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
DUPUCATED, 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.

rights reserved.

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

## 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 .................................................. 1–5
  - Variable Parameters .......................................................... 1–6
  - Flowchart Symbols ............................................................ 1–7

## Chapter 2: General Information

- Introduction ............................................................................ 2–1
  - Environment ...................................................................... 2–1
- Capabilities ............................................................................ 2–1
  - File Access ........................................................................ 2–2
  - Character Set Support ....................................................... 2–2
  - Field Definition .................................................................. 2–3
  - Logic Process ..................................................................... 2–3
  - File Output .......................................................................... 2–4
  - Report Output ..................................................................... 2–4
  - Virtual File Manager .......................................................... 2–5
  - Debugging Capabilities ....................................................... 2–5
- Current Technology ............................................................... 2–6
- Summary of Revisions .......................................................... 2–6
  - Expanded SQL Processing .................................................. 2–6
  - IF BREAK/HIGHEST BREAK Class Tests ............................... 2–6
  - RETURN-CODE for VSE ...................................................... 2–6
  - CA-Librarian Macro Support ............................................... 2–7
  - New CONTROLSKIP Parameter on the REPORT Statement .... 2–7
  - Read Access to LINE-COUNT and PAGE-COUNT ................ 2–7
  - RESET Parameter on FILE Statement ................................. 2–7
Default Disk Type (VSE Only) .......................................................... 2–7
EVEN Parameter on DEFINE Statement ............................................. 2–7
HEADING Parameter in the SQL INCLUDE Statement ....................... 2–7
WITH HOLD Option on SQL DECLARE CURSOR Statement (DB2 Only) .. 2–8
CONNECT Statement Improves Data Access ..................................... 2–8
SQLSYNTAX Parameter on PARM Statement ................................... 2–8
PLAN Parameter on PARM Statement (DB2 Only) .......................... 2–8
PLANOPTS Parameter on PARM Statement (CA-Datacom SQL only) ... 2–8
SELECT INTO Support (DB2 Only) ................................................... 2–8
DB2 - DL/I Batch Support ............................................................. 2–9
IDD Updating ............................................................................. 2–9
IDMS RETURN KEY Support ......................................................... 2–9
SQL with CA-IDMS ................................................................... 2–9
CA-Datacom SQL Support .............................................................. 2–9
Year 2000 Support .................................................................... 2–10

Chapter 3: System Overview

Modes of Operation ........................................................................ 3–1
Structure of a Program ..................................................................... 3–1
  Environment Definition Section .................................................. 3–2
  Library Definition Section .......................................................... 3–2
  Activity Definition Section ......................................................... 3–2
  Program Flow .......................................................................... 3–5
Syntax Rules .................................................................................. 3–7
  Statement Area ........................................................................ 3–7
  Character Sets ...................................................................... 3–8
  Multiple Statements ................................................................. 3–10
  Comments ............................................................................ 3–10
  Continuations ...................................................................... 3–11
  Words and Delimiters ............................................................... 3–12
  Keywords ............................................................................. 3–12
  Multiple Parameters ................................................................. 3–12
  Field Names ....................................................................... 3–13
  Labels ................................................................................ 3–13
  EBCDIC Alphabetic Literals ....................................................... 3–13
  Numeric Literals .................................................................. 3–14
  EBCDIC Format Hexadecimal Literals ......................................... 3–14
  DBCS Format Literals ............................................................... 3–14
  DBCS Format Hexadecimal Literals .......................................... 3–15
  MIXED Format Literals ............................................................ 3–15
Alphabetic Conversion Literals ............................................................ 3–15
Katakana Conversion Literals .............................................................. 3–16
Hiragana Conversion Literals .............................................................. 3–17
User Converted Literals ................................................................. 3–17
Format and Conversion Rules .................................................. 3–18
Format Relationship Rules ...................................................... 3–20
Identifiers ................................................................................. 3–21
Arithmetic Operators ................................................................. 3–21

Chapter 4: Environment Definition
PARM Statement.................................................................................. 4–1
  Environment Modification .................................................................... 4–1
  Usage ................................................................................................. 4–1
PARM Statement Syntax ....................................................................... 4–2
PARM Statement Parameters ........................................................... 4–5
PARM Statement Examples .............................................................. 4–17

Chapter 5: Data Definition
Defining Data .................................................................................. 5–1
  File Attributes ............................................................................... 5–2
  Field Data ........................................................................................ 5–2
  COPY Statement ............................................................................. 5–2
  Data Reference ............................................................................... 5–2
FILE Statement .................................................................................. 5–3
  Syntax ................................................................................................. 5–3
FILE Statement Parameters ........................................................... 5–4
  File Types ........................................................................................ 5–5
  Device Types .................................................................................. 5–9
  Record Format ................................................................................ 5–11
  Examples ......................................................................................... 5–14
DEFINE Statement ............................................................................. 5–15
  DEFINE within an Activity ............................................................. 5–15
  File Fields ......................................................................................... 5–15
  Working Storage Fields ............................................................. 5–16
    Basic Field Definition ............................................................... 5–16
  DEFINE Syntax ............................................................................ 5–17
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>5–18</td>
</tr>
<tr>
<td>Location</td>
<td>5–18</td>
</tr>
<tr>
<td>Attributes</td>
<td>5–19</td>
</tr>
<tr>
<td>Data Format</td>
<td>5–19</td>
</tr>
<tr>
<td>DEFINE Statement Examples</td>
<td>5–25</td>
</tr>
<tr>
<td>Record Layouts</td>
<td>5–25</td>
</tr>
<tr>
<td>Overlay Redefinition</td>
<td>5–26</td>
</tr>
<tr>
<td>Working Storage Initialization</td>
<td>5–26</td>
</tr>
<tr>
<td>Varying Length Fields</td>
<td>5–26</td>
</tr>
<tr>
<td>Displaying Varying Length Fields</td>
<td>5–27</td>
</tr>
<tr>
<td>Assigning and Moving Varying Length Fields</td>
<td>5–27</td>
</tr>
<tr>
<td>Alternate Report Headings</td>
<td>5–28</td>
</tr>
<tr>
<td>Signed/Unsigned Rules</td>
<td>5–28</td>
</tr>
<tr>
<td>Signed</td>
<td>5–28</td>
</tr>
<tr>
<td>Unsigned</td>
<td>5–29</td>
</tr>
<tr>
<td>Edit Masks</td>
<td>5–29</td>
</tr>
<tr>
<td>Editing Rules</td>
<td>5–30</td>
</tr>
<tr>
<td>Leading Zeros</td>
<td>5–30</td>
</tr>
<tr>
<td>Printing</td>
<td>5–30</td>
</tr>
<tr>
<td>Suppressing</td>
<td>5–30</td>
</tr>
<tr>
<td>Replacing</td>
<td>5–31</td>
</tr>
<tr>
<td>Negative Numbers</td>
<td>5–31</td>
</tr>
<tr>
<td>COPY Statement</td>
<td>5–31</td>
</tr>
<tr>
<td>Syntax</td>
<td>5–31</td>
</tr>
<tr>
<td>Example</td>
<td>5–32</td>
</tr>
<tr>
<td>COPY Rules for Database Files</td>
<td>5–32</td>
</tr>
<tr>
<td>Examples</td>
<td>5–33</td>
</tr>
<tr>
<td>Data Reference</td>
<td>5–33</td>
</tr>
<tr>
<td>Unique Name</td>
<td>5–33</td>
</tr>
<tr>
<td>Qualification</td>
<td>5–34</td>
</tr>
<tr>
<td>Indexing</td>
<td>5–34</td>
</tr>
<tr>
<td>System-Defined Fields</td>
<td>5–34</td>
</tr>
<tr>
<td>General Fields</td>
<td>5–34</td>
</tr>
<tr>
<td>File Fields</td>
<td>5–35</td>
</tr>
<tr>
<td>Report Fields</td>
<td>5–36</td>
</tr>
</tbody>
</table>
Chapter 6: Processing Activities

Introduction ........................................................................................................... 6–1
JOB Activities ........................................................................................................ 6–1
SORT Activities ..................................................................................................... 6–1
JOB Statement ....................................................................................................... 6–1
Syntax ..................................................................................................................... 6–1
Parameters ............................................................................................................. 6–2
Job Flow ................................................................................................................ 6–4
SORT Statement .................................................................................................... 6–7
Syntax ..................................................................................................................... 6–7
Parameters ............................................................................................................. 6–8
Sorting Files .......................................................................................................... 6–14
Sort Procedures .................................................................................................... 6–15
Sort Flow ................................................................................................................ 6–16
SELECT Statement ............................................................................................... 6–18
Syntax ..................................................................................................................... 6–18
Sorting a Selected Portion of a File ................................................................. 6–18

Chapter 7: Assignments and Moves

Introduction ........................................................................................................... 7–1
Arithmetic Expressions ....................................................................................... 7–1
Syntax ..................................................................................................................... 7–1
Operation ................................................................................................................. 7–2
Parentheses .......................................................................................................... 7–2
Evaluations ........................................................................................................... 7–3
Assignment Statement .......................................................................................... 7–5
Format 1 (Normal Assignment) ........................................................................ 7–6
Format 2 (Logical Expression) ........................................................................... 7–8
Examples ............................................................................................................... 7–8
EBCDIC To DBCS Conversion ............................................................................ 7–10
Format 1 (Normal Assignment) ........................................................................ 7–11
MOVE Statement .................................................................................................. 7–15
Syntax ..................................................................................................................... 7–15
Variables ................................................................................................................ 7–16
Example ................................................................................................................ 7–17
MOVE LIKE Statement ......................................................................................... 7–17
Syntax ..................................................................................................................... 7–17
Operation ............................................................................................................... 7–17
Example ................................................................................................................ 7–18
Chapter 8: Decision and Branching Logic

Introduction ................................................................................ 8–1
Conditional Expressions ................................................................ 8–1
  Simple Conditions ....................................................................... 8–2
  Extended Conditions ................................................................. 8–2
  DBCS Considerations ................................................................. 8–3
  Varying Length Fields ................................................................. 8–3
  Combined Conditions ................................................................. 8–3
Field Relational Condition ............................................................ 8–4
  Syntax ................................................................................... 8–4
  Operation ............................................................................... 8–4
  Alphanumeric Subjects .............................................................. 8–5
  Numeric Subjects ..................................................................... 8–5
  Mixed Subjects ........................................................................ 8–5
  DBCS Subjects .......................................................................... 8–6
  Example .................................................................................. 8–7
Field Series Condition ................................................................. 8–7
  Operation ............................................................................... 8–8
  Rules for Evaluation ................................................................ 8–8
  Example .................................................................................. 8–9
Field Class Condition ................................................................... 8–9
  Syntax ................................................................................... 8–9
  Operation ............................................................................... 8–10
  Rules for Evaluation ................................................................ 8–10
  Example .................................................................................. 8–12
Field Bits Condition ..................................................................... 8–13
  Syntax ................................................................................... 8–13
  Operation ............................................................................... 8–13
  Example .................................................................................. 8–14
File Presence Condition .............................................................. 8–14
  Syntax ................................................................................... 8–14
  Operation ............................................................................... 8–14
  Example .................................................................................. 8–15
File Presence Series Condition ..................................................... 8–16
  Syntax ................................................................................... 8–16
  Operation ............................................................................... 8–16
  Example .................................................................................. 8–16
Record Relational Condition ......................................................... 8–17
  Syntax ................................................................................... 8–17
  Operation ............................................................................... 8–17
  Example .................................................................................. 8–18
## IF, ELSE-IF, ELSE, and END-IF Statements
- Syntax ................................................................. 8–18
- Operation ............................................................... 8–19
- Example ............................................................... 8–20

## DO and END-DO Statements
- Syntax ................................................................. 8–22
- Operation ............................................................... 8–22
- Nesting ................................................................. 8–23
- Example ............................................................... 8–24

## CASE Statement
- Syntax ................................................................. 8–25
- Operation ............................................................... 8–25
- Example ............................................................... 8–27

## GOTO Statement
- Syntax ................................................................. 8–27
- Operation ............................................................... 8–27
- Example ............................................................... 8–28

## Statement Label
- Syntax ................................................................. 8–28
- Operation ............................................................... 8–29
- Example ............................................................... 8–30

## PERFORM Statement
- Syntax ................................................................. 8–31
- Operation ............................................................... 8–31
- Example ............................................................... 8–31

## STOP Statement
- Syntax ................................................................. 8–32
- Operation ............................................................... 8–32
- Example ............................................................... 8–32
- Conditional Execution .............................................. 8–32
- Example ............................................................... 8–33
- Termination ........................................................... 8–33

### Chapter 9: Input/Output Specification

- Introduction ......................................................... 9–1
- STATUS Parameter .................................................. 9–1
- DISPLAY Statement ............................................... 9–2
  - Syntax ............................................................... 9–2
  - Example ............................................................... 9–7
Chapter 10: Procedure Processing

Introduction .......................................................... 10–1
Procedure Syntax ................................................. 10–1
Syntax ................................................................. 10–2
Operation ............................................................ 10–2
Invoking Procedures .............................................. 10–3
PERFORM Statement ............................................. 10–3
Syntax ................................................................. 10–3
START and FINISH Parameters (JOB) ................. 10–4
BEFORE Parameter (SORT) .................................. 10–4
Special-name Report Procedures ......................... 10–5
Procedure Placement ............................................ 10–5
Example ............................................................ 10–6
Chapter 11: Report Processing

Introduction ................................................................. 11–1
  Basic Report Structure .................................................. 11–2
  PRINT Statement Processing ...................................... 11–2
  PRINT Workfile Processing ......................................... 11–3
  Report Formats ......................................................... 11–4
  Standard Format ....................................................... 11–4
  Label Format .......................................................... 11–5
  DBCS Reporting Restrictions ...................................... 11–5
  Report Definition Statements ................................... 11–6
  Structure of Report Definition .................................. 11–7
REPORT Statement ..................................................... 11–7
  Syntax ........................................................................ 11–8
REPORT Statement Parameters .................................. 11–9
  Format Determination Parameters ............................. 11–9
  File Directing Parameters ........................................ 11–12
  Spacing Control Parameters ...................................... 11–12
  Testing Aid Parameters ............................................. 11–17
  IMS-only Parameter ................................................ 11–17
  DBCS/MIXED Considerations ..................................... 11–17
  Varying Length Field Considerations ......................... 11–18
Report Definition Statements ...................................... 11–18
  SEQUENCE Statement .............................................. 11–18
  Syntax ........................................................................ 11–18
  Operation .................................................................... 11–19
  CONTROL Statement ................................................. 11–24
  SUM Statement ........................................................ 11–25
  TITLE Statement ....................................................... 11–26
  HEADING Statement .................................................. 11–29
  LINE Statement ....................................................... 11–30
Standard Reports ......................................................... 11–32
  Titles ........................................................................... 11–32
  Headings ..................................................................... 11–34
  Line Group .................................................................. 11–35
  Line Item Positioning ............................................... 11–36
  Special Positioning .................................................... 11–37
  Pre-printed Form Production ..................................... 11–38
  SPREAD Option ......................................................... 11–39
Label Reports .............................................................. 11–40
  CONTROL Statement ................................................. 11–41
Sequenced Reports ................................................................. 11–41
SEQUENCE Statement .......................................................... 11–41
CONTROL Reports ................................................................. 11–42
Terminology ........................................................................ 11–42
Data Reference ..................................................................... 11–43
TALLY .................................................................................. 11–43
LEVEL .................................................................................. 11–44
BREAK-LEVEL ...................................................................... 11–44
IF BREAK/IF HIGHEST BREAK Processing ...................... 11–45
Control Report Contents ...................................................... 11–45
DTLCTL .............................................................................. 11–48
SUMCTL .............................................................................. 11–50
DTLCOPY ............................................................................ 11–52
DTLCOPYALL ...................................................................... 11–54
Control Field Values in Titles ............................................. 11–54
Overflow of Total Values ................................................... 11–56
Controlling Overflow ......................................................... 11–57
Summary File ...................................................................... 11–59
Report Procedures ............................................................... 11–60
Coding Techniques .............................................................. 11–61
Special-name Report Procedures ...................................... 11–63
REPORT-INPUT ................................................................. 11–64
BEFORE-LINE and AFTER-LINE ........................................ 11–66
BEFORE-BREAK .................................................................. 11–68
AFTER-BREAK .................................................................... 11–69
ENDPAGE ............................................................................ 11–70
TERMINATION .................................................................... 11–71
Report Work Files .............................................................. 11–72
Routing Printed Output ....................................................... 11–73

