</td>
<td>T0005</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>INVSM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>ONLST</td>
<td>ADDED</td>
<td>EENDT</td>
<td>062881</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>060181</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M2001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ONLINE CONVERT STUDY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLST</td>
<td>T0001</td>
<td>ADDED</td>
<td>EENDT</td>
<td>061081</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000400</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>060181</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DETERMINE NEEDS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLST</td>
<td>T0001</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>ONLST</td>
<td>T0002</td>
<td>ADDED</td>
<td>EENDT</td>
<td>062181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000800</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>061181</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ONLINE MONITOR ANALYSIS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLST</td>
<td>T0002</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>ONLST</td>
<td>T0003</td>
<td>ADDED</td>
<td>EENDT</td>
<td>062481</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000150</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>062281</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DET. MANPOWER AVAIL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLST</td>
<td>T0003</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>ONLST T0004</td>
<td>ADDED</td>
<td>EENDT</td>
<td>062881</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000150</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>062581</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>PREPARE REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONLST T0004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>TAXCH T0001</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021481</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020181</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>PAYROLL TAX CHANGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAXCH T0001</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020481</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020281</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ANALYZE CHANGES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAXCH T0002</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020681</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020581</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DETERMINE PROGRAM CHANGES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>PREN1</td>
<td>T0001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>TAXCH T0003</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021081</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000200</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020681</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>MAKE PROGRAM CHANGES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PREN1</td>
<td>T0002</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>TAXCH T0003</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAXCH T0004</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021381</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>021001</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>TEST CHANGES</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PREN1</td>
<td>T0003</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>TAXCH T0004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAXCH T0005</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021481</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000080</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>021481</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>PUT IN PRODUCTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PREN1</td>
<td>T0004</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>TAXCH T0005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
</tr>
</tbody>
</table>
File Update Reports

Project Status - EXAMPLE 18.2

*** PROJECT MANAGEMENT SYSTEM EXAMPLE 18.2: FILE UPDATE ***

12/03/83                   LISTING OF EDITED TRANSACTIONS                   PAGE      1

TRAN-REC

CTINVSMT0001AENDT030281
CTINVSMT0001PCTCP100
ITINVSMT0001AMAN 0000070
CTINVSMT0002AENDT030481
CTINVSMT0002ASTDT030381
CTINVSMT0002PCTCP100
ITINVSMT0002AMAN 0000072
CTINVSMT0003ASTDT030581
CTINVSMT0003PCTCP099
ITINVSMT0003AMAN 0000042
CTONLSTT0001ASTDT060181
CTONLSTT0001PCTCP030
ITONLSTT0001AMAN 0000100
APRJE       EENDT031082
APRJE       ESTDT010482
APRJE       MGRIDM0004
APRJE       NAME ESTABLISH RJE SITE
ATRJE  T0001EENDT011582
ATRJE  T0001EMAN 0000070
ATRJE  T0001ESTDT010482
ATRJE  T0001MGRIDM0001
ATRJE  T0001NAME DETERMINE USER NEEDS
ATRJE  T0002EENDT011582
ATRJE  T0002EMAN 0000070
ATRJE  T0002ESTDT010482
ATRJE  T0002MGRIDM0002
ATRJE  T0002NAME DETERMINE EQUIP NEEDS
ATRJE  T0003EENDT012082
ATRJE  T0003EMAN 0000100
ATRJE  T0003ESTDT011882
ATRJE  T0003MGRIDM0002
ATRJE  T0003NAME GET TECHNICAL INFO
ATRJE  T0004EENDT012082
ATRJE  T0004EMAN 0000060
ATRJE  T0004ESTDT012082
ATRJE  T0004MGRIDM0002
ATRJE  T0004NAME GET COMM. LINE INFO
ATRJE  T0005EENDT012982
12/03/83  LISTING OF EDITED TRANSACTIONS  PAGE  2

TRANS-REC
ATRJE T0005EMAN 0000500
ATRJE T0005ESTDT012182
ATRJE T0005MGRIDM0003
ATRJE T0005NAME MATCH EQUIP TO NEEDS
ATRJE T0006EENDT020182
ATRJE T0006EMAN 0000060
ATRJE T0006ESTDT020182
ATRJE T0006MGRIDM0002
ATRJE T0006NAME ORDER COMMUNICATIONS LINE
ATRJE T0007EENDT020182
ATRJE T0007EMAN 0000020
ATRJE T0007ESTDT020182
ATRJE T0007MGRIDM0002
ATRJE T0007NAME SCHEDULE LINE INSTALL
ATRJE T0008EENDT020282
ATRJE T0008EMAN 0000060
ATRJE T0008ESTDT020282
ATRJE T0008MGRIDM0002
ATRJE T0008NAME ORDER COMPUTER EQUIPMENT
ATRJE T0009EENDT020282
ATRJE T0009EMAN 0000020
ATRJE T0009ESTDT020282
ATRJE T0009MGRIDM0002
ATRJE T0009NAME SCHEDULE COMPUTER INSTALL
ATRJE T0010EENDT021282
ATRJE T0010EMAN 0000060
ATRJE T0010ESTDT021282
ATRJE T0010MGRIDM0001
ATRJE T0010NAME PREPARE TRAINING PLAN
ATRJE T0011EENDT021282
ATRJE T0011EMAN 0000040
ATRJE T0011ESTDT021282
ATRJE T0011MGRIDM0001
ATRJE T0011NAME SCHEDULE USER TRAINING
ATRJE T0012EENDT021782
ATRJE T0012EMAN 0000200
ATRJE T0012ESTDT021582
ATRJE T0012MGRIDM0003
12/03/83        LISTING OF EDITED TRANSACTIONS       PAGE       3

TRANS-REC
ATRJE  T0012NAME PREPARE TEST PLAN
ATRJE  T0013EENDT021882
ATRJE  T0013EMAN 0000040
ATRJE  T0013ESTDT021882
ATRJE  T0013MGRIDM00001
ATRJE  T0013NAME ESTABLISH USER ID'S
ATRJE  T0014EENDT030182
ATRJE  T0014EMAN 0000080
ATRJE  T0014ESTDT030182
ATRJE  T0014MGRIDM00002
ATRJE  T0014NAME SUPERVISE EQUIP. DELIVERY
ATRJE  T0015EENDT030382
ATRJE  T0015EMAN 0000080
ATRJE  T0015ESTDT030382
ATRJE  T0015MGRIDM00003
ATRJE  T0015NAME PERFORM STAND-ALONE TESTS
ATRJE  T0016EENDT030682
ATRJE  T0016EMAN 0000200
ATRJE  T0016ESTDT030682
ATRJE  T0016MGRIDM00003
ATRJE  T0016NAME PERFORM RJE-HOST TESTS
ATRJE  T0017EENDT031082
ATRJE  T0017EMAN 0000240
ATRJE  T0017ESTDT031082
ATRJE  T0017MGRIDM00001
ATRJE  T0017NAME CONDUCT USER TRAINING
CPTAXCH  AENDT021381
CPTAXCH  ASTDT020281
CPTAXCH  PCTCP100
CPTAXCHT0001AENDT020481
CPTAXCHT0001ASTDT020481
CPTAXCHT0001PCTCP100
ITTAXCHT0001AMAN 0000100
CPTAXCHT0002ASTDT020481
CPTAXCHT0002ASTDT020681
CPTAXCHT0002PCTCP100
ITTAXCHT0002AMAN 0000120
CPTAXCHT0003AENDT020981
CPTAXCHT0003ASTDT020681
CPTAXCHT0003PCTCP100
ITTAXCHT0003AMAN 0000180
CPTAXCHT0004AENDT021181

0 ERRORS FOUND

12/03/83        LISTING OF EDITED TRANSACTIONS       PAGE       4

TRANS-REC
CPTAXCHT0004AENDT021081
CPTAXCHT0004PCTCP100
ITTAXCHT0004AMAN 0000100
CPTAXCHT0005ASTDT021281
CPTAXCHT0005PCTCP090
ITTAXCHT0005AMAN 0000100

0 ERRORS FOUND
12/03/83 TRANSACTION ERROR REPORT 2 PAGE 1

TRANS-REC

CTTAXC002A0STD020681
DUPLICATE TRANSACTION DROPPED

12/03/83 CHANGE REGISTER PAGE 1

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>CHANGE-MSG</th>
<th>FIELD-ID</th>
<th>FIELD-DATA</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVSM</td>
<td>T0001</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>030281</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>CHANGED</td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000070</td>
<td>3</td>
</tr>
<tr>
<td>INVSM</td>
<td>T0002</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>030481</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0002</td>
<td>CHANGED</td>
<td>ASTDT</td>
<td>030381</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0002</td>
<td>INCREMENTED</td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0002</td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000072</td>
<td>4</td>
</tr>
<tr>
<td>INVSM</td>
<td>T0003</td>
<td>CHANGED</td>
<td>ASTDT</td>
<td>030581</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0003</td>
<td>CHANGED</td>
<td>PCTCP</td>
<td>090</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0003</td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000042</td>
<td>3</td>
</tr>
<tr>
<td>INVSM</td>
<td>T0003</td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>ONLST</td>
<td>T0001</td>
<td>CHANGED</td>
<td>ASTDT</td>
<td>060181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>CHANGED</td>
<td>PCTCP</td>
<td>030</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000100</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>RJE</td>
<td>ADDED</td>
<td>EENDT</td>
<td>031082</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>030482</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0004</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ESTABLISH RJE SITE</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>RJE</td>
<td>T0001</td>
<td>ADDED</td>
<td>EENDT</td>
<td>011582</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000700</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>ADDED</td>
<td>ESTDT</td>
<td>010482</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0001</td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DETERMINE USER NEEDS</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>RJE</td>
<td>T0002</td>
<td>ADDED</td>
<td>EENDT</td>
<td>011582</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000700</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>010482</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>DETERMINE EQUIP NEEDS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0002</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0003</td>
<td>ADDED</td>
<td>EENDT</td>
<td>012082</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>011882</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>GET TECHNICAL INFO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0003</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0004</td>
<td>ADDED</td>
<td>EENDT</td>
<td>012082</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>012082</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>GET COMM. LINE INFO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0004</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0005</td>
<td>ADDED</td>
<td>EENDT</td>
<td>012982</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>012182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>MATCH EQUIP TO NEEDS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0005</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0006</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ORDER COMMUNICATIONS LINE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>T0006</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>RJE</td>
<td>T0007</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020182</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000020</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>SCHEDULE LINE INSTALL</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0008</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020282</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020282</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>ORDER COMPUTER EQUIPMENT</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0009</td>
<td>ADDED</td>
<td>EENDT</td>
<td>020282</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000020</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020282</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>SCHEDULE COMPUTER INSTALL</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0010</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021282</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>00000500</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>020382</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>PREPARE TRAINING PLAN</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0011</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021282</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>EMAN</td>
<td>0000040</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>ESTDT</td>
<td>021282</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDED</td>
<td>NAME</td>
<td>SCHEDULE USER TRAINING</td>
<td></td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>RJE</td>
<td>T0012</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021782</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000200</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>021582</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>PREPARE TEST PLAN</td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0013</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021882</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000040</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>021882</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>ESTABLISH USER ID'S</td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0014</td>
<td>ADDED</td>
<td>EENDT</td>
<td>030182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000080</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>030182</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0002</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>SUPERVISE EQUIP. DELIVERY</td>
<td>5</td>
</tr>
<tr>
<td>RJE</td>
<td>T0015</td>
<td>ADDED</td>
<td>EENDT</td>
<td>030382</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000080</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>030382</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>PERFORM STAND-ALONE TESTS</td>
<td>5</td>
</tr>
<tr>
<td>PROJECT NUMBER</td>
<td>TASK NUMBER</td>
<td>CHANGE-MSG</td>
<td>FIELD-ID</td>
<td>FIELD-DATA</td>
<td>TALLY</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>------------</td>
<td>-------</td>
</tr>
<tr>
<td>RJE</td>
<td>T0016</td>
<td>ADDED</td>
<td>EENDT</td>
<td>030682</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000200</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>030482</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0003</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>PERFORM RJE-HOST TESTS</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0016</td>
<td>ADDED</td>
<td>EENDT</td>
<td>031082</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EMAN</td>
<td>0000240</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ESTDT</td>
<td>030882</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MGRID</td>
<td>M0001</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAME</td>
<td>CONDUCT USER TRAINING</td>
<td></td>
</tr>
<tr>
<td>RJE</td>
<td>T0017</td>
<td>ADDED</td>
<td>EENDT</td>
<td>021381</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>020281</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0001</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>020481</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>020281</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000100</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0002</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>021181</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>021081</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000120</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0003</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>021281</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>021081</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>090</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000100</td>
<td></td>
</tr>
</tbody>
</table>

12/03/83                         CHANGE REGISTER                    PAGE      6

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>CHANGE-MSG</th>
<th>FIELD-ID</th>
<th>FIELD-DATA</th>
<th>TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXCH</td>
<td>T0003</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>020981</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>020681</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000180</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0004</td>
<td>CHANGED</td>
<td>AENDT</td>
<td>021181</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ASTDT</td>
<td>021081</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000100</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0005</td>
<td>CHANGED</td>
<td>ASTDT</td>
<td>021281</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCTCP</td>
<td>090</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCREMENTED</td>
<td>AMAN</td>
<td>0000100</td>
<td></td>
</tr>
<tr>
<td>TAXCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>21</td>
</tr>
</tbody>
</table>
As shown in the report output (shown later), three report programs illustrated in Examples 17.3, 17.4, and 17.5, are provided to give you an idea of what is possible using the Project Management System Master File. The function and operation of each program is documented within the example.