Chapter 12: File Processing
Introduction ......................................................................... 12–1
Control of Input/Output ....................................................... 12–1
Data Access Modes ............................................................. 12–2
Record Format ................................................................. 12–2
Record Addressability ....................................................... 12–3
System-Defined File Fields .............................................. 12–3
Error Conditions .............................................................. 12–4
Data Availability Tests ..................................................... 12–5
Opening and Closing Files ............................................... 12–5
Chapter 13: Table and Array Processing

Introduction ................................................................. 13–1
Table Definition ............................................................ 13–1
Defining Tables ............................................................. 13–2
SEARCH Statement ....................................................... 13–3
Syntax .............................................................. 13–3
Operation ............................................................ 13–3
Searching Tables ......................................................... 13–5
Single Dimension Arrays ............................................... 13–5
Index Attribute .......................................................... 13–5
Example .............................................................. 13–6
Multiple Dimension Arrays ............................................. 13–7
### Subprograms

- Programming Languages .......................................................... 14–1
- Program Loading ...................................................................... 14–2
- Storage Management .................................................................. 14–3
- Linkage (Register Usage) Conventions ..................................... 14–3
  - Linkage Register Usage ......................................................... 14–3
  - Register Save Area Usage ..................................................... 14–3
  - Assembler Subprogram Linkage ............................................ 14–4
  - COBOL Subprogram Linkage ................................................ 14–5
  - VS FORTRAN Subprogram Linkage ....................................... 14–7
  - PL/I Subprogram Linkage ...................................................... 14–7
- Parameter Lists ........................................................................ 14–8
  - Parameter List Format .......................................................... 14–8
  - Exit Parameter List ................................................................ 14–8
- CALL Statement ...................................................................... 14–11
  - Syntax .................................................................................. 14–11
  - Operation .............................................................................. 14–11
- CALL Parameter Lists ............................................................... 14–12
- LE-Enabled CA-Easytrieve/Plus ................................................ 14–12
  - CA-Easytrieve/Plus Calling COBOL ...................................... 14–13
  - CA-Easytrieve/Plus Calling Assembler ................................. 14–13
  - CA-Easytrieve/Plus Calling LE CEExxxx Routines ............... 14–13
  - CA-Easytrieve/Plus Calling PL/I .......................................... 14–13
  - File Exit Programs ................................................................ 14–14
- Error Condition Handling ......................................................... 14–14

### System Facilities

- Introduction .............................................................................. 15–1
- Compiler Directives .................................................................. 15–2
  - End ...................................................................................... 15–2
  - Syntax .................................................................................. 15–2
Chapter 16: Macros

Introduction ........................................................................................................ 16–1
Macro Invocation Statement .............................................................................. 16–1
  Syntax ........................................................................................................... 16–1
  Invoking Macros ......................................................................................... 16–2
Macro Library .................................................................................................... 16–3
  Macro Library Security ............................................................................... 16–3
Macro Definition .................................................................................................. 16–5
  Prototype Statement .................................................................................... 16–6
  Syntax .......................................................................................................... 16–6
  Operation ...................................................................................................... 16–6
Positional Parameters ................................................................. 16–7
Keyword Parameters ..................................................................... 16–7
Prototype Examples ................................................................. 16–7
Macro Body .................................................................................. 16–8
Macro Termination Command .................................................. 16–8
Instream Macros ........................................................................... 16–8
Syntax ....................................................................................... 16–9
Operation .................................................................................... 16–9
Example ....................................................................................... 16–10
Macro Processing ........................................................................... 16–10
Parameter Substitution ............................................................... 16–11
Examples ..................................................................................... 16–11
'&' and '.' in a Macro ............................................................... 16–12

Appendix A: Diagnostics
Introduction .................................................................................. A–1
Diagnostic Message Format ...................................................... A–1
Message ID .................................................................................. A–1
Diagnostic Message ................................................................. A–1
Message Supplement ............................................................... A–2
Operational Diagnostic Messages ............................................. A–2
Syntax Diagnostic Messages ...................................................... A–9
DQSCGEN Messages ................................................................. A–44
IDD Interface Messages ............................................................ A–46
SQL Supplemental Diagnostic Messages ................................. A–48

Appendix B: Keywords
Introduction ................................................................................ B–1
Symbol References ........................................................................ B–1
Reserved Words .......................................................................... B–3

Appendix C: Options Table
Introduction ................................................................................ C–1
Selectable Options ......................................................................... C–1
Appendix D: Examples

Introduction ............................................................................................................. D–1
Coding Conventions .............................................................................................. D–1
Inventory Sample File .......................................................................................... D–2
Personnel Sample File .......................................................................................... D–3
GETDATE Macro ..................................................................................................... D–11
Basic Examples ..................................................................................................... D–12
  Example 1 - Employees in Region 1 ................................................................. D–12
  Example 2 - Proposed Salary Schedules ............................................................ D–13
  Example 3 - Employee Letters ......................................................................... D–15
  Example 4 - Mailing Labels ............................................................................. D–19
  Example 5 - Tally Reports ................................................................................ D–21
  Example 6 - Phone Number Selection .............................................................. D–26
  Example 7 - Salary Tally Report ....................................................................... D–27
  Example 8 - File Expansion ............................................................................. D–29
  Example 9 - Average Regional Gross Salary .................................................... D–31
  Example 10 - Central Region Employees .......................................................... D–33
  Example 11 - Inventory Report by City ............................................................. D–37
  Example 12 - Expanded Inventory Report ........................................................ D–39
  Example 13 - Error Correction ........................................................................ D–42
  Example 14 - Inventory Reduction ................................................................. D–43
  Example 15 - Inventory File Update ................................................................. D–46
  Example 16 - Reorder Notification Report ........................................................ D–49
CA-Easytrieve/Earl ............................................................................................... D–55
JCL Examples for the OS/390 Operating System ................................................ D–57
  JCL Examples: OS/390 JCL Systems ................................................................. D–63
JCL Examples for the VSE Operating System ...................................................... D–67
  JCL Examples: VSE JCL Systems ..................................................................... D–71

Appendix E: CA-Easytrieve Year 2000 Support

Introduction ........................................................................................................... E–1
  Obtaining a Four-Digit System Date ................................................................. E–1
  Printing a Four-Digit Year on Reports ............................................................. E–1
DATECONV ............................................................................................................. E–2
  Syntax ................................................................................................................ E–2
    Parameters ...................................................................................................... E–2
    Operation - INLINE ....................................................................................... E–2
    Operation - Database .................................................................................... E–3
  Example ............................................................................................................. E–3
INDEX

DATEVAL .................................................. E-4
Syntax ..................................................... E-4
Parameters ............................................. E-4
Operation - INLINE ................................... E-5
Operation - Database .............................. E-6
Example .................................................. E-6
Chapter 1

About This Guide

Purpose and Audience

This Reference Guide is designed for use by you, the CA-Easytrieve/Plus programmer. It assumes that you are familiar with the CA-Easytrieve language and understand basic data processing concepts.

Use this guide with CA-Easytrieve/Plus, Version 6.2.

Organization

This guide is divided into several chapters:

- “General Information” introduces you to CA-Easytrieve/Plus and its capabilities. It also lists the environments under which CA-Easytrieve/Plus executes, and contains version summaries for the latest releases of the product.
- “System Overview” examines the modes in which CA-Easytrieve/Plus operates, the structure of a CA-Easytrieve/Plus program, and the syntax rules of CA-Easytrieve/Plus statements.
- “Environment Definition” discusses the PARM statement and how you can use it to customize the operating environment for the duration of a program's compilation and execution.
- “Data Definition” describes the methods of defining data.
- “Processing Activities” discusses the two CA-Easytrieve/Plus processing activities: JOB and SORT.
- “Assignments and Moves” discusses the ways to move or manipulate data within your program using the Assignment, MOVE, and MOVE LIKE statements.
- “Decision and Branching Logic” describes the CA-Easytrieve/Plus statements that control the execution of your program by means of decision and branching logic.
“Input/Output Specification” presents the CA-Easytrieve/Plus statements used for controlled input and output processing. These statements cause an input or output action to occur.

“Procedure Processing” describes procedures and shows you how to invoke them in your program.

“Report Processing” discusses report processing using the PRINT statement.

“File Processing” describes the processing of sequential, random, multiple input, and Host Disk files.

“Table and Array Processing” discusses the CA-Easytrieve/Plus facilities for processing the two basic categories of tabular information: file-oriented table information processed by SEARCH, and arrays, segmented data, and data strings processed using subscripting or index manipulation.

“Subprograms” describes the techniques used to invoke subprograms of other languages.

“System Facilities.” discusses the format of compilation listing options, execution listings, and the abnormal termination facilities of CA-Easytrieve/Plus.

“Macros” describes the CA-Easytrieve/Plus macro facility.

“Diagnostics” lists CA-Easytrieve/Plus diagnostic messages with brief explanations for each.

“Keywords” contains a list of CA-Easytrieve/Plus symbols and reserved words.

“Options Table” lists the basic parameters that control CA-Easytrieve/Plus operation.

“Examples” illustrates how to use CA-Easytrieve/Plus to solve a variety of basic data processing problems.

“CA-Easytrieve Year 2000 Support” describes features to permit you to access the system date in a four-digit year format and to have the full year specified on your reports.

The Index provides listings to facilitate references to screens and tables, terms and procedures.
### Other CA-Easytrieve/Plus Publications

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<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 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 Application Guide</td>
<td>Describes basic syntax (a subset of the syntax in the CA-Easytrieve/Plus Reference Guide) and operation, and provides a series of actual applications, from single examples to full systems. The Application Guide is an excellent tool for the business-oriented professional.</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>
</tbody>
</table>
Related Publications

The following publications, produced by Computer Associates, are either referenced in this documentation or are recommended reading:

- CA-Pan/SQL SQL Interface Installation Guide
- CA-Corporate Tie Installation Guide
- CA-Panvalet System Management Guide
- CA-Librarian Command Reference (Batch) Manual
- CA-Librarian Lock Facility Manual

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

- IBM VSE/VSAM Messages and Codes Manual
- IBM DOS Supervisor and I/O Macro Manual
- IBM OS Data Management Macro Instructions Manual
- IBM VSAM Programmer's Guide
- DBMS Programmer's Guide
- CMS Command and Macro Reference Manual
- SQL/DS Messages and Codes for VSE (SH24-5019)
- SQL/DS Messages and Codes for VM/SP (SH24-5070)
- IBM Database 2 Messages and Codes (SC26-4113)
- ORACLE Error Messages and Codes Manual (3605)
- IBM SQL Message Manuals
- IBM Program Description and Operation Manual
- FACOM Sort/Merge Program Description Manual
- IBM Sort/Merge Program - Kanji/Chinese Manual
### Documentation Conventions

The following conventions are used throughout this guide 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></td>
<td>(OR bar)</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.</td>
</tr>
<tr>
<td></td>
<td>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>A data field defined in your program.</td>
</tr>
<tr>
<td>file-name</td>
<td>A 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>A numeric literal (a whole number greater than zero).</td>
</tr>
<tr>
<td>job-name</td>
<td>Name of a JOB activity.</td>
</tr>
<tr>
<td>letter</td>
<td>A single alphabetic character (such as an edit mask identifier).</td>
</tr>
<tr>
<td>literal</td>
<td>A 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 a 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>
Flowchart Symbols

The following flowchart symbols illustrate the same concepts throughout this manual.

- **input/output**
- **document**
- **process annotation**
- **terminal**
- **punch card**
- **decision point**
- **magnetic tape**
- **connector**
- **online storage**
Chapter 2

General Information

Introduction

CA-Easytrieve/Plus is an information retrieval and data management system designed to simplify computer programming. Its English-like language and simple declarative statements provide the new user with the tools needed to produce comprehensive reports with ease, while its enhanced facilities provide the experienced data processor with the capabilities to perform complex programming tasks.

Environment

CA-Easytrieve/Plus operates on the IBM 370, 30xx, 43xx, and compatible processors in the VSE/SP, VSE/ESA, MVS/SP, MVS/ZA, MVS/ESA, or VM/CMS environments. Under TSO, CMS, and ICCF, CA-Easytrieve/Plus can run interactively for data inquiry, analysis, and reporting. The output can be either returned back to your terminal screen or routed to a printer.

Capabilities

The extensive capabilities of CA-Easytrieve/Plus provide you with more benefits than those available in standard information retrieval systems. CA-Easytrieve/Plus has the capabilities of a retrieval system as well as the comprehensiveness and flexibility required for complex reports, data extraction and file maintenance requirements.
Capabilities

File Access

The file access features of CA-Easytrieve/Plus provide all standard retrieval system capabilities, plus the following:

- Accepts up to 890 input or output files.
- Synchronizes file processing (based on keys) of an unlimited number of files, including matched conditions and duplicate checking. This reduces complex matching logic down to one statement.
- Tests for file availability and current record count.
- Prints statistics on files used, including number of records processed and attributes of each file.
- Provides in-core binary search of external or instream table files.
- Prints file status and error analysis report at point of error during abnormal termination.
- Provides an easy method for establishing temporary work files without special job control or file allocation statements.

Character Set Support

CA-Easytrieve/Plus supports both Expanded Binary Coded Decimal Interchange Code (EBCDIC) and Double Byte Character Set (DBCS) character representations. EBCDIC is a character set system that associates one character with a particular 8-bit binary value. This means that each EBCDIC character occupies one unit of storage; therefore EBCDIC is termed as a single-byte character set. EBCDIC supports the definition of 256 different characters.

For languages based on the Chinese system of writing, the character set may include 50,000 different characters. To represent such a character set, each character is associated with a particular 16-bit binary number. This means that two units of storage are occupied to represent a single character and thus the name Double Byte Character Set.
Field Definition

The methods CA-Easytrieve/Plus uses define all types of record structures and field formats are consistent and easy to use, including:

- Defining all field formats, including binary and unsigned packed fields.
- Supporting alphanumeric field types containing both EBCDIC and DBCS format data plus a MIXED field type for those fields that contain a mixture of both EBCDIC and DBCS format characters.
- Providing flexible edit masks for report formats or displaying data, including blank-when-zero, automatic DBCS conversion, and hex display.
- Establishing EBCDIC, DBCS, and MIXED initial values for working storage fields.
- Providing default report headings to enhance standards.
- Allowing multiple use of field definitions with the COPY keyword, reducing coding and maintenance.

Logic Process

The purpose of any information retrieval and application development system is to provide complete conditional logic. CA-Easytrieve/Plus provides this logic, plus the following:

- Provides standard programming constructions, such as nested IFs, DO WHILE, and PERFORM statements.
- Provides powerful calculation capabilities, including bit manipulation.
- Performs special tests useful in editing, including alphabetic, numeric, spaces, zero, and test under mask.
- Enables string manipulation.
- Supports move for corresponding fields.
- Includes special one-time procedures for start of processing and finish of processing.
- Sorts on any number of keys.
Capabilities

File Output

Routine file maintenance is faster and simpler because of the enhanced capabilities of CA-Easytrieve/Plus including:

- Loading and updating files, including VSAM, IMS/DLI, CA-IDMS, and SQL.
- Saving report extract work files for subsequent use.
- Providing a selective hex dump of a file or specific fields.

Report Output

The reporting features of CA-Easytrieve/Plus make producing reports a simple, uncomplicated process. The flexibility built into the system through specialized report procedures makes it easy to produce customized reports without compromise. CA-Easytrieve/Plus:

- Produces unlimited reports from a single pass of the data.
- Automatically formats reports including where character output sizes vary due to different data types (EBCDIC and DBCS formats) and font specifications.
- Provides customizing alternatives to all report format features.
- Provides mailing labels of any size.
- Provides control breaks on any number of keys.
- Automatically creates a summary file containing subtotals.
- Processes only those fields that are required by your REPORT statements.
- Generates reports to separate logical printers or other output media.
- Provides control break level access for special logic processing, which is useful when only certain report lines are to be generated for certain specific levels of control breaks.
- Provides specialized report procedures for user flexibility (BEFORE/AFTER LINE, ENDPAGE, TERMINATION, BEFORE/AFTER BREAK, REPORT INPUT).
- Allows for explicit positioning of print layout for pre-printed forms.
Virtual File Manager

VFM provides an easy method for establishing temporary work files without special job control or file allocation statements. By using VFM, you can establish your own extract or temporary files using only CA-Easytrieve/Plus keywords. VFM's own data management techniques ensure its operating efficiency standards, including:

- Maintaining more information in memory. If the memory area is exhausted, VFM writes the excess data to a single spill area.
- Defining only one physical file.
- Determining the best blocking factor based on device type, providing a 90 percent disk utilization.
- Releasing and recovering occupied space as the virtual file is read back into your program.
- Automatically spooling files created as a result of sequenced reports or multiple reports in the same activity.