**Project Status - EXAMPLE 18.3**

```plaintext
1 *
2 * PROJECT MANAGEMENT SYSTEM - EXAMPLE 18.3
3 *
4 %PROJLIB
47 *
48 * THIS JOB PRODUCES TWO REPORTS
49 * ONE IS FOR THE PROJECT/TASK MANAGER
50 * THE OTHER IS AN OVER STATUS REPORT FOR EACH PROJECT
51 *
52 JOB
53 *
54 IF TASK-NO NOT SPACES
55 PRINT MANAGER-REPORT
56 PRINT STATUS-REPORT
57 END-IF
58 *
59 *
60 REPORT STATUS-REPORT LINESIZE 80 SPACE 1
61 CONTROL PROJ-NO
62 TITLE 'PROJECT STATUS REPORT'
63 LINE PROJ-NO TASK-NO MANAGER-ID EST-START-DATE EST-END-DATE - EST-MAN-HRS ACT-MAN-HRS-50-FAR
64 *
65 REPORT MANAGER-REPORT LINESIZE 80 SPACE 1
66 SEQUENCE MANAGER-ID PROJ-NO TASK-NO
67 CONTROL MANAGER-ID NEWPAGE PROJ-NO
68 TITLE 'PROJECT LIST BY MANAGER'
69 TITLE 3 'MANAGER ID:' -1 MANAGER-ID
70 LINE PROJ-NO TASK-NO EST-START-DATE EST-END-DATE EST-MAN-HRS
71 *
```
<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>MANAGER ID</th>
<th>EST START</th>
<th>EST END</th>
<th>EST HOURS</th>
<th>ACTUAL HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVSM</td>
<td>T0001</td>
<td>M0001</td>
<td>03/01/81</td>
<td>03/02/81</td>
<td>8.0</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>T0002</td>
<td>M0002</td>
<td>03/02/81</td>
<td>03/03/81</td>
<td>8.0</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>T0003</td>
<td>M0003</td>
<td>03/03/81</td>
<td>03/03/81</td>
<td>4.0</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td>T0004</td>
<td>M0002</td>
<td>03/04/81</td>
<td>03/04/81</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td>T0005</td>
<td>M0002</td>
<td>03/04/81</td>
<td>03/05/81</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30.0</td>
<td>20.4</td>
</tr>
</tbody>
</table>

| ONLST         | T0001       | M0001      | 06/01/81  | 06/10/81| 40.0      | 10.0         |
|               | T0002       | M0002      | 06/11/81  | 06/21/81| 80.0      | 0.0          |
|               | T0003       | M0003      | 06/22/81  | 06/24/81| 15.0      | 0.0          |
|               | T0004       | M0003      | 06/25/81  | 06/28/81| 15.0      | 0.0          |
|               |             |            |           |         | 150.0     | 10.0         |

| RJE           | T0001       | M0001      | 01/04/82  | 01/15/82| 70.0      | 0.0          |
|               | T0002       | M0002      | 01/04/82  | 01/15/82| 70.0      | 0.0          |
|               | T0003       | M0002      | 01/18/82  | 01/29/82| 10.0      | 0.0          |
|               | T0004       | M0002      | 01/20/82  | 01/29/82| 6.0       | 0.0          |
|               | T0005       | M0003      | 01/21/82  | 01/29/82| 50.0      | 0.0          |
|               | T0006       | M0002      | 02/01/82  | 02/01/82| 6.0       | 0.0          |
|               | T0007       | M0002      | 02/01/82  | 02/01/82| 2.0       | 0.0          |
|               | T0008       | M0002      | 02/02/82  | 02/02/82| 6.0       | 0.0          |
|               | T0009       | M0002      | 02/02/82  | 02/02/82| 2.0       | 0.0          |
|               | T0010       | M0001      | 02/03/82  | 02/12/82| 50.0      | 0.0          |
|               | T0011       | M0001      | 02/12/82  | 02/12/82| 4.0       | 0.0          |
|               | T0012       | M0003      | 02/15/82  | 02/17/82| 20.0      | 0.0          |
|               | T0013       | M0001      | 02/18/82  | 02/18/82| 4.0       | 0.0          |
|               | T0014       | M0002      | 03/01/82  | 03/01/82| 8.0       | 0.0          |
|               | T0015       | M0003      | 03/03/82  | 03/03/82| 8.0       | 0.0          |
|               | T0016       | M0003      | 03/04/82  | 03/06/82| 20.0      | 0.0          |
|               | T0017       | M0001      | 03/08/82  | 03/10/82| 24.0      | 0.0          |
|               |             |            |           |         | 360.0     | 0.0          |

| TAXCH         | T0001       | M0001      | 02/02/81  | 02/04/81| 10.0      | 10.0         |
|               | T0002       | M0002      | 02/05/81  | 02/06/81| 10.0      | 12.0         |
|               | T0003       | M0002      | 02/06/81  | 02/10/81| 20.0      | 18.0         |
|               | T0004       | M0002      | 02/10/81  | 02/13/81| 10.0      | 10.0         |
|               | T0005       | M0002      | 02/14/81  | 02/14/81| 8.0       | 10.0         |
|               |             |            |           |         | 58.0      | 60.0         |
### PROJECT LIST BY MANAGER

**12/03/83**  
**MANAGER ID:** M0001

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVSM</td>
<td>T0001</td>
<td>03/01/81</td>
<td>03/02/81</td>
<td>8.0</td>
</tr>
<tr>
<td>INVSM</td>
<td>T0001</td>
<td>03/01/81</td>
<td>03/02/81</td>
<td>8.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLST</td>
<td>T0001</td>
<td>06/01/81</td>
<td>06/10/81</td>
<td>40.0</td>
</tr>
<tr>
<td>ONLST</td>
<td>T0001</td>
<td>06/01/81</td>
<td>06/10/81</td>
<td>40.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJE</td>
<td>T0001</td>
<td>01/04/82</td>
<td>01/15/82</td>
<td>70.0</td>
</tr>
<tr>
<td>T0010</td>
<td>02/03/82</td>
<td>02/12/82</td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td>T0011</td>
<td>02/12/82</td>
<td>02/12/82</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>T0013</td>
<td>02/18/82</td>
<td>02/18/82</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>T0017</td>
<td>03/08/82</td>
<td>03/10/82</td>
<td>24.0</td>
<td></td>
</tr>
</tbody>
</table>

RJE: 152.0

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXCH</td>
<td>T0001</td>
<td>02/02/81</td>
<td>02/04/81</td>
<td>10.0</td>
</tr>
<tr>
<td>TAXCH</td>
<td>T0001</td>
<td>02/02/81</td>
<td>02/04/81</td>
<td>10.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVSM</td>
<td>T0002</td>
<td>03/02/81</td>
<td>03/03/81</td>
<td>8.0</td>
</tr>
<tr>
<td>T0004</td>
<td>03/04/81</td>
<td>03/04/81</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>T0005</td>
<td>03/04/81</td>
<td>03/05/81</td>
<td>4.0</td>
<td></td>
</tr>
</tbody>
</table>

INVSM: 18.0

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONLST</td>
<td>T0002</td>
<td>06/11/81</td>
<td>06/21/81</td>
<td>80.0</td>
</tr>
<tr>
<td>ONLST</td>
<td>T0002</td>
<td>06/11/81</td>
<td>06/21/81</td>
<td>80.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJE</td>
<td>T0002</td>
<td>01/04/82</td>
<td>01/15/82</td>
<td>70.0</td>
</tr>
<tr>
<td>T0003</td>
<td>01/18/82</td>
<td>01/20/82</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>T0004</td>
<td>01/20/82</td>
<td>01/20/82</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>T0006</td>
<td>02/01/82</td>
<td>02/01/82</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>T0007</td>
<td>02/01/82</td>
<td>02/01/82</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>T0008</td>
<td>02/02/82</td>
<td>02/02/82</td>
<td>6.0</td>
<td></td>
</tr>
<tr>
<td>T0009</td>
<td>02/02/82</td>
<td>02/02/82</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>T0014</td>
<td>03/01/82</td>
<td>03/01/82</td>
<td>8.0</td>
<td></td>
</tr>
</tbody>
</table>

RJE: 110.0

<table>
<thead>
<tr>
<th>PROJECT NUMBER</th>
<th>TASK NUMBER</th>
<th>EST START DATE</th>
<th>END DATE</th>
<th>EST HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAXCH</td>
<td>T0002</td>
<td>02/05/81</td>
<td>02/06/81</td>
<td>10.0</td>
</tr>
<tr>
<td>T0003</td>
<td>02/06/81</td>
<td>02/10/81</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>T0004</td>
<td>02/10/81</td>
<td>02/13/81</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>T0005</td>
<td>02/14/81</td>
<td>02/14/81</td>
<td>8.0</td>
<td></td>
</tr>
</tbody>
</table>

TAXCH: 256.0
Project Summary

Project Summary - EXAMPLE 18.4

1 *
2 * PROJECT MANAGEMENT SYSTEM - EXAMPLE 18.4
3 *
4 %PROJLIB
47 PCT PCT-COMPLETE  2 P 0    . * REDEFINE PCT COMPLETE TO TOTAL
49 *
50 * THIS EXAMPLE IS A SIMPLE SUMMARY REPORT OF ALL PROJECTS
51 *
52 JOB
53 *
54 IF TASK-NO NOT SPACES . * ONLY GET TASK RECORDS
55 PRINT PROJECT-SUMMARY
56 END-IF
58 *
59 REPORT PROJECT-SUMMARY LINESIZE 80 SPACE 1 SUMMARY
60 CONTROL PROJ-NO
61 TITLE 'SUMMARY LISTING OF PROJECTS AND MAN HOURS'
62 HEADING TALLY ('NUMBER' 'OF' 'TASKS')
63 LINE 1 PROJ-NO ACT-MAN-HRS-SO-FAR EST-MAN-HRS PCT TALLY
64 *
65 BEFORE-BREAK. PROC
67 PCT = PCT / TALLY . * COMPUTE AVE PCT COMPLETE
69 END-PROC
### Project Completion - EXAMPLE 18.5

1 * 
2 * PROJECT MANAGEMENT SYSTEM - EXAMPLE 18.5 
3 * 
4 %PROJLIB 
47 * 
48 * THIS JOB PRODUCES TWO REPORTS 
49 * THE FIRST GIVES THE PERCENT VARIANCE OF ESTIMATED VERSUS ACTUAL 
50 * COMPLETION TIMES. 
51 * 
52 * THE SECOND GIVES THE ESTIMATED TIME TO COMPLETE FOR THE 
53 * TASKS THAT HAVE STARTED BUT NOT FINISHED YET 
54 * 
55 * 
56 * 
57 * 
58 JOB 
59 * 
60 IF TASK-NO SPACES OR ACT-MAN-HRS-SO-FAR = 0 
61 GO TO JOB 
62 END-IF 
63 IF PCT-COMPLETE = 100 
64 VARIANCE = ((ACT-MAN-HRS-SO-FAR - EST-MAN-HRS) / EST-MAN-HRS) * 100 
65 PRINT COMPLETE-REPORT 
66 ELSE 
67 HRS-TO-GO = (ACT-MAN-HRS-SO-FAR / (PCT-COMPLETE / 100)) - 
68 ACT-MAN-HRS-SO-FAR 
69 PRINT PREDICTIONS 
70 END-IF 
71 * 
72 REPORT COMPLETE-REPORT LINESIZE 80 
73 SEQUENCE VARIANCE D 
74 CONTROL PROJ-NO 
75 TITLE 1 'LISTING OF ALL COMPLETED TASKS' 
76 TITLE 2 'WITH THE VARIANCE OF ACTUAL TO ESTIMATED TIME' 
77 LINE 1 PROJ-NO TASK-NO MANAGER-ID VARIANCE 
78 * 
79 BEFORE-BREAK. PROC 
80 VARIANCE = VARIANCE / TALLY 
81 END-PROC 
82 * 
83 REPORT PREDICTIONS LINESIZE 80 
84 SEQUENCE HRS-TO-GO D 
85 TITLE 'LISTING OF UNCOMPLETED TASKS AND HOURS TO COMPLETE' 
86 LINE PROJ-NO TASK-NO MANAGER-ID HRS-TO-GO 
88 *
### Listing of All Completed Tasks

**Project Number** | **Task Number** | **Manager ID** | **Variance**
--- | --- | --- | ---
TAXCH | T0002 | M0002 | 20.0
TAXCH | T0002 | M0002 | 20.0
INVSMT | T0001 | M0001 | 12.5
INVSMT | T0001 | M0001 | 12.5
TAXCH | T0004 | M0002 | 0.0
TAXCH | T0001 | M0001 | 0.0
TAXCH | | | 0.0
INVSMT | T0002 | M0002 | 10.0-
INVSMT | T0002 | M0002 | 10.0-
TAXCH | T0003 | M0002 | 10.0-
TAXCH | T0003 | M0002 | 10.0-
| | | 2.0

### Listing of Uncompleted Tasks and Hours to Complete

**Project Number** | **Task Number** | **Manager ID** | **Hrs-To-Go**
--- | --- | --- | ---
ONLST | T0001 | M0001 | 23.3
TAXCH | T0005 | M0002 | 1.1
INVSMT | T0003 | M0003 | 0.4
## Functions