Debugging Capabilities

The debugging aids of CA-Easytrieve/Plus ensure that all information necessary to pinpoint the cause of an abnormal termination is easily readable by:

- Providing an error analysis report which pinpoints most errors immediately including the source statement number in error and/or a FLOW table of what statements were executed in what series.
- Providing optional data processing oriented displays, such as DMAPs and PMAPs.
- Trapping invalid file references during execution to prevent a system dump.
Summary of Revisions

Current Technology

CA-Easytrieve/Plus represents the maximum in efficiency because it has been developed with the latest in programming technology, including:

- Mapping programs in 4K segments.
- Mapping working storage on double word boundary.
- Providing a one-pass compiler.
- Directly generating the object code.
- Providing PUSH/POP facilities for MACRO.
- Providing security on VSAM, IMS, and CA-IDMS usage.
- Automatic EBCDIC to DBCS conversion facilities plus user exits for implementing Phonetic Translation routines.

Summary of Revisions

Expanded SQL Processing

The CA-Easytrieve/Plus SQL interface has been expanded to:

- Support CA-IDMS/SQL and CA-Datacom/SQL. This provides complete SQL access for retrieval and maintenance of databases.
- Support of DB2 version 3.1 SQL commands.
- Support of SQL/DS version 3.2 SQL commands.

IF BREAK/HIGHEST BREAK Class Tests

New IF BREAK/HIGHEST BREAK class tests can be used as alternatives to IF LEVEL testing in report break procedures.

RETURN-CODE for VSE

RETURN-CODE is a system-defined four-byte binary field. When set in a CA-Easytrieve/Plus program, the contents of RETURN-CODE are returned to the operating system when the program completes. This facilitates conditional execution of subsequent JCL steps.
CA-Librarian Macro Support

The CA-Easytrieve/Plus Macro Facility now enables you to store and retrieve macros directly to or from CA-Librarian.

New CONTROLSKIP Parameter on the REPORT Statement

For use with CONTROL reports, the value of the CONTROLSKIP parameter on the REPORT statement determines the number of blank lines to be inserted between total lines and the next detail line.

Read Access to LINE-COUNT and PAGE-COUNT

CA-Easytrieve/Plus now provides the capability to perform logic based on the contents of the system defined fields, PAGE-COUNT and LINE-COUNT. The most common use is in report procedures whenever automatic report printing of CA-Easytrieve/Plus is being augmented by DISPLAY statements.

RESET Parameter on FILE Statement

A new RESET parameter has been added to the FILE statement to use with the overwrite feature in version 2.1 of CA-Corporate Tie. RESET forces a file of the same name that exists on the Host Disk to be overwritten.

Default Disk Type (VSE Only)

The default type of disk has been changed from 3340 to 3380. This value is used when DISK is specified for site options VFMDEV, DEVICE, or MACDEV, or when you specify the DISK option on a FILE statement.

EVEN Parameter on DEFINE Statement

The new EVEN parameter on the DEFINE statement indicates that a packed decimal field is to contain an even number of digits.

HEADING Parameter in the SQL INCLUDE Statement

The new HEADING parameter on the SQL INCLUDE statement automatically causes the remarks in the DBMS system catalog entry for a column to be copied into a HEADING parameter of the generated DEFINE statement.
WITH HOLD Option on SQL DECLARE CURSOR Statement (DB2 Only)

Support has been added for the WITH HOLD option on the SQL DECLARE CURSOR statement. This permits the application program to monitor cursor positioning across SQL commits.

CONNECT Statement Improves Data Access

Use of the CONNECT statement with IBM's Distributed Relational Database Architecture (DRDA) enables a CA-Easytrieve/Plus program running on the mainframe to access, update, and report on data at multiple remote relational database management systems within a single job.

SQLSYNTAX Parameter on PARM Statement

You can specify an SQLSYNTAX value of NONE and a BIND option of STATIC-ONLY to bypass the dynamic prepares of your SQL statement, along with the DB2 authorization checking at compile time. SQL statements are then syntax checked by the DB2 preprocessor.

For all databases, an SQLSYNTAX value of PARTIAL can be specified to perform gross-level syntax checking in the case where your DBMS is not available.

PLAN Parameter on PARM Statement (DB2 Only)

You can now specify separate values for both the DB2 plan and the static-command-program. This enables the implementation of DB2 packages for your CA-Easytrieve/Plus SQL programs.

PLANOPTS Parameter on PARM Statement (CA-Datacom SQL only)

You can provide the SQL Interface with an options module to specify your CA-Datacom SQL options for the new plan.

SELECT INTO Support (DB2 Only)

The SQL SELECT INTO statement is now supported if a PARM SQLSYNTAX value of NONE is specified with PARM BIND STATIC-ONLY. This enables you to code Singleton Select statements and avoid coding DECLARE CURSOR statements.
DB2 - DL/I Batch Support

DB2 - DL/I synchronization is achieved through the use of DB2-DLI Batch Support rather than the DB2 Call Attach. You specify which mode of execution with the new DLI SQL parameter of the Execution Options Table.

IDD Updating

CA-Easytrieve/Plus now maintains the relationships between CA-Easytrieve/Plus programs and dictionary entities by updating the IDD at compile time. Unless the subschema requires it, registration can be bypassed by coding the RETRIEVAL parameter on the IDD statement.

IDMS RETURN KEY Support

The IDMS RETURN statement now supports retrieving a record's symbolic key.

SQL with CA-IDMS

CA-Easytrieve/Plus programs can access, report, and update CA-IDMS information using SQL statements with the CA-IDMS version 12.0 SQL option.

CA-Datacom SQL Support

CA-Easytrieve/Plus now has direct access to CA-Datacom/DB using SQL statements if you have version 8.0 or higher of CA-Datacom/DB with the SQL option.

Support of IBM's SMS BLKSIZE=0 Feature for PRINTER Files

Specifying BLOCK0=P in the Options Table, permits the operating system to determine the optimum blocksize for PRINTER files when logical record length and blocksize are not included on the FILE statement.

Additional Support of IBM's SMS BLKSIZE=0 Feature for Disk and Tape Files

Specifying BLOCK0=D in the Options Table, permits the operating system to determine the optimum blocksize for disk and tape data sets when logical record length and blocksize are not included on the FILE statement and DSORG is not included on the DD statement in the JCL.
Year 2000 Support

A new system-defined field has been added that contains the century. SYSDATE-LONG is a 10 A field and is the same format as SYSDATE except that the year is prefixed with the century. Like SYSDATE, the format of the data is controlled by the Options Table value DATE= and SEPDATE=.

Two new options have been added to the REPORT statement. They are SHORTDATE and LONGDATE. With SHORTDATE, the date on reports is displayed using a two-digit year. With LONGDATE, the date on reports is displayed using the century and the two-digit year.

A new option has been added to the Options Table to specify the default date that appears on REPORTs. The new option is LONGDTE and the default is LONGDTE=NO.

The compile listing displays SYSDATE or SYSDATE-LONG depending on the LONGDTE value in the Options Table.
Modes of Operation

You can better understand how CA-Easytrieve/Plus works by examining the modes in which it operates, the structure of a program, and the syntax rules of CA-Easytrieve/Plus statements.

CA-Easytrieve/Plus provides five modes of operation that facilitate production as well as ad hoc programming. The modes of operation (determined by the PARM statement parameters you choose) are:

- Syntax check CA-Easytrieve/Plus source statements.
- Syntax check and compile CA-Easytrieve/Plus source statements.
- Syntax check, compile, and execute CA-Easytrieve/Plus source statements (the default).
- Syntax check and compile CA-Easytrieve/Plus source statements and produce an object module.
- Execute previously link-edited programs.

Structure of a Program

Each CA-Easytrieve/Plus program contains an optional environment definition section, an optional library definition section, and one or more activity sections.
**Environment Definition Section**

The environment definition section establishes parameters for the program. This section permits you to override standard CA-Easytrieve/Plus options and to choose one of the modes of operation.

**Library Definition Section**

The library definition section describes the data to be processed by the program. It describes data files and their associated fields, as well as any working storage requirements of the program.

**Activity Definition Section**

The activity definition section is the only mandatory section of your program. There are two types of activities -- JOB and SORT. You can code any number of JOB or SORT activities in any order.

- JOB activities read information from files, examine and manipulate data, write information to files, and initiate printed reports.
- SORT activities create sequenced files.

You can code one or more procedures (PROCs) at the end of each activity.

You can code one or more REPORT subactivities after the PROCs at the end of each JOB activity. You must code PROCs used within a REPORT subactivity immediately after the REPORT subactivity in which you use them.

The general order of CA-Easytrieve/Plus statements within a program is:

```
<table>
<thead>
<tr>
<th>Environment</th>
<th>PARM ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library</td>
<td>FILE ...</td>
</tr>
<tr>
<td></td>
<td>DEFINE ...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Activities</td>
<td>JOB</td>
</tr>
<tr>
<td></td>
<td>(statements)</td>
</tr>
<tr>
<td></td>
<td>(job procedures)</td>
</tr>
<tr>
<td></td>
<td>REPORT</td>
</tr>
<tr>
<td></td>
<td>(report procedures)</td>
</tr>
<tr>
<td></td>
<td>SORT</td>
</tr>
<tr>
<td></td>
<td>(sort procedures)</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
```
The following code illustrates an example program showing the order of CA-Easytrieve/Plus statements.

```
ENVIRONMENT  PARM ABEXIT SNAP DEBUG(PMAP DMAP FLOW)
  *
FILE PERSNL FB(150 1800)
  *
LIBRARY      NAME            17   20 A
   NAME-LAST NAME     8 A
   NAME-FIRST NAME +8 12 A
   DATE-OF-BIRTH 103   6 N
   TELEPHONE       117   10 N
   SEX             127   1 N
   MALES           W   2 N
   FEMALES         W   2 N
   *
FILE SORTWRK FB(150 1800)  VIRTUAL
COPY PERSNL
  *
ACTIVITIES   SORT PERSNL TO SORTWRK USING +
   (NAME-LAST NAME-FIRST) NAME MYSORT +
   BEFORE SCREENER
  *
SCREENER. PROC
   IF SEX = 1, 2
   SELECT
   END-IF
END-PROC
  *
JOB INPUT SORTWRK NAME MYPROG +
FINISH FINISH-PROCEDURE
   IF SEX = 1
   PRINT REPORT1
   PERFORM JOB-PROCEDURE
   END-IF
   IF SEX = 2
   PRINT REPORT2
   PERFORM JOB-PROCEDURE
   END-IF
  *
JOB-PROCEDURE. PROC
   IF SEX = 1
   FEMALES = FEMALES + 1
   END-IF
   IF SEX = 2
   MALES = MALES + 1
   END-IF
END-PROC
  *
FINISH-PROCEDURE. PROC
   DISPLAY 'NUMBER OF FEMALES = ' FEMALES
   DISPLAY 'NUMBER OF MALES   = ' MALES
END-PROC
  *
```
REPORT REPORT1
LINE NAME SEX DATE-OF-BIRTH TELEPHONE
*
ENDPAGE. PROC
   DISPLAY '*** END OF REPORT1 ***'
END-PROC
*
REPORT REPORT2
LINE NAME SEX DATE-OF-BIRTH TELEPHONE
*
ENDPAGE. PROC
   DISPLAY '*** END OF REPORT2 ***'
END-PROC
Program Flow

The following exhibit illustrates the general flow of a typical program.

You can code multiple JOBs, SORTs, and REPORTs within one program.
The following diagram illustrates the flow of an activity within a typical program.

```
trans

JOB
IF
PUT

edit
report

edited
trans

SORT

sorted
trans

old
master

new
master

JOB
PRINT
PRINT
PRINT
PRINT
PUT
```
This program performs editing functions, creates a new file, and then passes that file to the next activity. It also uses the transactions to update the master file.

**Syntax Rules**

The free-form English language structure of CA-Easytrieve/Plus makes it easy for you to develop an efficient, flexible programming style. To avoid programming errors, follow the simple syntax rules of CA-Easytrieve/Plus.

**Statement Area**

All source statements are records of 80 characters each. The system installation option SCANCOL (explained in Appendix C, “Options Table”) establishes a statement area within the 80 available positions. The default statement area is in columns 1 through 72.

For example, although positions 1 to 80 are available, ‘SCANCOL=(7,72)’ establishes the statement area as positions 7 to 72. This enables optional data (for example, sequence numbers and program identifiers) to be entered on the record, but still be ignored by CA-Easytrieve/Plus. The complete record is always printed on the statement listing.

```
1....6 7......................................2 3......0
001000                                          PRGMNAME
    ignored          statement area                  ignored
```
Character Sets

Both EBCDIC and DBCS data processing are applicable to China (Hanzi characters), Japan (Kanji, Hiragana, and Katakana characters), and Korea (Haja and Hangual characters). CA-Easytrieve/Plus supports both character sets based on the following assumptions and rules:

1. All the syntax rules described in this chapter apply to EBCDIC data only. DBCS data in the CA-Easytrieve/Plus statement area is not processed for continuation characters, delimiters, words, identifiers, and so on.

2. A DBCS character occupies two bytes in storage. If not identified as DBCS characters, these same two bytes would be processed as a pair of single byte EBCDIC characters. To distinguish EBCDIC data from DBCS data, CA-Easytrieve/Plus uses two shift code systems. The first system, called the Wrapping shift code system, takes the form of two codes -- one code preceding and the second following the DBCS data. These codes wrap or enclose the DBCS data, thereby identifying the beginning and end of DBCS data. The term associated with the code that precedes the DBCS data is a Shift-Out code (shift-out of EBCDIC). The code that delimits (separates) the DBCS data is called a Shift-In code (shift-in to EBCDIC). These codes can be one or two bytes in length.

The second system of shift codes, called Header shift codes, uses a one- or two-byte code to identify that the following byte is a binary number whose value defines the amount of non-EBCDIC data that follows. Immediately beyond that point, the processing of EBCDIC data can resume.
The following table illustrates the use of both shift code systems.

<table>
<thead>
<tr>
<th>1. WRAPPING SHIFT CODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE: 'EEEE[DDBBDBDB]EEEE'</td>
</tr>
<tr>
<td>CA-Easytrieve Plus delimiter.</td>
</tr>
<tr>
<td>EBCDIC data as part of literal.</td>
</tr>
<tr>
<td>Shift-in code.</td>
</tr>
<tr>
<td>Double byte data as part of literal.</td>
</tr>
<tr>
<td>Shift-out code.</td>
</tr>
<tr>
<td>EBCDIC data as part of Alphabetic literal.</td>
</tr>
<tr>
<td>CA-Easytrieve Plus delimiter. Must be EBCDIC.</td>
</tr>
<tr>
<td>CA-Easytrieve Plus keyword. Must be EBCDIC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2. HEADER SHIFT CODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE: 'EEEE?DBDBDBDBDBBBBEEEE'</td>
</tr>
<tr>
<td>CA-Easytrieve Plus delimiter.</td>
</tr>
<tr>
<td>EBCDIC data as part of literal.</td>
</tr>
<tr>
<td>Double byte data as part of literal.</td>
</tr>
<tr>
<td>One byte binary number. Value is 8 to indicate 8 bytes of DBCS data.</td>
</tr>
<tr>
<td>Header shift code.</td>
</tr>
<tr>
<td>EBCDIC data as part of Alphabetic literal.</td>
</tr>
<tr>
<td>CA-Easytrieve Plus delimiter. Must be EBCDIC.</td>
</tr>
<tr>
<td>CA-Easytrieve Plus keyword. Must be EBCDIC.</td>
</tr>
</tbody>
</table>

- A shift code is a special one-or two-byte character contained in the CA-Easytrieve/Plus statement area. Shift code values are defined in the DBCS Options module (see the *CA-Easytrieve/Plus Installation Guide* for more details on the Options module). Each shift code value uniquely identifies the DBCS code system of the data. If the system cannot be uniquely identified, a default is assumed. You can alter this default at compile time with the PARM statement (for more information, see the PARM statement description in Chapter 4).
In the statement area, CA-Easytrieve/Plus requires shift codes to distinguish DBCS data from EBCDIC data. Once a CA-Easytrieve/Plus word has been identified, the word is known to be of EBCDIC, DBCS, or MIXED data format. CA-Easytrieve/Plus only maintains shift codes for MIXED words. The CA-Easytrieve/Plus compiler identifies the statement containing the word and when necessary, performs the required processing to remove the shift codes and convert EBCDIC data.