This appendix contains an alphabetical list and brief description of the subset of CA-Easytrieve/Plus statements covered in this Application Guide.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Establishes the value of a field.</td>
</tr>
<tr>
<td>CONTROL</td>
<td>Identifies control fields used for a control report.</td>
</tr>
<tr>
<td>DEFINE</td>
<td>Specifies data fields within a file or within working storage.</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Prints data from specified fields to the system printer file or a named printer file.</td>
</tr>
<tr>
<td>DO</td>
<td>Controls repetitive program logic based on the truth value of associated conditional expressions.</td>
</tr>
<tr>
<td>ELSE</td>
<td>Identifies the statement(s) to be executed when the condition in an IF statement tests false.</td>
</tr>
<tr>
<td>END-DO</td>
<td>Identifies the end of the statements associated with a DO statement.</td>
</tr>
<tr>
<td>END-IF</td>
<td>Identifies the end of the statements associated with an IF statement.</td>
</tr>
<tr>
<td>END-PROC</td>
<td>Identifies the end of a procedure.</td>
</tr>
<tr>
<td>FILE</td>
<td>Describes a file used by your program.</td>
</tr>
<tr>
<td>GET</td>
<td>Reads the next sequential record of a file into storage.</td>
</tr>
<tr>
<td>GOTO</td>
<td>Passes control to the specified location in the program.</td>
</tr>
<tr>
<td>HEADING</td>
<td>Specifies an alternate column heading for a field on a report.</td>
</tr>
<tr>
<td>IF</td>
<td>Controls the execution of associated statements depending on the truth value of conditional expressions.</td>
</tr>
<tr>
<td>Statement</td>
<td>Function</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>JOB</td>
<td>Defines an activity that retrieves input files, examines and manipulates data, initiates printed reports, or produces output files.</td>
</tr>
<tr>
<td>LINE</td>
<td>Defines the contents of a report line.</td>
</tr>
<tr>
<td>PARM</td>
<td>Establishes program-level environment parameters.</td>
</tr>
<tr>
<td>PERFORM</td>
<td>Transfers control to a procedure and returns control to the next executable statement after the procedure is executed.</td>
</tr>
<tr>
<td>POINT</td>
<td>Locates the position of a specified record in a file.</td>
</tr>
<tr>
<td>PRINT</td>
<td>Outputs data to a file for inclusion in a report.</td>
</tr>
<tr>
<td>PROC</td>
<td>Identifies the beginning of a procedure.</td>
</tr>
<tr>
<td>PUT</td>
<td>Outputs a record to a file.</td>
</tr>
<tr>
<td>READ</td>
<td>Provides random access to keyed and relative-record files.</td>
</tr>
<tr>
<td>RECORD</td>
<td>Identifies IMS/DLI database segments available for processing.</td>
</tr>
<tr>
<td>REPORT</td>
<td>Establishes the type and characteristics of a report.</td>
</tr>
<tr>
<td>RETRIEVE</td>
<td>Specifies the segments to be automatically input from an IMS/DLI database.</td>
</tr>
<tr>
<td>SEARCH</td>
<td>Provides access to table information.</td>
</tr>
<tr>
<td>SELECT</td>
<td>Used in a sort procedure to select individual records for the sort output.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Specifies the order of a report.</td>
</tr>
<tr>
<td>SORT</td>
<td>Inputs a sequential file and outputs the result of the sort operation onto an output file.</td>
</tr>
<tr>
<td>STOP</td>
<td>Terminates an activity.</td>
</tr>
<tr>
<td>TITLE</td>
<td>Defines report title items and their position on the title line.</td>
</tr>
<tr>
<td>WRITE</td>
<td>Updates or deletes existing records and adds new records in the processing of keyed and relative-record files.</td>
</tr>
</tbody>
</table>
Cross-Reference of Statement

This appendix presents a cross-reference listing of CA-Easytrieve/Plus statements to the specific examples in Chapters 15 through 18. In addition, a second listing is provided which cross-references CA-Easytrieve/Plus functions or facilities to the same examples.

Within these two larger groupings, the statements and facilities are grouped by program section; that is, Library, JOB/SORT Activity, REPORT Activity, and so forth. Within the subgroups, the statements and facilities are presented alphabetically.

The examples are numbered by a two-part designator which includes the chapter number. That is, Example 15.10 is the 10th example presented in Chapter 15, Example 16.14 is the 14th example in Chapter 16, and so on. Please refer to the Table of Contents or Exhibit Indexes to determine the exact page number on which to locate a specific example.
<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>18.01</td>
<td>FILE</td>
<td>15.01</td>
</tr>
<tr>
<td>DEFINE (EXPLICIT)</td>
<td>16.12</td>
<td></td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td>DEFINE (IMPLICIT)</td>
<td>15.05</td>
<td></td>
<td>15.06</td>
</tr>
<tr>
<td></td>
<td>15.02</td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.03</td>
<td></td>
<td>15.08</td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>15.08</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.13</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>15.14</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td></td>
<td>15.18</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>16.03</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.04</td>
<td></td>
<td>16.06</td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>16.07</td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>16.09</td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td>17.00</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td>17.01</td>
</tr>
<tr>
<td></td>
<td>17.00</td>
<td></td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>17.01</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>17.04</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td>18.00</td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>18.00</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>18.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## CA-Easytrieve/Plus Example Cross-reference

### Statements for JOB/SORT

<table>
<thead>
<tr>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE</th>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSIGNMENT</td>
<td>15.02</td>
<td>GOTO JOB</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>15.03</td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>15.14</td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td>IF</td>
<td>15.01</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td>15.04</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td></td>
<td>15.06</td>
</tr>
<tr>
<td></td>
<td>17.01</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>17.05</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEFINE (EXPLICIT)</td>
<td>18.01</td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>15.08</td>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>16.07</td>
<td></td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>16.09</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td>17.04</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td>STATEMENT/ FACILITY</td>
<td>EXAMPLE NUMBER</td>
<td>STATEMENT/ FACILITY</td>
<td>EXAMPLE NUMBER</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>---------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>DO WHILE</td>
<td>16.09</td>
<td>JOB</td>
<td>17.01</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td></td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td>17.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17.05</td>
</tr>
<tr>
<td>GOTO</td>
<td>17.02</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td>JOB</td>
<td></td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>15.01</td>
<td>MOVE</td>
<td>15.08</td>
</tr>
<tr>
<td></td>
<td>15.02</td>
<td>MOVE LIKE</td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.03</td>
<td>PERFORM</td>
<td>15.08</td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>15.06</td>
<td></td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td>15.08</td>
<td>POINT</td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>16.01</td>
</tr>
<tr>
<td></td>
<td>15.13</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>15.14</td>
<td></td>
<td>16.03</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td>16.07</td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td>16.12</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
<td></td>
<td>16.14</td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference

**Statements for JOB/SORT**

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td></td>
<td>PROC</td>
<td>15.08</td>
</tr>
<tr>
<td>15.01</td>
<td></td>
<td>15.15</td>
<td></td>
</tr>
<tr>
<td>15.02</td>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td>15.03</td>
<td></td>
<td>16.04</td>
<td></td>
</tr>
<tr>
<td>15.04</td>
<td></td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td>15.05</td>
<td></td>
<td>16.07</td>
<td></td>
</tr>
<tr>
<td>15.06</td>
<td></td>
<td>16.14</td>
<td></td>
</tr>
<tr>
<td>15.07</td>
<td></td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td>15.09</td>
<td></td>
<td>17.03</td>
<td></td>
</tr>
<tr>
<td>15.10</td>
<td></td>
<td>17.04</td>
<td></td>
</tr>
<tr>
<td>15.11</td>
<td></td>
<td>17.05</td>
<td></td>
</tr>
<tr>
<td>15.12</td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>15.14</td>
<td></td>
<td>15.08</td>
<td></td>
</tr>
<tr>
<td>15.15</td>
<td>PUT</td>
<td>15.08</td>
<td></td>
</tr>
<tr>
<td>15.16</td>
<td></td>
<td>15.14</td>
<td></td>
</tr>
<tr>
<td>15.17</td>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td>16.01</td>
<td></td>
<td>16.04</td>
<td></td>
</tr>
<tr>
<td>16.02</td>
<td></td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td>16.03</td>
<td></td>
<td>16.07</td>
<td></td>
</tr>
<tr>
<td>16.04</td>
<td></td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td>16.05</td>
<td></td>
<td>17.01</td>
<td></td>
</tr>
<tr>
<td>16.06</td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>16.08</td>
<td></td>
<td>16.08</td>
<td></td>
</tr>
<tr>
<td>16.09</td>
<td>READ</td>
<td>16.08</td>
<td></td>
</tr>
<tr>
<td>16.10</td>
<td></td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td>16.11</td>
<td></td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td>17.03</td>
<td></td>
<td>16.04</td>
<td>SEARCH</td>
</tr>
<tr>
<td>17.04</td>
<td></td>
<td>15.09</td>
<td></td>
</tr>
<tr>
<td>17.05</td>
<td></td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td>18.01</td>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td>18.03</td>
<td></td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td>18.04</td>
<td></td>
<td>17.03</td>
<td></td>
</tr>
<tr>
<td>18.05</td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>18.08</td>
<td>STOP</td>
<td>16.08</td>
<td></td>
</tr>
<tr>
<td>16.10</td>
<td></td>
<td>16.12</td>
<td></td>
</tr>
<tr>
<td>16.11</td>
<td></td>
<td>16.14</td>
<td></td>
</tr>
<tr>
<td>17.01</td>
<td></td>
<td>16.04</td>
<td>SEARCH</td>
</tr>
<tr>
<td>17.04</td>
<td></td>
<td>15.09</td>
<td></td>
</tr>
<tr>
<td>17.05</td>
<td></td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td>18.01</td>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td>18.03</td>
<td></td>
<td>17.02</td>
<td></td>
</tr>
<tr>
<td>18.04</td>
<td></td>
<td>17.03</td>
<td></td>
</tr>
<tr>
<td>18.05</td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>18.08</td>
<td>STOP EXECUTE</td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>WRITE</td>
<td></td>
<td>16.08</td>
<td></td>
</tr>
<tr>
<td>16.10</td>
<td></td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td>16.12</td>
<td></td>
<td>17.02</td>
<td></td>
</tr>
</tbody>
</table>