A word can only contain DBCS data belonging to one code system. Mixing code systems within one word generates an error.

Only one shift code system can be defined for a DBCS code system.

Once CA-Easytrieve/Plus finds a shift-out code in a word, the data that follows will not be processed as DBCS data when:
- The end statement area is reached before the related shift-in code is found.
- The shift-in code is found but it is not on a double byte boundary.
- The shift-out code is found as part of the identified DBCS data.

Once CA-Easytrieve/Plus finds a Header shift code in a word, the data that follows is not processed as DBCS data when:
- The end statement area is reached before the binary data count is expired.
- The value of the data byte count is not a multiple of two.

Multiple Statements

The statement area normally contains a single statement. However, you can enter multiple statements on a single record. The EBCDIC character string '. ' (period followed by a space) indicates the end of a statement. Another CA-Easytrieve/Plus statement begins at the next available position of the statement area (after the space). For example, the following two CA-Easytrieve/Plus statements are on one record:

```
COST = FIXED + VARIABLE.  PRICE = COST + PROFIT
```

Comments

When the first non-blank character of a statement is an '*' (asterisk), the remainder of that record is a comment statement. You can use comment statements at any place within a program, except within a continued statement and not prior to or within instream macros. A statement containing all blanks is treated as a comment.
A comment statement can contain both EBCDIC and DBCS data in the remainder of the record. Any DBCS data must be identified by shift code(s). The shift code(s) must uniquely identify the code system of the DBCS data contained on the comment statement.

### Continuations

The last non-blank character of a statement terminates the statement unless that character is a - (hyphen) or a + (plus). The - indicates that the statement continues at the start of the next statement area. The + indicates that the statement continues with the first non-blank character in the next statement area. The difference between - and + is important only when continuing words. Continuation between words is the same for both. The following continued statements produce identical results:

```
FIELD-NAME W 6 A +
VALUE 'ABC-
DEF'
```

```
FIELD-NAME W 6 A -
VALUE 'ABC+
DEF'
```

To continue a statement defining DBCS data, you must delimit the DBCS data. This means a shift-in code must precede the continuation character and a shift-out code must precede the continuing DBCS data on the next record. The following illustrates continuing a DBCS literal.

```
FIELD-NAME W 10 K +
[value ' [DBDBDB]'

The [ and ] indicate shift-out and shift-in codes.
```

```
FIELD-NAME W 10 K +
VALUE '?6DB8DBDB+
?4DB8B'
```

The ? indicates a Header shift code followed by byte count.
Words and Delimiters

One or more words make up each CA-Easytrieve/Plus statement. A word can be a keyword, field name, literal, or symbol. All words begin with a non-blank character. A delimiter or the end of the statement area terminates these words. Delimiters make statements readable but are not considered part of the attached word. CA-Easytrieve/Plus word delimiters are:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>space</td>
<td>Basic delimiter in each statement.</td>
</tr>
<tr>
<td>'</td>
<td>apostrophe</td>
<td>Encloses literals which are alphabetic.</td>
</tr>
<tr>
<td>.</td>
<td>period</td>
<td>Terminates a statement.</td>
</tr>
<tr>
<td>,</td>
<td>comma</td>
<td>Used optionally for readability.</td>
</tr>
<tr>
<td>()</td>
<td>parentheses</td>
<td>Encloses multiple parameters and portions of arithmetic expressions (the left parenthesis acts as a basic delimiter).</td>
</tr>
<tr>
<td>:</td>
<td>colon</td>
<td>Used as a delimiter for file, record, and field qualifications.</td>
</tr>
</tbody>
</table>

At least one space must follow all delimiters except for the '(' (left parenthesis) and ':' (colon). The word RECORD-COUNT is shown below with various delimiters:

RECORD-COUNT
FILEONE:RECORD-COUNT
(RECORD-COUNT)
'RECORD-COUNT'
RECORD-COUNT,
RECORD-COUNT.

Keywords

Keywords are words which have specific meaning to CA-Easytrieve/Plus. Some keywords are reserved words. You can use non-reserved keywords in the appropriate context as field names. For more information on keywords and reserved words, refer to Appendix B, “Keywords.”

Multiple Parameters

You must enclose multiple parameters within parentheses to indicate group relationships. The following example is a CA-Easytrieve/Plus statement with multiple parameters:

CALL PGMNAME USING(FIELDA, FIELDB, FIELDCC)
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, and all field names within a single file, must be unique, and all field names within a single file. If you use the same field name in more than one file, or in a file and in working storage, 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.

The field name RECORD-COUNT can be qualified in the following ways:

- FILEA: RECORD-COUNT
- FILEA:RECORD-COUNT
- WORK : RECORD-COUNT

Labels

Labels identify specific JOBs, PROCedures, REPORTs, and statements. Labels can be 40 characters long, can contain any character other than a delimiter, and can begin with A-Z or 0-9; they cannot consist of all numeric characters.

EBCDIC Alphabetic Literals

Alphabetic literals are words enclosed in apostrophes, and can be 254 characters long. They can only contain EBCDIC characters. Whenever an alphabetic literal contains an embedded apostrophe, you must code two apostrophes. For example, the literal O’KELLY is coded as:

'O’KELLY'
Numeric Literals

Numeric literals can contain 18 numeric digits (EBCDIC characters 0 to 9). You can indicate the algebraic sign of a numeric literal by attaching a + (plus) or a - (minus) prefix to the numeral. Also, you can use a single decimal point to indicate a maximum precision up to 18 decimal positions. The following examples are valid numeric literals:

123
+123
-123.4321

EBCDIC Format Hexadecimal Literals

Hexadecimal literals are words used to code EBCDIC values which contain characters not available on standard data entry keyboards. Prefix an EBCDIC hexadecimal literal with X' (the letter X and an apostrophe), and terminate it with an apostrophe. CA-Easytrieve/Plus compresses each pair of digits that you code within the apostrophes into one character. CA-Easytrieve/Plus enables only the EBCDIC digits 0 to 9 and the letters A to F. The following hexadecimal literal defines two bytes of binary zeros:

X'0000'

DBCS Format Literals

DBCS format literals contain DBCS characters only. Enclose a DBCS format literal within apostrophes. A DBCS format literal can be 254 bytes long, including the shift codes. Once the shift codes are removed, the maximum length of a DBCS literal is 126 characters. CA-Easytrieve/Plus processes DBCS format literals without the shift codes. Examples of DBCS literals follow:

' [DBDBDBDBDBDB]'  
' ?0DBDBDBDBB'

The [ and ] indicate shift-out and shift-in codes.

The ? indicates a Header shift code followed by byte count.
DBCS Format Hexadecimal Literals

DBCS format hexadecimal literals are words that you use to code DBCS values that contain characters not available on standard data entry keyboards. Prefix a DBCS hexadecimal literal with D' (the letter D and an apostrophe) and terminate it with an apostrophe. CA-Easytrieve/Plus compresses each group of four digits coded within the apostrophes into one DBCS format character. The code system of the literal is the current DBCS processing code system. For DBCS format hexadecimal literals, CA-Easytrieve/Plus allows only the EBCDIC digits 0 to 9 and the EBCDIC letters A to F. You cannot code shift codes as part of a DBCS format hexadecimal literal. The following hexadecimal literal defines two Double Byte spaces in the IBM DBCS coding standards:

D'40404040'

MIXED Format Literals

MIXED format literals are words containing both EBCDIC and DBCS characters. Enclose MIXED format literals within apostrophes. The presence of shift codes identifies DBCS subfields. Shift codes also identify the code system of that DBCS data. The word coded within the apostrophes (including the shift codes) cannot exceed 254 bytes in length. A MIXED literal is defined in the following example:

'EEE[E [DBDB]]'

The [ and ] indicate shift-out and shift-in codes.

Alphabetic Conversion Literals

Alphabetic conversion literals are words containing both EBCDIC and DBCS characters. Enclose alphabetic conversion literals within apostrophes. The presence of shift codes identifies DBCS subfields. Shift codes also identify the code system of that DBCS data. CA-Easytrieve/Plus converts the literal into a DBCS literal. CA-Easytrieve/Plus converts any EBCDIC format data into the corresponding DBCS format data and removes all shift codes.

Prefix an alphabetic conversion literal with E' (the letter E and an apostrophe) and terminate it with an apostrophe. The word coded within the apostrophes (including the shift codes) and the word built by the conversion process cannot exceed 254 bytes (127 DBCS characters).
The following example illustrates the definition of an alphabetic conversion literal and the resultant DBCS literal:

```
K' e e E E E
```

Result → `DBDBDBDBDB`

```
K' e e E E E [ D B D E D B ] '
```

Result → `DBDBDBDB` `DBDBDB`

### Katakana Conversion Literals

Katakana conversion literals are words coded using both EBCDIC and DBCS format characters. The presence of shift codes identifies DBCS subfields. Shift codes also identify the code system of that DBCS data. CA-Easytrieve/Plus interprets the EBCDIC subfields as the EBCDIC representation of Katakana characters. Katakana characters are represented in the EBCDIC system by those binary values normally occupied by the English lower case character set. CA-Easytrieve/Plus converts each Kana character coded in the conversion literal into the corresponding DBCS format character. If diacritical marks are present, CA-Easytrieve/Plus merges each diacritical mark with the preceding EBCDIC character to give the DBCS format character that the character and the mark form. CA-Easytrieve/Plus converts any other EBCDIC characters into their appropriate DBCS format value.

Prefix a Katakana conversion literal with K' (the letter K and an apostrophe) and terminate it with an apostrophe. The word coded within the apostrophes (including the shift codes) and the word built by the conversion process cannot exceed 254 bytes (127 DBCS characters).

The following example illustrates the definition of an alphabetic conversion literal and the resultant DBCS literal:

```
K' e e E E E
```

Result → `DBDBDBDBDB`
Please note that the conversion technique described here for Katakana conversion literals is used by CA-Easytrieve/Plus when CA-Easytrieve/Plus converts EBCDIC format data in EBCDIC and MIXED format literals and fields into DBCS format data.

Hiragana Conversion Literals

Hiragana conversion literals are words coded using both EBCDIC and DBCS format characters. The presence of shift codes identifies DBCS subfields. The shift codes also identify the code system of that DBCS data. CA-Easytrieve/Plus interprets EBCDIC subfields as the EBCDIC representation of Katakana characters. In the Japanese language, for each Katakana symbol there is an equivalent Hiragana symbol. By indicating a word as a Hiragana conversion literal, CA-Easytrieve/Plus interprets each Katakana EBCDIC character as its equivalent Hiragana character and then converts the character into the corresponding DBCS format character. CA-Easytrieve/Plus merges any diacritical marks found in the literal with the preceding Katakana character. CA-Easytrieve/Plus inserts into the literal the applicable DBCS value for the merged characters. CA-Easytrieve/Plus converts any other EBCDIC characters into their appropriate DBCS format value.

Prefix a Hiragana conversion literal with H' (the letter H and an apostrophe) and terminate it with an apostrophe. The word coded within the apostrophes (including the shift codes) and the word built by the conversion process cannot exceed 254 bytes (127 DBCS characters).

The following example illustrates the definition of an alphabetic conversion literal and the resultant DBCS literal:

```
FIELD-NAME W 10 K VALUE H 'éééÉ'
```

Result → DBDEBDDBDBDB

User Converted Literals

User-converted literals are words specified in the CA-Easytrieve/Plus source code as either EBCDIC, MIXED, or DBCS literals that must be converted by a user-supplied routine into an alphabetic EBCDIC, an alphabetic MIXED, or an alphabetic DBCS literal. For a detailed description of coding a user conversion subprogram see the CA-Easytrieve/Plus Installation Guide.
Prefix a user-converted literal with any alphabetic character (other than the letters D, E, H, K, and X) and an apostrophe (for example, Q’). Terminate a user converted literal with an apostrophe. You can obtain the alphabetic characters that your installation supports, and their meaning in a CA-Easytrieve/Plus program, from your CA-Easytrieve/Plus system administrator.

The word coded within the apostrophes cannot exceed 254 EBCDIC characters. CA-Easytrieve/Plus does not convert any portion of the literal -- it passes it unaltered to the user supplied routine. The presence of shift codes identifies DBCS subfields. Shift codes also identify the code system of that DBCS data. If the literal is an alphabetic DBCS literal, then CA-Easytrieve/Plus removes the shift code characters and passes only the DBCS data to the user routine. If the literal is an alphabetic MIXED format literal, then the shift codes are maintained in the literal.

Format and Conversion Rules

During compile, CA-Easytrieve/Plus converts all literals coded in the source program into the correct DBCS code system and data format (EBCDIC, MIXED, or DBCS) as dictated by the CA-Easytrieve/Plus statement upon which they appear. To understand the process used to determine the correct code system and data format, it is important to identify the element of each CA-Easytrieve/Plus statement that is interpreted as the subject of that statement. It is this subject that dictates the correct code system and format type.

Each CA-Easytrieve/Plus statement has a subject element whose DBCS code system and Data Format define the DBCS code system and data format of all the other elements that appear on that statement. The following table details the subject element of those CA-Easytrieve/Plus statements that support the coding of literals. Those that do not have a subject element are also included. They are indicated by the words “Not Applicable.”

<table>
<thead>
<tr>
<th>Statement</th>
<th>Subject Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE file-name ....</td>
<td>file-name</td>
</tr>
<tr>
<td>DEFINE field-name ....</td>
<td>field-name</td>
</tr>
<tr>
<td>Assignment - field-name = ....</td>
<td>field-name</td>
</tr>
<tr>
<td>IF field-name ....</td>
<td>field-name</td>
</tr>
<tr>
<td>DO WHILE field-name ....</td>
<td>field-name</td>
</tr>
<tr>
<td>RETRIEVE WHILE field-name ....</td>
<td>field-name</td>
</tr>
<tr>
<td>MOVE ....</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>POINT file-name ....</td>
<td>file-name</td>
</tr>
<tr>
<td>CALL program-name ....</td>
<td>Not Applicable</td>
</tr>
</tbody>
</table>
Using the identified subject element of each CA-Easytrieve/Plus statement, the following table defines the rules for determining the DBCS Code System and Data Format for a literal. If a literal is not in the required DBCS code system or data format, then at compile time CA-Easytrieve/Plus converts the literal to the correct DBCS code system and data format.

In the following table, the code ASIS means that the data format (EBCDIC, MIXED, or DBCS) of the literal coded in the CA-Easytrieve/Plus source is retained by the CA-Easytrieve/Plus compiler (it is not converted).

<table>
<thead>
<tr>
<th>Statement-Keyword</th>
<th>Data Format of Literal</th>
<th>DBCS Code System of Literal</th>
</tr>
</thead>
<tbody>
<tr>
<td>FILE - EXIT USING</td>
<td>ASIS</td>
<td>PROCESSING</td>
</tr>
<tr>
<td>DEFINE - HEADING VALUE</td>
<td>ASIS</td>
<td>PROCESSING</td>
</tr>
<tr>
<td>IF/DO....WHILE Assignment</td>
<td>field-name</td>
<td>field-name</td>
</tr>
<tr>
<td>MOVE</td>
<td>ASIS</td>
<td>PROCESSING</td>
</tr>
<tr>
<td>POINT</td>
<td>ASIS</td>
<td>file-name</td>
</tr>
<tr>
<td>HEADING</td>
<td>ASIS</td>
<td>file-name(printer)</td>
</tr>
<tr>
<td>TITLE</td>
<td>ASIS</td>
<td>file-name(printer)</td>
</tr>
<tr>
<td>LINE</td>
<td>ASIS</td>
<td>file-name(printer)</td>
</tr>
</tbody>
</table>
Format Relationship Rules

CA-Easytrieve/Plus processes EBCDIC, MIXED, and DBCS data formats, but does not support all the possible relationships that can exist between these data formats. The following table defines the relationships CA-Easytrieve/Plus supports. CA-Easytrieve/Plus does not support the relationships not defined in the table. Compilation errors occur if you specify them in your CA-Easytrieve/Plus program.

If a conversion is necessary, the conversion column in the table indicates the additional processing that can apply to get the Object into the correct DBCS code system and applicable data format. Conversion codes of A, K, H, and U indicate the form of Conversion to be applied to any EBCDIC data found in the Object of the CA-Easytrieve/Plus statement. These codes mean A for Alphabetic conversion; K Katakana conversion; H for Hiragana conversion; and U for a User conversion routine. The letter F means that CA-Easytrieve/Plus re-formats the object to meet the requirements of the subject element. This category includes the re-formatting of Numeric data into the numeric format of the subject and also the re-formatting from one data format (EBCDIC, MIXED, or DBCS) into the data format of the subject element. The letter C defines that CA-Easytrieve/Plus performs a DBCS code conversion when the DBCS code system of the data identified in the object does not match the DBCS code system of the subject element.

<table>
<thead>
<tr>
<th>Subject Element data format</th>
<th>Supported Object data format</th>
<th>Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - EBCDIC Alpha</td>
<td>EBCDIC Alphabetic field</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EBCDIC Zoned Numeric field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Unsigned Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Binary field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Alphabetic Literal</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EBCDIC Hexadecimal Literal</td>
<td></td>
</tr>
<tr>
<td>N - Zoned Numeric</td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Unsigned Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Binary field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Numeric Literal</td>
<td></td>
</tr>
<tr>
<td>P - Packed</td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Unsign Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Binary field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Numeric Literal</td>
<td>F</td>
</tr>
<tr>
<td>U - Unsigned Packed</td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Unsign Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Binary field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Numeric Literal</td>
<td>F</td>
</tr>
<tr>
<td>B - Binary</td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Unsign Packed field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Binary field</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>EBCDIC Numeric Literal</td>
<td>F</td>
</tr>
</tbody>
</table>
Identifiers

Identifiers are words which name things (field name, statement labels, and so forth) in CA-Easytrieve/Plus. Identifiers cannot contain these delimiters.

. comma
' apostrophe
( left parenthesis
) right parenthesis
: colon

Arithmetic Operators

CA-Easytrieve/Plus arithmetic expressions (see Chapter 7, “Assignments and Moves”) use the following arithmetic operators:

* multiplication
/ division
+ addition
- subtraction

The arithmetic operator must lie between two spaces.
This chapter discusses a method for customizing the operating environment for the duration of a program's compilation and execution. The environment or general systems standards for a program are determined by the parameters in the system options table and the DBCS options module. The PARM statement and its parameters override selected general standards for a program. Its alteration lasts for only as long as the program is running.

The PARM statement and its parameters override selected general standards for a program. Its alteration of the environment lasts for only as long as the program is running.

Specification of the PARM statement is optional. Code the PARM statement only to modify the environment for your program. If used, the PARM statement must be the first statement in your CA-Easytrieve/Plus job, except when instream macros are used, then the PARM immediately follows the last MEND statement.

PARM establishes program level parameters in the following areas:

- SYNTAX, COMPILE, and LINK determine the mode of CA-Easytrieve/Plus execution.
- ABEXIT, DEBUG, and LIST establish control over system facilities associated with compiler output and execution error handling.
- DEVICE establishes the specific device type to be used during file processing.
- PRESIZE, EXITSTR, and VFM establish system control parameters.
- SORT controls the interface to your installation's sort program.
- VFM establishes system control parameters.
- BIND, PLAN, PREPNAME, SQLID, SSID, PLANOPTS, SQLSYNTAX, and USERID establish parameters for SQL execution.

**PARM Statement Syntax**

Code PARM statement parameters and their subparameters in any order. As shown in the following syntax, you must code multiple subparameters within parentheses. The complete syntax of the PARM statement is:

```plaintext
PARM +
   [ { SNAP } ]
   [ ABEXIT {NOSNAP} ] +
   [ { NO } ]
   [ {DYNAMIC } ]
   [ BIND {STATIC-ONLY} ] +
   [ {ANY } ]
   [COMPILE] +
   [ { IBM } ] [ { IBMKOREA} ]
   [ { JEF } ] [ { JEF4040 } ]
   [ { KEIS } ] [ { KEIS4040 } ]
   [ { JEF4040 } ] [ { JIPSE } ] [ { JIPSE } ]
   [ { JIS } ] [ { JIS } ]
   [ { KEIS } ] [ { KEIS } ]
   [ { KEIS4040 } ] [ { MELCOM } ]
   [ { SHOWA } ] [ { SHOWA } ]
   [ { TORA Y} ] [ { TORA Y} ]
   [DEBUG( { CLIST } { PMAP } { DMAP } { FLDCHK } {FLOW } {FLOWSIZ literal-1} ] +
   [ {NOCLIST} {NOPMAP} {NODMAP} {NOFLDCHK} {NOFLOW} ]
   [ { STATE } { XREF {LONG } ]
   [ { NOSTATE } { NOXREF {SHORT} } ] +
   [ {DISK} ]
   [ {FBA } ]
   [ {TAPE } ]
   [ {3330 } ]
   [DEVICE  {3340} ] +
   [ {3350 } ]
   [ {3375 } ]
   [ {3390 } ]
```

PARM Statement Syntax
[TP ]  +
[NOTP]
[

[VIRTUAL]  +
[REAL ]
[

[WORK { DA ]] ) ] +
{ literal-9 ]

[SQILD 'owner'] +
[
[SQLSYNTAX {PARTIAL}] ]
{NONE ]

[SSID('ssid')] +

[SYNTAX]  +

[USERID ('user-id2' ['password'])] +
[
{ DISK ]
{ FBA ]
{ MEMORY ]
{ 3330 ]

[VFM {[literal-10] [DEVICE 3340 ]]] ]
{ 3350 ]
{ 3375 ]
{ 3380 ]
{ 3390 ]
}
PARM Statement Parameters

[ { ABEXIT { SNAP } { NOSNAP } } { NO } ]

ABEXIT indicates the level of control exercised over program interrupt codes 1 through 11. SNAP prints a formatted dump of CA-Easytrieve/Plus storage areas along with an error analysis report. NOSNAP prints only an error analysis report. NO inhibits CA-Easytrieve/Plus interception of program interrupts.

[ { BIND { DYNAMIC } { STATIC-ONLY } } { ANY } ]

BIND is an SQL-related parameter that identifies the type of SQL bind that you want for the execution of your application program. BIND is currently only used by the DB2 SQL interface. It is ignored in other environments.

BIND DYNAMIC results in the dynamic execution of the SQL statements in your program. Dynamic processing requires SQL statements to be dynamically “prepared” before they can be executed. The SQL interface controls the SQL environment and does not prepare SQL statements again unless a syncpoint has been taken.

BIND STATIC-ONLY indicates that your application program is to execute statically. This option requires the creation of a “static-command-program” that is then processed by the DB2 preprocessor. The DB2 preprocessor generates a DBRM and finally a PLAN. During the execution of your application program, the SQL interface processes the SQL statements in the “static-command-program.” If any errors are found in the “static-command-program” or its PLAN, SQL processing terminates.

BIND ANY indicates that a “static-command-program” is to be generated and a PLAN created, as with an option of STATIC-ONLY. However, if the SQL interface encounters any errors with the “static-command-program” or its PLAN during the execution of your application program, it switches to dynamic processing.

BIND STATIC-ONLY or BIND ANY requires a value for the PLAN and LINK parameters. PLAN specifies the name of the “static-command-program” and its DB2 PLAN name. LINK identifies the load module name of your link-edited CA-Easytrieve program. Your CA-Easytrieve application program must run as a link-edited program for static SQL processing.
DYNAMIC is the default mode of execution if no value is specified for the BIND parameter in the program or in the Options Table. Otherwise, the BIND value in the Options Table becomes the default. The following table illustrates the use of the BIND parameter with values specified in the Options Table.

<table>
<thead>
<tr>
<th>BIND Parameter Value Specified</th>
<th>Value Specified in the Options Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>b</td>
</tr>
<tr>
<td>No BIND parameter specified</td>
<td>BIND defaults to DYNAMIC</td>
</tr>
<tr>
<td>ANY</td>
<td>ANY is the BIND parameter</td>
</tr>
<tr>
<td>STATIC-ONLY</td>
<td>STATIC-ONLY is the BIND parameter</td>
</tr>
<tr>
<td>DYNAMIC</td>
<td>DYNAMIC is the BIND parameter</td>
</tr>
</tbody>
</table>

[COMPILE]

COMPILE terminates execution after the completion of the syntax check and compile operations.

[DBCSCODE]

DBCSCODE and its subparameters define the DBCS code system that CA-Easytrieve/Plus is to use as the Source code system and the Processing code system. This option is invalid if the DBCS option is not supported.

[SOURCE]

This option alters the DBCS code system identified by a set of shift codes when the shift code values are not unique to one DBCS code system. For example, IBM systems use a shift-out code of X'0E', but in some situations, the X'0E' code may be used to identify JEF DBCS code. When CA-Easytrieve/Plus is processing words coded in CA-Easytrieve/Plus source, the DBCS options module defines the actual code system to be associated with the X'0E' shift-out code (for example, IBM). Therefore, if the source contained JEF code identified by the X'0E' shift-out code, then this subparameter of the PARM statement can be used to alter the system default.
Valid DBCS code systems that can be specified with the SOURCE keyword are those code systems in the following list that have been defined in the DBCS options module. For more information on the DBCS options module, refer to the CA-Easytrieve/Plus Installation Guide.

- IBM DBCS Code System for Japan
- IBMKOREA DBCS Code System for Korea
- JEF (FACOM) Code
- JEF4040 (FACOM) Code
- JIPSE (NEC) Code
- JIS (Japanese Industrial Standards)
- KEIS (HITACHI) Code
- KEIS4040 (HITACHI) Code
- MELCOM Code
- SHOWA Information System (SIS) Code
- TORAY Code.

This option alters the code system that CA-Easytrieve/Plus uses for both DBCS and MIXED fields in Working Storage and in those files where the DBCSCODE parameter was not specified on the FILE statement. If you do not use this option, then CA-Easytrieve/Plus uses the default code system defined in the DBCS options module. For information on the DBCS Options module, refer to the CA-Easytrieve/Plus Installation Guide.

You should pay special attention to the use of this option when using multiple DBCS code systems in your program. An unwise choice could result in a much longer execution time than is necessary. By evaluating the best Processing code system for the program, you can avoid unnecessary DBCS code conversions.
Valid DBCS code systems that you can specify with the PROCESS keyword are those code systems in the following list that have been defined in the DBCS options module. For more information on the DBCS options module, refer to the CA-Easytrieve/Plus Installation Guide:

- IBM DBCS Code System for Japan
- IBMKOREA DBCS Code System for Korea
- JEF (FACOM) Code
- JEF4040 (FACOM) Code
- JIPSE (NEC) Code
- JIS (Japanese Industrial Standards)
- KEIS (HITACHI) Code
- KEIS4040 (HITACHI) Code
- MELCOM Code
- SHOWA Information System (SIS) Code
- TORAY Code.

[DEBUG]

DEBUG and its subparameters control generation of certain system outputs. These outputs are used to analyze programming errors that cause abnormal execution termination. As shown in the syntax, subparameters prefixed with NO inhibit the named operation.

[CLIST]

CLIST creates a condensed listing of the executable program produced by the compiler.

[PMAP]

PMAP creates a complete listing of the executable program produced by the compiler.

CLIST and PMAP are mutually exclusive subparameters.

[DMAP]

DMAP creates a listing of the data map for each file and its associated fields.

[FLDCHK]

FLDCHK validates all data references during program execution. A data reference is invalid if a field-name was referenced in a file which had no active record. Invalid references (for example, data reference after end-of-file) might otherwise cause a program interruption or incorrect program results.
FLOW activates a trace of the statements being executed. The statement numbers are printed in the associated analysis report.

FLOWSIZ literal-1

FLOWSIZ establishes the number of entries in the trace table for the flow option. Literal-1 is a numeric value from 1 to 4096.

STATE saves the statement number of the statement currently being executed. The statement number is then printed in the associated abnormal termination messages.

XREF causes the creation of a cross reference listing of each field name, file name, procedure name, segment name, report name, and statement label. LONG implies that entries are listed even though they are not referenced. SHORT causes only referenced entries to be listed.

DEVICE (VSE only) specifies the default device type for all files defined in the library section. For programs executing on DOS AF2 or later, the specific disk device type is determined when the file is opened. DISK indicates that the value is to be taken from the DISK parameter of the options table.

ENVIRONMENT (OS/390 only) instructs CA-Easytrieve/Plus to establish the proper execution environment prior to calling any COBOL subprograms. The environment is established prior to each JOB activity that contains a CALL statement and is terminated after the activity for which it was established. When used on the PARM statement, it establishes the default (NONE or COBOL) for the entire CA-Easytrieve/Plus program. When this parameter is absent, the default for ENVIRONMENT depends on how the ENVIRON system option was set at installation. Using NONE overrides a system default of COBOL and COBOL overrides a system default of NONE. See Chapter 14, “Subprograms,” for more information about this parameter.

EXITSTR literal-2

EXITSTR (VSE only) specifies the available storage at execution time for user called programs and for non-VSAM I/O exits. Literal-2 must be a value from 0 to 4096. The literal-2 value represents 1024-byte units of storage.
[LINK (program-name-1 [R])]

LINK terminates execution after the completion of syntax check and compile operations. Program-name-1 is used to create the link edit control statement that names the new program. In OS/390 systems, the optional subparameter R specifies that the new program replaces an existing program with the same name.

[List]

LIST controls the printing of certain system outputs. Subparameters prefixed with NO inhibit the named operation.

(FILE)

FILE prints file statistics at the end of each JOB or SORT activity.

([Parm])

PARM prints system parameters at the conclusion of the syntax check operation.

[Plan (planname [command-program-name])]

PLAN is an SQL parameter. Currently, it is used only by the DB2 SQL interface.

The PLAN parameter enables you to specify values for the "static-command-program" and its DB2 PLAN. The name you specify for the "static-command-program" must be a valid one- to eight-character load module name. This name must be different from program-name-1 specified for the LINK parameter.

The value specified for planname must be the one- to eight-character name of the DB2 PLAN that identifies the DBRM of the given "static-command-program."

Because the link-edit of the "static-command-program" and the bind of the DB2 PLAN are performed outside the control of CA-Easytrieve, you must specify the correct names on the batch JCL to ensure successful execution of your program. See the CA-Easytrieve SQL Interface Option Guide for information on how to generate the "static-command-program."

If not specified, command-program-name defaults to planname.

[Planopts 'plan-options-module']

PLANOPTS is an SQL parameter. Currently, this parameter is only used by the CA-Datacom/DB SQL interface.

Use PLANOPTS to specify the name of the plan options module that is to override the default CA-Pan/SQL plan options module, DQSMPLN®. See the CA-Pan/SQL SQL Interface Installation Guide for more information about generating a plan options module.
[PREPNAME (program-name-2 ['user-id1'])]

PREPNAME is an SQL parameter. Currently, it is used by the SQL/DS and
CA-Datacom/DB SQL interfaces.

For the SQL/DS SQL interface, the PREPNAME parameter enables you to
specify the name of the access module or “package” that is to be associated with
the SQL statements for this application program.

For the CA-Datacom/DB SQL interface, PREPNAME enables you to specify the
access plan.

For either database, the PREPNAME parameter also enables you to specify an
owner ID ('userid-id1') for the access module or access plan. See your specific
database documentation for information about obtaining an authorization ID.

If PREPNAME is not specified, program-name-2 defaults to program-name-1 on
the LINK parameter. If the LINK parameter is not specified, program-name-2
defaults to the value specified in the Options Table.