---

**CA-Easytrieve/Plus Example Cross-reference**

**Statements for JOB/SORT**

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>16.04</td>
</tr>
<tr>
<td>SORT</td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td>STOP</td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
</tr>
<tr>
<td></td>
<td>17.01</td>
</tr>
<tr>
<td>STOP EXECUTE</td>
<td>18.01</td>
</tr>
<tr>
<td>WRITE</td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
</tr>
</tbody>
</table>
## CA-Easytrieve/Plus Example Cross-reference

### Statements for REPORT

<table>
<thead>
<tr>
<th>STATEMENT/</th>
<th>EXAMPLE</th>
<th>STATEMENT/</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACILITY</td>
<td>NUMBER</td>
<td>FACILITY</td>
<td>NUMBER</td>
</tr>
<tr>
<td>CONTROL</td>
<td>15.02</td>
<td>LINE</td>
<td>15.01</td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td></td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.04</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.06</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td>15.08</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>16.04</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>16.09</td>
<td></td>
<td>16.01</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td>16.03</td>
</tr>
<tr>
<td></td>
<td>17.05</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>18.03</td>
<td></td>
<td>16.06</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td>DISPLAY</td>
<td>15.11</td>
<td>17.03</td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td>17.05</td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td>18.01</td>
<td>18.03</td>
</tr>
<tr>
<td>HEADING</td>
<td>15.09</td>
<td>18.04</td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td>MOVE</td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td>STATEMENT/FACILITY</td>
<td>EXAMPLE/NUMBER</td>
<td>STATEMENT/FACILITY</td>
<td>EXAMPLE/NUMBER</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>PROC</td>
<td>15.02</td>
<td>REPORT</td>
<td>17.05</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td>SEQUENCE</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td></td>
<td>15.04</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td>15.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td>REPORT</td>
<td>15.01</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.02</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>15.03</td>
<td></td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>15.06</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>16.03</td>
</tr>
<tr>
<td></td>
<td>15.08</td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>16.03</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>16.06</td>
</tr>
<tr>
<td></td>
<td>15.14</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>17.05</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td>SUM</td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## CA-Easytrieve/Plus Example Cross-reference
### Statements for REPORT

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE</td>
<td></td>
<td>TITLE</td>
<td></td>
</tr>
<tr>
<td>15.01</td>
<td></td>
<td>15.02</td>
<td></td>
</tr>
<tr>
<td>15.05</td>
<td></td>
<td>15.06</td>
<td></td>
</tr>
<tr>
<td>15.07</td>
<td></td>
<td>15.09</td>
<td></td>
</tr>
<tr>
<td>15.10</td>
<td></td>
<td>15.11</td>
<td></td>
</tr>
<tr>
<td>15.12</td>
<td></td>
<td>15.14</td>
<td></td>
</tr>
<tr>
<td>15.15</td>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td>15.17</td>
<td></td>
<td>16.01</td>
<td></td>
</tr>
<tr>
<td>16.02</td>
<td></td>
<td>16.03</td>
<td></td>
</tr>
<tr>
<td>16.04</td>
<td></td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td>16.06</td>
<td></td>
<td>16.08</td>
<td></td>
</tr>
<tr>
<td>16.09</td>
<td></td>
<td>16.10</td>
<td></td>
</tr>
<tr>
<td>16.11</td>
<td></td>
<td>17.03</td>
<td></td>
</tr>
<tr>
<td>17.05</td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>18.03</td>
<td></td>
<td>18.04</td>
<td></td>
</tr>
<tr>
<td>18.05</td>
<td></td>
<td>18.05</td>
<td></td>
</tr>
</tbody>
</table>

## Facilities for MACROS

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYWORD SUBSTITUTION</td>
<td>17.00</td>
<td>POSITIONAL SUBSTITUTION</td>
<td>16.12</td>
</tr>
<tr>
<td>MACRO DEFINITION</td>
<td>16.12</td>
<td>MACRO DEFINITION</td>
<td>18.00</td>
</tr>
<tr>
<td>16.14</td>
<td></td>
<td>17.00</td>
<td></td>
</tr>
<tr>
<td>18.00</td>
<td></td>
<td>18.00</td>
<td></td>
</tr>
</tbody>
</table>
## CA-Easytrieve/Plus Example Cross-reference

Facilities for LIBRARY

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARRAY</td>
<td>17.00</td>
<td>MACRO CALL</td>
<td>15.01</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.00</td>
<td></td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.04</td>
</tr>
<tr>
<td>FIELD</td>
<td></td>
<td>15.05</td>
<td></td>
</tr>
<tr>
<td>REDEFINITION</td>
<td></td>
<td>15.06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
<td></td>
<td>15.11</td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td>18.00</td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.01</td>
</tr>
<tr>
<td>HEADING</td>
<td></td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.02</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>15.03</td>
<td></td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td></td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>17.01</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td></td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
<td>17.04</td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
<td>17.05</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>17.00</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td>18.00</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td>INDEXING</td>
<td>16.14</td>
<td>MASK</td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>17.00</td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>18.00</td>
<td></td>
<td>18.00</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>18.01</td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference

#### Facilities for LIBRARY

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL VALUE</td>
<td></td>
<td>15.07</td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.00</td>
<td>16.08</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.02</td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18.01</td>
<td>16.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.09</td>
<td></td>
</tr>
<tr>
<td>PRINTER FILE</td>
<td></td>
<td>17.04</td>
<td>16.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18.01</td>
<td>16.09</td>
</tr>
<tr>
<td>S-FIELD</td>
<td></td>
<td>15.05</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.07</td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.09</td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.12</td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.17</td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td>TABLE - EXTERNAL</td>
<td></td>
<td>15.10</td>
<td>15.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td>TABLE - INSTREAM</td>
<td></td>
<td>15.09</td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.16</td>
<td>16.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.00</td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.02</td>
<td>16.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18.01</td>
<td>16.14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17.00</td>
</tr>
<tr>
<td>VIRTUAL FILE</td>
<td></td>
<td>16.02</td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.04</td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.05</td>
<td>17.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.07</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.11</td>
<td>18.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18.01</td>
<td></td>
</tr>
<tr>
<td>VSAM ESDS</td>
<td></td>
<td>16.07</td>
<td>16.10</td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference

Facilities for JOB/SORT

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEFORE SORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>16.04</td>
<td>16.02</td>
</tr>
<tr>
<td>BLANK FIELDS</td>
<td>15.08</td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.08</td>
</tr>
<tr>
<td>FILE REFORMAT</td>
<td>15.08</td>
<td>16.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.11</td>
</tr>
<tr>
<td>FILE UPDATING</td>
<td>15.14</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.08</td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td>17.02</td>
</tr>
<tr>
<td>FILE-STATUS</td>
<td>16.09</td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td>PERCENT</td>
</tr>
<tr>
<td>FINISH PROCEDURE</td>
<td>15.08</td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td>16.07</td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td>15.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td>INDEXING</td>
<td>17.02</td>
<td>16.05</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td>16.14</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td>17.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.04</td>
</tr>
<tr>
<td>MACRO CALL</td>
<td>15.17</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>16.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.14</td>
<td>RECORD-COUNT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16.07</td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td>ROUNding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15.03</td>
</tr>
<tr>
<td>MULTIPLE INPUTS</td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference

**Facilities for JOB/SORT**

<table>
<thead>
<tr>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SORT</td>
<td>16.04 VSAM PATH</td>
<td>16.09</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.07 ZERO FIELDS</td>
<td>15.08</td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>START PROCEDURE</td>
<td>17.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARIANCE CALCULATION</td>
<td>18.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM FILE INPUT</td>
<td>16.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM FILE LOAD</td>
<td>16.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM FILE UPDATE</td>
<td>16.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.10</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference

Facilities for REPORT

<table>
<thead>
<tr>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/ FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAR GRAPH</td>
<td>15.07</td>
<td>DETAIL REPORT</td>
<td>15.01</td>
</tr>
<tr>
<td>BEFORE-BREAK</td>
<td>15.02</td>
<td></td>
<td>15.06</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td>15.01</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td>15.14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td>15.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td>16.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td>16.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td>16.06</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td>16.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td>16.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td>BLANK FIELDS</td>
<td>15.07</td>
<td>ENDPAGE</td>
<td>16.03</td>
</tr>
<tr>
<td>CONTROL REPORT</td>
<td>15.02</td>
<td>FORM LETTER</td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>15.04</td>
<td>17.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.10</td>
<td>17.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td>FORM REPORT</td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td>15.15</td>
<td>LEVEL</td>
<td>16.01</td>
</tr>
<tr>
<td></td>
<td>16.02</td>
<td>MAILING LABELS</td>
<td>15.03</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td>15.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.04</td>
<td>17.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16.89</td>
<td>MULTIPLE REPORTS</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>17.03</td>
<td>15.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.04</td>
<td>15.04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td>15.05</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.03</td>
<td>15.10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td>15.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>15.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18.05</td>
</tr>
</tbody>
</table>
### CA-Easytrieve/Plus Example Cross-reference
#### Facilities for REPORT

<table>
<thead>
<tr>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
<th>STATEMENT/FACILITY</th>
<th>EXAMPLE NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERCENT</td>
<td></td>
<td>SUMMARY</td>
<td></td>
</tr>
<tr>
<td>CALCULATION</td>
<td>15.02</td>
<td>REPORT</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td>15.12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>16.01</td>
</tr>
<tr>
<td>PRINTER FILE</td>
<td>17.04</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>18.01</td>
<td></td>
<td>16.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17.05</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>15.02</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td>TALLY</td>
<td>15.02</td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td>15.05</td>
</tr>
<tr>
<td></td>
<td>15.11</td>
<td></td>
<td>15.07</td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td>15.09</td>
</tr>
<tr>
<td></td>
<td>15.17</td>
<td></td>
<td>15.10</td>
</tr>
<tr>
<td></td>
<td>16.01</td>
<td></td>
<td>16.01</td>
</tr>
<tr>
<td></td>
<td>16.03</td>
<td></td>
<td>16.02</td>
</tr>
<tr>
<td></td>
<td>18.04</td>
<td></td>
<td>17.03</td>
</tr>
<tr>
<td></td>
<td>18.05</td>
<td></td>
<td>18.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>18.05</td>
</tr>
<tr>
<td>RATIO</td>
<td></td>
<td>TERMINATION</td>
<td>16.03</td>
</tr>
<tr>
<td>CALCULATION</td>
<td>15.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECORD-COUNT</td>
<td>15.08</td>
<td>TITLE VARIABLES</td>
<td>15.17</td>
</tr>
<tr>
<td>RETURN-CODE</td>
<td>18.01</td>
<td></td>
<td>18.03</td>
</tr>
<tr>
<td>ROUNDING</td>
<td>15.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUMMARY FILE</td>
<td>16.02</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Index

## A

<table>
<thead>
<tr>
<th>Item</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER-BREAK procedure</td>
<td>8-18</td>
</tr>
<tr>
<td>AFTER-LINE procedure</td>
<td>8-18, 8-20</td>
</tr>
<tr>
<td>Alphabetic literals</td>
<td>2-12</td>
</tr>
<tr>
<td>ALPHA-LIST</td>
<td>15-22</td>
</tr>
<tr>
<td>Alternate index</td>
<td>16-15</td>
</tr>
<tr>
<td>ARG, argument field</td>
<td>10-1</td>
</tr>
<tr>
<td>Arithmetic expression</td>
<td>5-2</td>
</tr>
<tr>
<td>Assignment statement</td>
<td>5-1, 9-2</td>
</tr>
<tr>
<td>arithmetic expression</td>
<td>5-2</td>
</tr>
<tr>
<td>equivalence</td>
<td>5-1</td>
</tr>
<tr>
<td>Automatic I/O</td>
<td>9-1, 15-2</td>
</tr>
<tr>
<td>Automatic input with RETRIEVE</td>
<td>11-6</td>
</tr>
<tr>
<td>Average regional gross salary</td>
<td>15-21</td>
</tr>
</tbody>
</table>

## B

<table>
<thead>
<tr>
<th>Item</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch processing</td>
<td>17-22</td>
</tr>
<tr>
<td>BEFORE-BREAK procedure</td>
<td>8-18</td>
</tr>
<tr>
<td>BEFORE-LINE procedure</td>
<td>8-18, 8-20</td>
</tr>
<tr>
<td>Blank when zero (BWZ)</td>
<td>3-12</td>
</tr>
<tr>
<td>Block Descriptor Word (BDW)</td>
<td>3-6</td>
</tr>
<tr>
<td>BWZ, (blank when zero)</td>
<td>3-12</td>
</tr>
</tbody>
</table>