**Note:** You should specify a unique value for program-name-2 for each
CA-Easytrieve/Plus program. If you use the same name for either parameter
value for all of your programs, database catalog contention can occur, or an
existing access module could be replaced with another one. See your database
administrator for information about establishing naming conventions.

User-id1 must be an alphabetic literal with a maximum length of 30 characters.
The maximum length for program-name-2 is eight characters.

PREPNAME can be abbreviated as PREP.

[PRESIZE literal-3]

PRESIZE establishes the record length for the compiler's work file. Literal-3 must
be a numeric value from 512 to 32767. Code this parameter only to inhibit the
generation of error message B079 (see Appendix A, “Diagnostics”).

[RESTARTABLE...]

(OS/390 only) RESTARTABLE is an IMS-only parameter for use with the
Extended Checkpoint/Restart Facility. For details see the CA-Easytrieve/Plus
IMS/DLI Interface Option Guide.

[SORT]

SORT overrides the default parameters used to interface with your installation's
sort program. Refer to the CA-Easytrieve/Plus Installation Guide for details of these
SORT parameters.
ALTSEQ identifies the collating sequence table for the sort process. NO indicates usage of the standard table. YES identifies an alternate table. Program-name-3 specifies the name of the table that you provide. When you omit program-name-3, the default name is EZTPAQTT.

DEVICE literal-4

DEVICE (OS/390 only) establishes the device type for dynamically allocated sort work data sets. Literal-4 can be any valid unit name or generic device type.

DIAG and NODIAG (VSE only) control printing messages containing diagnostic information.

ERASE and NOERASE (VSE only) control clearing sort work data sets at the end of the sort.

MEMORY specifies the maximum amount of core storage used by the sort program. Literal-5 is the amount of storage made available for the sort and must be a value from 16 to 4096. MAX enables the sort program to obtain maximum storage available. Literal-6 is the amount of storage released (for system use) after the MAX amount has been reserved. A minus sign must immediately precede literal-6. Literal-5 and literal-6 values represent 1024-byte units of storage.

MSG controls the content and routing of sort program messages.

ALL, CRITICAL, DEFAULT, and NO establish the content of sort messages. DEFAULT requests output at the level established when the sort program was installed.
PRINTER and CONSOLE specify the location that receives messages from the sort program.

RELEASE determines the amount of core storage reserved from the sort program. The value of literal-7 should be set large enough to supply all of the core storage needs of any exits used as a part of the sort process. Literal-7 must be a numeric value from 0 to 1024. The value represents 1024-byte units of storage.

SYS (VSE only) specifies the logical unit assignment for sequential disk type sort work data sets. You can code from one to eight values. Literal-8 must be a value from 1 to 240.

TP and NOTP (VSE only) control the merge order of the sort to reduce contention for I/O resources.

VIRTUAL and REAL (VSE only) control fixing storage pages. REAL allows pages to be fixed.

WORK specifies the type and number of work data sets used by the sort.

VSE DA indicates that one direct access type data set is used. Literal-9 specifies the number of sequential disk type data sets used. Valid values for literal-9 are 1 to 8.

OS/390 The value of literal-9 controls the allocation of work data sets. When literal-9 is zero, you must supply DD statements for all work data sets (none are dynamically allocated). A literal-9 value from 1 to 31 specifies the number of work data sets dynamically allocated by the sort program.
SQLID is an SQL parameter. Currently, this parameter is used only by the DB2 SQL interface.

SQLID enables you to change the authorization ID of your SQL session. If you specify a value for ‘owner’, the DB2 SET CURRENT SQLID command is executed by the DB2 SQL interface at compile time. For the SET CURRENT SQLID command to execute successfully, you must have installed the CA-Pan/SQL Interface for a DB2 release of 2.1 or greater. You must also have the correct DB2 authorization to execute the SET CURRENT SQLID command. See your DB2 documentation for more information about the SET CURRENT SQLID command.

This parameter is in effect only for the compilation of your application program, unless your program is coded using automatic processing. If your program is coded using automatic processing, the SET CURRENT SQLID command is executed again at the start of runtime. For native SQL processing, you must code the SET CURRENT SQLID command in your program if you want to change the value for the current authorization ID.

See the CA-Easytrieve/Plus SQL Interface Option Guide for more information.

Use SQLSYNTAX to specify the level of SQL syntax checking that is to be performed on the SQL statements coded in your program.

Specify FULL to indicate that detail level syntax checking should be performed. An SQL PREPARE statement is executed by the CA-Pan/SQL interface for those SQL statements that can be dynamically prepared. If you specify FULL, your DBMS catalog must be available to CA-Easytrieve.

Specify PARTIAL to indicate that SQL statements in your program should be syntax checked for valid commands and secondary keywords. No connection is made to the DBMS catalog unless you have coded the SQL INCLUDE statement. If you coded an SQL INCLUDE statement, your DBMS catalog must be available to CA-Easytrieve. Your program cannot be executed until it has been fully syntax checked, as described above.

Specify NONE with a BIND STATIC-ONLY parameter if you want syntax checking to be performed by the DB2 preprocessor in a batch environment. NONE causes partial syntax checking, as described above. If no compile errors are found, your program executes, unless CA-Easytrieve errors are found. No connection is made to the DBMS catalog unless you have coded the SQL INCLUDE statement. If you coded an SQL INCLUDE statement, your DBMS catalog must be available to CA-Easytrieve.
If you specify NONE for a non-DB2 environment, partial syntax checking is performed, but the program is not executed until full syntax checking is performed.

[SSID 'ssid']

SSID is an SQL parameter. Currently, SSID is used by the DB2, SQL/DS and ORACLE SQL interfaces. The maximum length for SSID is 30 characters.

For DB2: You can use SSID to specify the DB2 subsystem ID, and/or a DB2 location ID for remote connections. If you specify a subsystem ID, it is used at both compile and runtime. The DB2 location ID is only used to establish a connection at compile time. For a remote connection at execution time, code an SQL CONNECT statement.

If you do not specify the DB2 subsystem ID, the subsystem ID from the Site Options Table is used.

<table>
<thead>
<tr>
<th>Valid Formats</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxx</td>
<td>DB2 subsystem ID.</td>
</tr>
<tr>
<td>xxxx/yyyyyyyy</td>
<td>DB2 subsystem ID and a DB2 location ID.</td>
</tr>
<tr>
<td>/yyyyyyyy</td>
<td>DB2 location ID.</td>
</tr>
</tbody>
</table>

If no DB2 subsystem ID is specified in the Options Table, the SQL interface uses the ID from the DB2 system default module DSNHDECP. The value of the subsystem ID is obtained at compile and runtime dynamically, therefore, there is no need to recompile your program to change the ID. See your DB2 systems programmer or administrator for the default values defined for your DB2 system.

For SQL/DS: You can specify a database ID for an explicit connection.

For ORACLE: SSID can be used to specify a subsystem ID other than the local subsystem.

If the SSID value is four characters or less, it is assumed to be an OS/390 subsystem ID. The subsystem ID is concatenated to the userid and password as follows:

```
userid/password@M:subsysid
```

If the SSID value is greater than four characters, it is assumed to be a non-OS/390 ID, and, therefore a completely valued connection ID must be provided which is concatenated to the userid/password values. No @ or driver prefix is inserted by the interface. You must specify all special characters.
PARM Statement Parameters

[SYNTAX]

SYNTAX terminates CA-Easytrieve/Plus processing after the syntax check operation.

[USERID ('user-id2' ['password'])]

USERID is an SQL parameter. Currently, USERID is used by the SQL/DS, ORACLE, and CA-IDMS SQL interfaces. USERID is used by the SQL interface to establish a connection to the database to compile the application program.

For SQL/DS and ORACLE, you can use 'user-ids' to specify a valid userid and password for an explicit CONNECT.

For CA-IDMS, you can use 'user-id2' to specify the CA-IDMS dictionary name for an explicit CONNECT.

If you do not code USERID, an implicit connection occurs according to the rules of the given database system.

‘User-id2' and 'password' must be alphabetic literals, and each parameter has a maximum length of 30 characters.

USERID can be abbreviated as USER.

[VFM]

VFM establishes the work area parameters used by the CA-Easytrieve/Plus Virtual File Manager access method.

[literal-10]

Literal-10 specifies the amount of core storage made available for the buffer pool. Valid numeric values for literal-10 are 6 to 4096. Literal-10 represents 1024-byte units of storage.

[DEVICE {MEMORY}]

DEVICE MEMORY inhibits the use of an overflow device.

[DEVICE]

(VSE only) DEVICE defines the device type for VFM's overflow file. For programs executing on DOS AF2 or later, the specific disk device type is determined when the file is opened. DISK indicates that the value is to be taken from the DISK parameter of the options table. MEMORY inhibits the use of an overflow device. Other valid values are the device type codes of all supported devices.
PARM Statement Examples

The following examples illustrate typical uses of the PARM statement:

PARM for production ...

    PARM LINK(MYPROG) DEBUG(PMAP, DMAP) +
        SORT (MSG (ALL, PRINTER))
    FILE PERSNL FB(150 1800)
    %PERSNL
    JOB INPUT PERSNL NAME MYPROG
        PRINT REPORT1
    *
    REPORT REPORT1
    LINE NAME DEPT

PARM for program testing ...

    PARM ABEXIT (SNAP) +
        DEBUG (PMAP, DMAP FLDCHK, FLOW, +
            FLOWSIZ (20), STATE) VFM (10)
    FILE PERSNL FB(150 1800)
    %PERSNL
    JOB INPUT PERSNL NAME MYPROG
        PRINT REPORT1
    *
    REPORT REPORT1
    LINE NAME DEPT SALARY-CODE
This chapter describes the methods of defining data. Normally, you define data fields in the section of your program called the library. The library defines the data in terms of fields, records, and files. A typical file layout follows.

```
Employee's Address Field
|---------/\---------|
Employee Record { Jones, John J. 16822 Evergreen Chicago ...}

P { Hammond, Martha 422 Ash Ave. Evanston ...}
E { Gray, Frederick 16 Apple St. Lockport ...}
R { Freud, William G. 754 Lake St. Peotone ...}
S { ____________________________________________}
O { ____________________________________________}
N { ____________________________________________}
E { ____________________________________________}
L { ____________________________________________}
F { ____________________________________________}
I { ____________________________________________}
L { ____________________________________________}
E { ____________________________________________}

Individual field data can also be defined later in JOB or SORT activities as the need arises.
File Attributes

Use the FILE statement to describe a file or a database. This discussion of the FILE statement includes the following information:

- Syntax of the FILE statement
- File types that CA-Easytrieve/Plus processes
- EXIT option
- Optional parameters specifying the device type for SAM files
- Record format of SAM, ISAM, and VSAM files for OS/390 and VSE programs
- FILE statement examples.

Field Data

Fields are defined in the library following the FILE statement, or later in the job by using the DEFINE statement. Two categories of data can be defined:

1. File data
   - fields defined within a record
2. Working storage data
   - fields defined in working storage

This chapter’s discussion of the DEFINE statement includes:

- Syntax of the DEFINE statement
- DEFINE statement examples
- Rules applying to signed and unsigned fields
- Edit MASK options.

COPY Statement

The COPY statement duplicates the field definitions of a named file to another file to alleviate re-entering the same definition.

Data Reference

Rules for referencing data in an activity and for referencing system-defined fields are discussed at the end of the chapter.
FILE Statement

Syntax

FILE statements must describe all files and databases that your program references. Code FILE statements at the beginning of the job after the PARM statement, if one is used. Not all parameters are necessary (or valid) for describing any one file. A review of all parameters quickly indicates those required for any particular file.

Code the optional parameters and subparameters of the FILE statement in any order following the file-name. As shown, you must code multiple subparameters within parentheses. The complete syntax of the FILE statement is shown here:

```plaintext
FILE  file-name  +
    [SYSxxx]  +
    [IS ]
    [VIRTUAL [RETAI
    [ ]
    [DLI (  
    [  [ dbd-name [ literal-2] ]
    [  [ CREATE [RESET] ]
    [V5 ([ES] [F] [PASSWORD 'literal-3'] [UPDATE ] [NOVERIFY]) ]
    [SQL ([select-clause]) ]
    [HOSTDISK ( {TO  ( field-name-1 field-name-2 ) } ]
    [  [ field-name-3 [field-name-4] ]
    [  [ FROM ( {'literal-4'} {'literal-5'} ) ]
    [  [ {  [field-name-5] [field-name-6] ]
    [  [ HOSTFILE ( {'literal-6'} {'literal-7'} ) ]
    [  [ {field-name-7} ]
    [  [ FORMAT ( {'literal-8'} ) [RESET] ]
    [  [ {  [field-name-8] ]
    [  [ EXIT (program-name [NR] [USING {field-name-8} ... ] [MODIFY]) ]
    [  [ CARD ]
    [  [ PUNCH ]
    [  [ PRINTER ]
    [  [ DISK [literal-10] ]
    [  [ TAPE ( [NL [literal-11] [UNLOAD] ] [REWIND] ]
    [  [ TAPE ( [NL [literal-11] [UNLOAD] ] [REWIND] ]
    [  [ TAPE ( [NL [literal-11] [UNLOAD] ] [REWIND] ]
    [  [ TAPE ( [NL [literal-11] [UNLOAD] ] [REWIND] ]
```
FILE Statement Parameters

file-name

File-name is a one to eight-character name (one to seven in VSE) used to define the file to CA-Easytrieve/Plus. All I/O statements that operate on the file refer to this name. File-name is also used on the JCL statement to reference the file. For IMS/DLI files, the file-name is an arbitrary one to eight-character name. Every FILE statement must have a file-name immediately following the FILE keyword. File-names must be unique within your program. The first three characters of file-name must be different from the value of the work data set name prefix option (WKDSNPF), discussed in Appendix C, “Options Table.”

[SYSxxx]

SYSxxx (VSE only) optionally establishes logical unit assignments. Valid entries are SYSLST, SYSPCH, SYSIPT, and SYS000 through SYS240.
Note: The use of IGN on the ASSGN statement in the JCL is not supported.

File Types

CA-Easytrieve/Plus processes all standard IBM file types including Sequential Access Method (SAM), Index Sequential Access Method (ISAM), Virtual Storage Access Method (VSAM), IMS/DLI, and the Virtual File Manager (VFM) of CA-Easytrieve/Plus. Also, it can process other file types using user supplied exits. If you do not specify a file type, CA-Easytrieve/Plus assumes the file is sequential. CA-Easytrieve/Plus processes all file types as both input and output, with the single exception of ISAM files which are processed as input-only files.

[IS]

IS designates an ISAM file. ISAM files are input-only files.

[VIRTUAL]

VIRTUAL identifies a file as a CA-Easytrieve/Plus virtual (VFM) file. CA-Easytrieve/Plus virtual files are temporary sequential work files that are normally deleted after execution of the activity in which they are read.

[RETAIN]

RETAIN inhibits the automatic deletion of a VFM file after it is read. The file is deleted after all activities have been processed.

[DLI]

DLI designates an IMS/DLI database. You must code a FILE statement for each Program Control Block (PCB) that is used.

{literal-1}

The relative position of the PCB in the PSB. Literal-1 must be a positive numeric literal. This enables you to access the IOPCB by specifying a 1 for literal-1.

{dbd-name}

Dbd-name names the Database Definition (DBD) in the Program Specification Block (PSB) to be processed.

[literal-2]

Literal-2 specifies the relative occurrence of like named DBD's within the PSB. Literal-2 is required only when two or more DBDs have the same name. It is valid only when it is coded with the dbd-name value.
FILE Statement Parameters

[RESET]

The RESET option causes all record areas to be set to binary zero prior to root segment retrieval. (See the CA-Easytrieve/Plus IMS/DLI Interface Option Guide.)

[IDMS]

IDMS designates a CA-IDMS database. You must code a FILE statement for each SUBSCHEMA that is used. Subschema-name names the database to be processed.

[RESET]

The RESET option causes all record areas to be set to binary zero prior to root record retrieval.

[VS]

VS designates a VSAM file. Default parameters for VSAM file processing are Key Sequenced Data Sets (KSDS) or Relative Record Data Sets (RRDS) with records of undefined length.

[ES]

Code ES to indicate an Entry Sequenced Data Set (ESDS).

[F]

Code F if all records in the file are equal in length to the maximum length defined for the file. This improves the efficiency of SORT activities.

[PASSWORD 'literal-3']

'Literal-3' is the optional one- to eight-character password for the VSAM file. You can specify the password as either an alphabetic literal or a hexadecimal quoted literal. Enclose these literals in single quotes. No DBCS data can be contained in literal-3.

[CREATE]

Use the CREATE option to load a VSAM file.

[RESET]

You can reload an existing file by coding the RESET option with CREATE. RESET assumes that the file was defined by IDCAMS with the VSAM attribute REUSE.

[UPDATE]

Code the UPDATE option if you wish to update the file with the WRITE statement.
[NOVERIFY]

Code NOVERIFY to ignore the VSAM open error code of 116(X'74'). This error occurs when the file is not properly closed by a previous job, or is being used by another CPU.

Caution: Indiscriminate use of this option can cause loss of data records.)

SQL [(select-clause)]

This parameter identifies the file as an SQL file. The only other options that can be used with SQL are DEFER and DBCSCODE. For details on specifying this option see the appropriate database interface option guide (DB2 * SQL/DS or ORACLE).

[HOSTDISK]

This parameter identifies this file as a CA-Corporate Tie Host Disk file. CA-Corporate Tie is a Computer Associates product that provides a transparent link between the mainframe and Personal Computers (PCs). A Host Disk is a CA-Corporate Tie file transfer facility. It transfers files between the mainframe and the PC.

In CA-Easytrieve/Plus, a HOSTDISK file is an output only file. When CA-Easytrieve/Plus writes a HOSTDISK record it converts the record field by field from a mainframe format to a PC format. The only other options that can be used with HOSTDISK are DEFER and PRINTER. See Chapter 12, “File Processing,” for more information about processing Host Disk files.

{TO ( field-name-1   field-name-2 )}

The TO parameter field-name-1 specifies a list of CA-Corporate Tie USER IDs or GROUP IDs who receive the file exported to the Host Disk. The TO parameter field-name-2 specifies a check list that indicates which of the receiving IDs are known to CA-Corporate Tie. Both parameters are required.

Field-name-1 must specify an eight-byte alphanumeric field that can occur a maximum of 50 times. To specify a single user, define a field-name-1 as an eight-byte alphanumeric, but without an OCCURS clause.

Field-name-2 must specify a one-byte alphanumeric field that must occur exactly the same number of times as field-name-1. When field-name-1 does not specify an OCCURS clause, you can not specify an OCCURS clause for field-name-2.
FILE Statement Parameters

\{FROM (\{'literal-4\'} [\{'literal-5\'}])\}

This parameter supplies the CA-Corporate Tie userid of the sender and, optionally, the userid's password. The password is not required if you are using the CA-Corporate Tie External Security Interface. It is used by CA-Corporate Tie to determine your authority to access the system and files on the Host Disk. Field-name-3 and field-name-4 must be alphanumeric fields eight or fewer bytes in length. Literal-4 and literal-5 must be alphanumeric literals eight or fewer bytes in length.

\{HOSTFILE (\{'literal-6\'} [\{'literal-7\'}])\}

The HOSTFILE parameter names a Host Disk file and, optionally, its Host Disk set. Field-name-5 must be an alphanumeric field, 12 or fewer bytes in length. Literal-6 must be an alphanumeric literal, 12 or fewer bytes in length. The 12-byte maximum consists of a filename eight or fewer bytes in length, a period, and an extension of three bytes or less in length. The naming convention for the filename must conform to the requirements of the specified PC format.

Field-name-6 must be an alphanumeric field eight or fewer bytes in length. Literal-7 must be an alphanumeric literal eight or fewer bytes in length.

\{FORMAT (\{'literal-8\'})\}

The FORMAT parameter specifies the exact PC format of the file. CA-Easytrieve/Plus converts the file to the PC format specified by the FORMAT parameter. Defined formats are LOTUS, DBASEIII, PRN, EZTPC, and BASIC. PRN is the only allowed format when the PRINTER keyword is specified. Field-name-7 must be an alphanumeric field eight or fewer bytes in length. Literal-8 must be an alphanumeric literal eight or fewer bytes in length.

[RESET] )

The RESET parameter sets the CA-Corporate Tie Overwrite File Flag to YES. This flag is used to indicate if you want to purge existing files of the same name. If RESET is not coded, the Overwrite File Flag is set to NO, and an additional file is created on the Host Disk.
FILE Statement Parameters

[EXIT]

The EXIT option invokes a user-written program for each CA-Easytrieve/Plus input and/or output operation for the file. EXIT is not valid for VFM or IMS/DLI. For ISAM and VSAM files, you must include the MODIFY option. You must also specify the WORKAREA option if the exit performs input or output services; you need not specify WORKAREA if you are using the MODIFY function for other than ISAM or VSAM files. (See Chapter 14, “Subprograms,” for more information.)

program-name [NR]

Program-name specifies the name of the user program. In VSE, the parameter NR following the program-name indicates that the program is not relocatable.

[USING ({field-name-8} ) ]
{literal-9 }

The USING option appends the associated parameters (field-name-8 or literal-9) to the standard parameter list passed to the exit program. Field names must be working storage or system defined fields and must be defined in the library section. If field-name-8 contains DBCS data, that data belongs to the DBCS code system defined as the Processing code. The field’s format remains unchanged. There is a limit of 62 fields that can be passed to the exit program.

[MODIFY]

The MODIFY option specifies that CA-Easytrieve/Plus provides input or output services, but the exit can inspect and modify each record after input and before output.

Device Types

The optional parameters CARD, PUNCH, PRINTER, DISK, and TAPE specify the device type for SAM files. For all VSE files, code one of these parameters (or enable the default from PARM or the Options Table). For OS/390, if you do not specify one of these parameters, the device type is determined by your JCL. TAPE and DISK are meaningless to OS/390. Tape positioning in OS/390 is determined by the termination disposition subparameter of the DISP parameter coded on the DD statement in the JCL. These parameters are:

- PASS, which forwards space to the end of the data set on the tape
- DELETE, which rewinds the current tape, and
- KEEP, CATLG, UNCATLG, all of which position the tape to reprocess the data set.
[CARD]

The CARD option retrieves the file data from the CA-Easytrieve/Plus system input stream (that is, by using SYSIN or SYSIPT). For syntax check, compile, and execute jobs, the data must follow an END delimiter. Only one file in a CA-Easytrieve/Plus execution can use the CARD option. Files using this option must be 80-character unblocked records.

[PUNCH]

The PUNCH option indicates punched card output. Files created with this option are 80-character unblocked records in OS/390 or 81-character unblocked records in VSE.

[PRINTER]

The PRINTER option indicates print output files. Although input/output statements (GET, PUT, READ, WRITE) cannot reference these printer files, the DISPLAY statement and REPORT statements can reference them. Unless otherwise coded, record length and blocksize default to one more than the LINESIZ entry in the Options Table. The additional character contains the ASA control character.

No fields can be defined for a PRINTER file. When coded for a file that specifies the HOSTDISK keyword, PRINTER must be specified before HOSTDISK and PRN is the only valid value for the HOSTDISK subparameter FORMAT.

[DISK [literal-10]]

This parameter is maintained for compatibility with prior releases, but is no longer used.

[TAPE]

(VSE only) The TAPE option indicates that the file is on magnetic tape.

[NL [literal-11]]

Use the NL parameter to specify nonlabeled tapes. Literal-11 specifies the number of tapes on which the file resides. The default is standard tape labels. The literal-11 default is one reel.

[REWIND]

Rewind the tape before each use.

[UNLOAD]

Rewind and unload the tape after use.
[NORWD]

Do not rewind either before or after tape use.

The default for tape rewinding and unloading is obtained from the Options Table.

**Record Format**

You can optionally code the record format of SAM and ISAM files for OS/390 programs, but you must code it for VSE programs. If you do not code record format in OS/390, CA-Easytrieve/Plus obtains it from the operating system when the file is opened.

```plaintext
[F, V, U]
[FB, VB]
```

CA-Easytrieve/Plus supports fixed (F), variable (V), and undefined (U) formats. Fixed and variable length records can be blocked (FB, VB).

[VBS]

(OS/390 only) OS/390 systems can process Variable Blocked Spanned (VBS) records using BFTEK=A processing. Spanned is valid only for variable blocked records.

literal-12

Literal-12 specifies the maximum record length.

{literal-13}

Literal-13 specifies the file's maximum block length.

For variable format files, permit four bytes of the record length for the Record Description Word (RDW) and, if the file is blocked, four bytes of the block size for the Block Description Word (BDW).
FILE Statement Parameters

{FULLTRK}

A block length designation of FULLTRK establishes an output block size that equals either the maximum track capacity of the direct access device or the next lower multiple of the record size for FB files. FULLTRK should not be used with tape output or FBA disk devices.

Note: To obtain an OS/390/DFP system determined block size within CA-Easytrieve/Plus you can do one of the following:

- Include the DSORG, LRECL and RECFM parameters in the original JCL or TSO dynamic allocation. This forces SMS to establish the block size before CA-Easytrieve/Plus gets control in OPEN processing. This is only applicable for disk data sets.

OR

- Define a value of zero for the block length value (literal-13). If you want CA-Easytrieve/Plus to pick up the logical record length from your JCL, code a zero for literal-12. You must also code a BLKSIZE=0 in your JCL or code no BLKSIZE parameter at all. For TAPE data sets, a zero blocksize must be coded on the FILE statement.

Examples:

FILE file-name FB(0 0)
This tells CA-Easytrieve/Plus to pick up the LRECL from the JCL and to utilize the block size set by System Management Storage (SMS).

FILE file-name FB(150 0)
This tells CA-Easytrieve/Plus to pick up the LRECL from this definition and to utilize the block size set by SMS.

FILE file-name FB(150 3000)
This tells CA-Easytrieve/Plus to pick up this definition and ignore both the JCL and the SMS determined block size.

Note: If you code a zero block size within CA-Easytrieve/Plus and/or in your JCL, and your data set is not SMS managed, your program abends with a 013 open problem.

[WORKAREA literal-14]

The WORKAREA option establishes the number of bytes to be allocated as a work area for the file. WORKAREA cannot be coded if CARD, DLI, or IDMS parameters are specified. Literal-14 specifies the number of bytes to be allocated and must be large enough to contain the longest record processed. This forces “move mode” instead of the default “locate mode.”

[CISIZE literal-15]

(VSE only) The CISIZE option sets literal-15 as the control interval size for sequential output on FBA devices.
The TABLE option declares the file as the source for a SEARCH statement to access a table.

The INSTREAM option indicates that the table file immediately follows the file description. The size of an INSTREAM table is limited only by the amount of available memory. Literal-16 specifies the maximum number of entries in an external table. If INSTREAM or literal-16 is not specified, the file is an external table whose maximum number of entries is limited by the Options Table parameter TBLMAX. See Chapter 13, “Table and Array Processing,” for more information about table processing.

BUFNO establishes the number of buffers allocated for the file. Literal-17 can be 1 or 2 for VSE programs and 1 through 255 for OS/390 programs. The default value is obtained from the Options Table.

Coding the DEFER option instructs CA-Easytrieve/Plus to delay the opening of the file until the first input or output operation for the file occurs. The default opens all referenced files at the beginning of each CA-Easytrieve/Plus activity.

(OS/390 only) This optional parameter sets the DCB A option for RECFM.

The EXTENDED option indicates that the file is to be associated with an extended reporting printer. This means that input/output statements (GET, PUT, READ, WRITE) cannot reference these printer files. However, the DISPLAY statement and REPORT statements can reference these printer files. Unless you code them, record length and blocksize default to those defined for the printer in the extended reporting options module.

The xrpt-printer entry identifies the extended reporting printer whose characteristics are to be associated with this file. You must define the xrpt-name in the extended reporting options module. See the CA-Easytrieve/Plus Extended Reporting Facility guide for more information about extended reporting.
FILE Statement Parameters

[DBSCODE]

DBSCODE defines the CA-Easytrieve/Plus code system that is associated with all CA-Easytrieve/Plus fields defined for this file. If you do not code this option, then CA-Easytrieve/Plus uses the Processing code system. The DBCS Options module identifies this code system. You can alter the code system through using the DBSCODE option of the PARM statement. If your installation does not support the DBCS option, then this option is invalid.

If the file being defined is associated with an extended reporting printer that does not support DBCS data, then this keyword is invalid. Should the extended reporting printer support DBCS data, then you can use the DBSCODE keyword to modify the DBCS code system of the printer.

Examples

The following examples illustrate FILE statements for various files. See Chapter 12, “File Processing,” for more information about processing files.

The first two examples below define the same file in OS/390 and VSE. In VSE, you must specify the data set attributes.

Define a sequential (SAM) file in OS/390:

```
FILE SEQFILE
```

Define a SAM file in VSE:

```
FILE SEQFILE FB(80 1600) DISK(3340)
```

Define an entry-sequenced, fixed-length VSAM file to be loaded:

```
FILE ENTSEQ VS(ES, F, CREATE (RESET))
```

Define a VSAM file for exit processing:

```
FILE XITFILE,+
  VS,+
  EXIT(XITNAME,+
      USING(XITFILE:RECORD-LENGTH,+
        'LIT1',+
        PARM2) +
      MODIFY) +
  WORKAREA(400)
```

Define a virtual file and RETAIN it for the duration of the CA-Easytrieve/Plus job processing:

```
FILE VRTFILE V(200) +
  VIRTUAL RETAIN
```
DEFINE Statement

DEFINE within an Activity

The DEFINE statement specifies data fields within a record or within working storage. You usually specify file fields and work fields in your CA-Easytrieve/Plus library section, but you can also define them within an activity as the following examples illustrate:

```
Define EMP#       9    5  N
Environment
Define NAME      17   20  A
Library
Define EMP-COUNT W    4  N
Report
```

When fields are defined within an activity, each field definition must start with the DEFINE keyword and physically be defined before the field is referenced. In the library, the use of the DEFINE keyword is optional.

File Fields

File fields are normally defined immediately following the associated FILE statement in the library section of the CA-Easytrieve/Plus program. Their rules of usage are:

1. CA-Easytrieve/Plus accepts an unlimited number of fields for each file.
2. Field names must be unique within a file.
3. You can define file fields anywhere in a CA-Easytrieve/Plus library section, or anywhere in a CA-Easytrieve/Plus activity except within a REPORT procedure.
Working Storage Fields

Working storage fields are normally defined in the CA-Easytrieve/Plus library section. Their rules of usage are:

1. CA-Easytrieve/Plus accepts an unlimited number of working storage fields.
2. Working storage fields must be uniquely named within working storage.
3. You can define working storage fields anywhere in a CA-Easytrieve/Plus library section, activity, or REPORT procedure.

Basic Field Definition

The basic field definition consists of the DEFINE keyword, the field name, its location, and its attributes. The sample definition below uses the field name FIELDA, has a location starting in position 100 in the record, and has the attribute 5 for length and A for alphabetic.

<table>
<thead>
<tr>
<th>Field-name</th>
<th>Location</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINE FIELDA</td>
<td>100</td>
<td>5 A</td>
</tr>
</tbody>
</table>

The complete syntax of the DEFINE statement follows.
**DEFINE Syntax**

Code the DEFINE statement in the library section of the program directly after the FILE statement or anywhere else in the activity. You must code the first three parameters (field-name-1, location, attributes) in the order shown in the following example. As indicated, code multiple subparameters within parentheses. The complete syntax of the DEFINE statement is:

```plaintext
DEFINE [file-name-1:]field-name-1 +
    { integer-1
      * [+integer-2] } +
    Location { W } +
      { S } +
      {{file-name-2:]field-name-2 [+integer-3]}
    { [A] } +
      { [K] } +
      { [M] } +
      { [N] [+integer-5] [EVEN] } +
    Attributes integer-4 { [P] +
      [B] +
      [U] +
    { [file-name-3:]field-name-3
      [VARYING] +
      [HEADING ([#integer-6] 'literal-1' ...)] +
      [INDEX (field-name-4 ...)] +
      { [
        [letter] [BWZ] [KANJI] ['literal-2'] ] +
        [MASK ({ HEX }{ }{ }) +
          [OCCURS integer-7] +
          [VALUE literal-3] +
          [RESET]
```
**Parameters**

**DEFINE**

You can omit the DEFINE keyword for fields defined immediately after the associated FILE statement or for working storage fields defined after any FILE statement. For definitions outside the library, the DEFINE keyword must precede each field definition and you must specify file-name-1 to identify the appropriate file/record or working storage.