## C

<table>
<thead>
<tr>
<th>Item</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARD input</td>
<td>9-4</td>
</tr>
<tr>
<td>Card punch</td>
<td>9-5</td>
</tr>
<tr>
<td>Central region employees</td>
<td>15-22</td>
</tr>
<tr>
<td>Child segment</td>
<td>11-1</td>
</tr>
<tr>
<td>Cluster combination</td>
<td>16-15</td>
</tr>
<tr>
<td>Coding techniques</td>
<td>8-17</td>
</tr>
<tr>
<td>Compile and link-edit load module</td>
<td>12-6, 13-7</td>
</tr>
<tr>
<td>COMPIL parameter</td>
<td>2-13</td>
</tr>
<tr>
<td>CONCAT macro</td>
<td>16-26</td>
</tr>
<tr>
<td>Conditional expressions</td>
<td>6-3</td>
</tr>
<tr>
<td>CONTROL statement</td>
<td>8-11</td>
</tr>
<tr>
<td>Cross-reference of statement</td>
<td>B-1</td>
</tr>
</tbody>
</table>

## D

<table>
<thead>
<tr>
<th>Item</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>11-1</td>
</tr>
<tr>
<td>Default value</td>
<td>8-8</td>
</tr>
<tr>
<td>DEFINE statement</td>
<td>3-8</td>
</tr>
<tr>
<td>Delimiters, end of statement area</td>
<td>2-11</td>
</tr>
<tr>
<td>DESC, description field</td>
<td>10-1</td>
</tr>
<tr>
<td>DISPLAY statement</td>
<td>7-2</td>
</tr>
<tr>
<td>report annotations</td>
<td>8-17</td>
</tr>
<tr>
<td>DO and END-DO statements</td>
<td>6-11</td>
</tr>
<tr>
<td>Documentation conventions</td>
<td>1-4</td>
</tr>
</tbody>
</table>
ELSE statement 6-10
Employee letters 15-5
Employees in Region 1 15-2
END-IF statement 6-10
ENDPAGE procedure 8-18
Equivalence assignment statement 5-1
Error correction 15-30
Expanded inventory report 15-27
External tables 3-7, 10-2
EZTPX01, passed parameters 16-28

GETDATE macro 16-25
GN (get next) call 11-6
GNP calls 11-6
GO TO statement 6-10
GOTO statement 6-12

Hash, report termination 8-19
HEADING statement 8-13
Hexadecimal, DISPLAY HEX 8-16

IF statement 6-9
IF, ELSE, and END-IF Statements 6-9
Indexed Sequential Access Method (ISAM) 9-6
Instream tables 10-2
INSTREAM tables 3-7
Inventory file 14-3
Inventory file update 15-33
Inventory reduction 15-31
Inventory report by city 15-26
ISAM (Indexed Sequential Access Method) 9-6
ISAM Files 9-6

JCL 13-1
JCL parameters, passing 16-28
JCL parameters, processing 16-28
JCL, passed parameters 16-28
JOB activities 4-1
    report input modification 8-20
    JOB activity 2-8, 8-2
Index – 3

**Job control setup command list (CLIST)** 17-3
**JOB statement** 4-3

**K**

Keyword
  **statement area** 2-10
Keywords 2-11

**L**

**Letter** 3-12
**Library**
  **field** 3-1
  **file** 3-1
  **record** 3-1
**LINE statement** 8-14
**Literals**
  **alphabetic** 2-12
  **numeric** 2-12

**M**

**Mailing label output program** 12-3, 13-3
**Mailing labels** 15-11
**MASK parameter** 3-12
**Mass mailings** 17-27
**Multiple statements** 2-10

**N**

**Numeric literals** 2-12

**O**

**Online processing** 17-2
**Output reports** 18-16

**P**

Parent segment 11-1
**PARM statement** 2-13
**PERFORM statement** 6-15
**PERSUPD**
  **CARD file** 12-5, 13-6
**POINT statement** 7-10
**Previously compiled and link-edited programs** 12-6, 13-7
**PRINT statement** 7-5
  **report input modification** 8-20
**Printing reports** 15-1
**Procedure** 2-8
**Procedure processing** 6-14
**Processing JCL parameters** 16-28
**PROCs (procedures)**
  **special-name procedures** 8-18
**Programs** 18-3
**Project definition** 18-1
**Project record** 18-1
**Project summary** 18-33
**Proposed salary schedules** 15-3
**Publications**
  **CA-Easytrieve/Plus** 1-3
  **related** 1-4
**PUNCH output** 9-5
**PUT statement** 7-9

**Q**

**Qualified field name** 2-12

**R**

**Random processing** 9-7
**READ statement** 7-11
Reading data files 15-1
Record 3-1
Record addition 9-10
Record deletion 9-10
Record Descriptor Word (RDW) 3-6
Record number 17-2
Record relational condition 6-9, 9-16
RECORD statement 11-3
Record update 9-11
Reformat printed output from IDCAMS 16-12
Reorder notification report 15-35
Report activities
  report procedures 8-18
REPORT activities
  report procedures 8-16
  REPORT-INPUT procedure 8-20
Report declaratives 4-1, 8-1, 8-14
Report generation 18-30
Report procedures 8-16
REPORT statement 8-5
  LEVEL 8-17
  special-name procedures 8-18
  SUMFILE 8-17
Report subactivity 2-9
Report types 8-3
REPORT-INPUT procedure 8-18, 8-20
reports
  annotations 8-20
  modifying data 8-20
Reports
  annotations 8-17
  control breaks 8-17, 8-18
RETRIEVE statement 11-4
Root segment 11-1
RPT-BY-DEPT 15-22
Rules of syntax 2-10

S
S working storage fields 8-18
Salary tally report 15-18
SAM Files 9-4
SEARCH statement 10-3
Segments 11-1
SELECT statement 8-20
Selected control break processing 16-2
SEQUENCE statement 8-11
Sequential Access Method (SAM) 9-4
Short report output program, sample 12-2, 13-2
SORT activities 4-1
SORT activity 2-9
SORT statement 4-6
Sorting input files 16-8
SORTPER
  sort output file 12-5
  sort output file (VSE) 13-6
Special report processing exits 16-5
Statement area
  keyword 2-10
Statement labels 6-13
STOP statement 6-15
  report proc warning 8-16
Structure 2-7
subprograms
  EZTPX01 16-28
Summary file processing 16-3
Synchronized file facility, file update 16-10
Synchronized file processing 9-11
Synchronized file processing program 12-4, 13-5
Syntax check 2-14
SYNTAX parameter 2-13
Table definition 10-1
Tables
   external 3-7
   INSTREAM 3-7
Tally reports 15-13
Task record 18-1
TERMINATION procedure 8-18
TITLE statement 8-12

Updating a VSAM ESDS file 16-22

VALUE parameter 3-14
VFM Files 9-5
VFM, Virtual File Manager 9-5
Virtual File Manager (VFM) 9-5
Virtual Storage Access Method (VSAM) 9-7
VSAM file processing 16-15
VSAM Files 9-7

Women's phone numbers 15-17
Words
   as field names 2-11
   as keywords 2-11
   as literals 2-11
   delimiters 2-11
   statements 2-11
Working storage
   report procedure fields 8-17
   S fields 8-18
WRITE statement 7-12