```
[file-name-1:]field-name-1
```

Field-name-1 is the name of the field you are defining. It can be from 1 to 40 alphanumeric characters in length, can contain any character other than a delimiter, and must begin with A-Z or 0-9; it cannot consist of all numeric characters. File-name-1 identifies the appropriate file/record or working storage for field-name-1.

**Location**

You must establish the location of the field's left-most (starting) position in one of the following ways:

```
{integer-1}
```

Integer-1 specifies the starting position relative to position 1 of the record.

```
{* [+integer-2]}
```

The * (asterisk) indicates that the field begins in the next available starting position (highest location defined so far, plus 1). The optional +integer-2 is an offset you want to add to the * value. There must be at least one blank between the * and the optional +integer-2.

```
{W or S}
```

Coding a W or S establishes a working storage field. CA-Easytrieve/Plus spools W fields to report (work) files; it does not spool S fields. For an example, see Chapter 11, “Report Processing: Report Procedures.”

```
[[file-name-2:]field-name-2 [+integer-3]]
```

Specify field-name-2 if you want an overlay redefinition. Specify the optional file-name-2 if the redefined field is in a file or record other than the file or record currently being defined. The optional +integer-3 permits you to offset the field from the beginning of field-name-2. If you use overlay redefinition, make sure that field-name-1 fits in the storage boundaries of field-name-2. Any indexes associated with field-name-2 also apply to field-name-1.

If both file-name-1 and file-name-2 are specified, they must identify the same file.
Attributes

For each field-name-1 you define, you must also specify the following attributes:

- Field length in bytes
- Data format
- Number (if any) of decimal positions.

\{integer-4\}

Integer-4 specifies the length (in bytes) of the defined field.

Data Format

Specify data format by selecting one of the following codes:

- \{A\} — alphanumeric — use when none of the numeric data types apply to the associated field.
- \{K\} — DBCS alphanumeric — use when the data in the associated field is known to be in DBCS format. The field's length must be a multiple of two. The data in this field is associated with the DBCS code system defined as the DBCS Processing code unless the field belongs to a file that has the DBCSCODE option coded on its FILE statement.
- \{M\} — MIXED alphanumeric — use when the data in the associated field is known to be either EBCDIC, DBCS, or a mixture of both. CA-Easytrieve/Plus processes the field assuming that it contains EBCDIC data. The DBCS data in this field must be identified by the shift codes of the field's DBCS code system. The field's DBCS code system is assumed to be the DBCS Processing code system unless the field belongs to a file that has the DBCSCODE option coded on its FILE statement. This field type is invalid for those DBCS code systems that do not have an assigned shift code system and therefore cannot support a MIXED field type.
- \{N\} — zoned decimal — the field contains digits 0 through 9 in external decimal form (that is, 0 = X'F0').
- \{P\} — packed decimal — the field contains numbers which meet IBM's definition of internal packed decimal. For instance, the two-byte packed field containing 123 looks like X'123F'.
- \{B\} — binary — the fields contain binary data. In a quantitative binary field (a field with zero (0) or more decimal places specified), the high order bit is the sign bit. In a non-quantitative binary field (a field with no decimal place specification), the high order bit is a binary digit.

For example, in a one-byte quantitative field the following is true:

\(\text{HEX) } 7F = \text{(BIN) } 0111 1111 = \text{(DECIMAL) } 127\)
\(\text{HEX) } 80 = \text{(BIN) } 1000 0000 = \text{(DECIMAL) } 128\)
For a one-byte non-quantitative binary field the following is true:

(HEX) 7F = (BIN) 0111 1111 = (DECIMAL) 127
(HEX) 80 = (BIN) 1000 0000 = (DECIMAL) 128

The following tables show the length equivalent and maximum possible values for both signed and unsigned binary fields:

### Signed Binary Fields (Quantitative)

<table>
<thead>
<tr>
<th>field length in bytes</th>
<th>maximum positive value</th>
<th>maximum negative value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>127</td>
<td>128-</td>
</tr>
<tr>
<td>2</td>
<td>32,767</td>
<td>32,768-</td>
</tr>
<tr>
<td>3</td>
<td>8,388,607</td>
<td>8,388,608-</td>
</tr>
<tr>
<td>4</td>
<td>2,147,483,647</td>
<td>2,147,483,648-</td>
</tr>
</tbody>
</table>

### Unsigned Binary Fields (Non-Quantitative)

<table>
<thead>
<tr>
<th>field length in bytes</th>
<th>maximum unsigned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>2</td>
<td>65,535</td>
</tr>
<tr>
<td>3</td>
<td>16,777,215</td>
</tr>
<tr>
<td>4</td>
<td>2,147,483,648</td>
</tr>
</tbody>
</table>

- {U} — unsigned packed decimal — used to reference part of a packed decimal field. For instance, the two-byte unsigned packed field containing 123 looks like X'0123'.

The maximum length and allowable decimal places for each field type are:

<table>
<thead>
<tr>
<th>Data Format Code</th>
<th>Maximum Length (bytes)</th>
<th>Maximum Number of Decimal Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>32,767*</td>
<td>not valid</td>
</tr>
<tr>
<td>K</td>
<td>32,766*</td>
<td>not valid</td>
</tr>
<tr>
<td>M</td>
<td>32,767*</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>

* For table file fields, ARG (argument) and DESC (description), the maximum length is 254. For more information on table files see Chapter 13.

{{integer-5}}

Integer-5 is an option that specifies the desired number of decimal positions for field-name-1. Integer-5 cannot be specified for data format 'A'. If integer-5 is specified, the field is considered to be quantitative.
**[EVEN]**

Use **EVEN** to indicate that a packed decimal field (P) is to contain an even number of digits. The high order digit is zero. For example, a two-byte packed even field can only contain two digits, such as X'012F'.

**Note:** **EVEN** is valid only for P fields.

`[file-name-3:]field-name-3`

When you use the optional `file-name-3`, the field length, data format, and number of decimal positions that you specified for `field-name-3` are duplicated for `field-name-1`. If `field-name-3` is in a different file or record, specify the name of that file or record.

**[VARYING]**

The **VARYING** option indicates that `field-name-1` is a varying length field. This means that the length of this field, for each occurrence in separate records, is unique. Varying length fields are alphanumeric and consist of a two-byte length value followed by the data.

You can specify **VARYING** on A, K, and M fields. When **VARYING** is specified, the length attribute (integer-4) is the total number of bytes the varying length field can occupy (that is, two-byte length plus maximum size of data).

You can specify **VARYING** for file fields or working storage fields. For file fields the starting position (integer-1) points to the two-byte length indicator. For both file fields and working storage fields, overlay redefinition begins with the two-byte length indicator.

When referencing a **VARYING** field in your job, the field name (`field-name-1`) can be used alone or suffixed as shown below.

Assume `field-name-1` is FLDA, then:

- FLDA references the entire field (both length and data) as a variable length field.
- FLDA:LENGTH references only the length (first two bytes) as a two-byte binary field.
- FLDA:DATA references the data portion of the field (from byte three on) as an alphanumeric field.

When a **VARYING** field is displayed in your output, the data window is based on the maximum length of the field (integer-4 minus two). The length indicator does not display in output unless **DISPLAY HEX** is specified.
Length restrictions for varying length fields are as follows:

<table>
<thead>
<tr>
<th>field type</th>
<th>minimum length</th>
<th>maximum length</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>32769</td>
</tr>
<tr>
<td>K</td>
<td>4</td>
<td>32768</td>
</tr>
<tr>
<td>M</td>
<td>4</td>
<td>32769</td>
</tr>
</tbody>
</table>

The default value for a varying field is a string of zero (0) length. However, if the VALUE option is coded, its value and length become the default for the field.

See examples of Varying Length Fields under “DEFINE Statement Examples” later in this chapter.

[HEADING ([#integer-6] 'literal-1' ...)]

The HEADING option specifies an alternate report heading for field-name-1 (the default is the actual field-name).

Integer-6 defines the font number of the font to be used to format 'literal-1' when you use field-name-1 in a report directed to an extended reporting printer. Any 'literal-1' that does not have a font assigned uses the default font identified in the extended reporting options module. If the report is directed to a normal printer, CA-Easytrieve/Plus ignores this value. See the CA-Easytrieve/Plus Extended Reporting Facility guide for more information about extended reporting.

'Literal-1' specifies the alternate heading and can be up to 40 characters long. 'Literal-1' must be enclosed in parentheses. Multiple literal-1's enclosed in the parentheses are stacked vertically over the field when printed. For detailed examples, see Chapter 11, “Report Processing: Report Definition Statements,” and the example in the Alternate Report Headings section.

[INDEX (field-name-4 ...)]

The INDEX option establishes indexes for field-name-1. Field-name-4 provides the name for the index. You can specify multiple indexes by coding a list of index names enclosed in parentheses.

CA-Easytrieve/Plus automatically allocates a four-byte field for each index. Any references you make to a field with the INDEX option cause that field's location to be adjusted by the index amount. See Chapter 13, “Table and Array Processing” for more information about indexing.

[MASK]

The MASK option establishes a pattern (print edit mask) for field-name-1.
Any letter from A through Y can be used as an optional mask identifier. You can use the letter to identify a new MASK or to retrieve a MASK that was previously defined either in the Options Table or by a MASK parameter on a previous field definition. If the new MASK that you identify does not already exist, CA-Easytrieve/Plus retains the mask for future reference. If you subsequently reference field-name-1 for printing, CA-Easytrieve/Plus automatically uses the associated letter identifier to determine the print MASK. Do not use the same identifier to establish more than one mask. You can define 192 unidentified Edit Masks and 25 identified Edit Masks, for a total of 217 Edit Masks.

The BWZ (blank when zero) option suppresses the printing of field-name-1 when it contains all zeros. BWZ can be used by itself or with other options on the MASK parameter.

The KANJI option converts the value in field-name-1 into its equivalent DBCS format. CA-Easytrieve/Plus performs the conversion after the Edit Mask or pattern has been applied to the numeric value in field-name-1 but before field-name-1 is printed. If the field-name-1 is zero and you have coded the BWZ option, the output displays DBCS spaces.

'Literal-2' defines an edit mask and must be enclosed within apostrophes. The actual print edit mask is coded according to the rules specified later in this chapter under the subject heading Edit Masks. System default masks for numeric fields are:

<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</td>
</tr>
<tr>
<td>1</td>
<td>ZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ</td>
</tr>
<tr>
<td>2</td>
<td>Z,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ.999</td>
</tr>
<tr>
<td>3</td>
<td>ZZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ.9999</td>
</tr>
<tr>
<td>4</td>
<td>ZZ,ZZZ,ZZZ,ZZZ,ZZZ.999999</td>
</tr>
<tr>
<td>5</td>
<td>Z,ZZZ,ZZZ,ZZZ,ZZZ.99999997</td>
</tr>
<tr>
<td>6</td>
<td>ZZ,ZZZ,ZZZ,ZZZ.9999999999</td>
</tr>
<tr>
<td>7</td>
<td>Z,ZZZ,ZZZ.9999999999999</td>
</tr>
<tr>
<td>8</td>
<td>ZZ,ZZZ.99999999999999998</td>
</tr>
<tr>
<td>9</td>
<td>ZZ.99999999999999999999997</td>
</tr>
<tr>
<td>10</td>
<td>ZZ.99999999999999999999999999999999999</td>
</tr>
<tr>
<td>11</td>
<td>ZZ.999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>12</td>
<td>ZZ.9999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>13</td>
<td>ZZ.99999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>14</td>
<td>ZZ.999999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>15</td>
<td>ZZ.9999999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>16</td>
<td>ZZ.99999999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>17</td>
<td>ZZ.999999999999999999999999999999999999999999999</td>
</tr>
<tr>
<td>18</td>
<td>ZZ.9999999999999999999999999999999999999999999999</td>
</tr>
</tbody>
</table>

* For zoned decimal fields with no decimals, the default mask is '999999999999999999'.
Default Edit Masks are defined in the Options Table when CA-Easytrieve/Plus is installed.

{HEX}

HEX is a special edit mask which instructs CA-Easytrieve/Plus to print the contents of field-name-1 in double-digit hexadecimal format. You can print fields of up to 50 bytes in length with the HEX mask, as the following example illustrates:

```
SOCIAL-SECURITY-NUMBER  4  5  P  MASK HEX
```

Please note that HEX edit masks are not allowed for VARYING fields.

[OCCURS integer-7]

The OCCURS option establishes an array for field-name-1. Integer-7 specifies the number of elements in the array (the number of occurrences of field-name-1). You can reference the elements of this array by manipulating the INDEX for field-name-1, if defined or a subscript. The maximum value 04 integer-7 is 32767. For more information about arrays, see Chapter 13, “Table and Array Processing.”

(VALUE literal-3]

The VALUE option initializes the contents of a field in working storage. Literal-3 can be any valid literal whose type matches the field-name-1 type. If literal-3 is non-numeric, it must be enclosed in apostrophes. You cannot use the VALUE option on an overlay redefinition, unless the original definition is an alphanumeric field. The maximum length for literal-3 is 254 bytes.

RESET

The RESET option is used only for W working storage fields. When coded on the field definition for a W field, RESET returns the field to its initial value whenever JOB or SORT is executed. RESET can be used with the OCCURS option for array fields but cannot be used for redefined fields (fields having overlay redefinition). When RESET is specified on a field in conjunction with the OCCURS option, the length of the resulting array cannot be greater than 65,520 bytes. Multiple fields specified with RESET options are reset in the order of the field definitions. The only exception to RESET is when W working fields are referenced in report processing. RESET is not performed during the printing of spooled reports.
DEFINE Statement Examples

The examples below illustrate two ways of describing a record from a personnel file. The first method uses an '*' to define the starting location of the fields. The second method uses absolute starting positions. In this case, both methods result in the same description.

**Note:** The DEFINE keyword is not needed when the field definitions immediately follow the FILE statement.

### Record Layouts

```
---------------- REGION
|--------------- BRANCH
|| ------------- SSN
|| | ------- EMP#
|| | 0000000001111 ... 78 (column number)
1234567890123      90

Method 1
FILE PERSNL FB(150 1800)
REGION * 1 N
BRANCH * 2 N
SSN * 5 P
EMP# * 5 N
JOB INPUT PERSNL NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1
LINE EMP# REGION BRANCH

Method 2
FILE PERSNL FB(150 1800)
REGION 1 1 N
BRANCH 2 2 N
SSN 4 5 P
EMP# 9 5 N
JOB INPUT PERSNL NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1
LINE EMP# REGION BRANCH
```
Overlay Redefinition

Overlay redefinition is yet another method of describing the personnel record, as the next example illustrates. NAME-LAST and NAME-FIRST overlay and redefine NAME.

```
FILE PERSNL FB(150 1800)
NAME              17 20 A
NAME-LAST          8  A
NAME-FIRST         +8 12 A
JOB INPUT PERSNL NAME MYPROG
*                  PRINT REPORT1
REPORT REPORT1     LINE NAME-FIRST NAME-LAST
```

When redefining a varying length field, redefinition begins with the length indicator (first two bytes) of the varying length field.

Working Storage Initialization

CA-Easytrieve/Plus initializes numeric work fields to zeros and alphabetic work fields to blanks. To initialize these fields to other values, use the VALUE option, as the following example shows:

```
DEFINE CURRENT-MONTH W 10 A VALUE 'JANUARY'
```

Varying Length Fields

The VARYING parameter on the DEFINE statement designates varying length fields. An example of a varying length field definition is shown below.

```
FLDA W 250 A VARYING
```

Because VARYING is used, this W type work field has two parts which are internally defined as follows:

- `W 2 B 0` for the two-byte field length
- `W 248 A` for the data

When this field is referenced in your job statements, you can designate the entire field, including the length, by specifying FLDA. Or, you can specify only the length portion or only the data portion of the field. For the W field defined above:

- `FLDA:LENGTH` references the binary portion only (bytes one and two)
- `FLDA:DATA` references the alphanumeric portion only (bytes 3 through 250)
- `FLDA` references the entire field (bytes 1 through 250)
Displaying Varying Length Fields

The display window for varying length fields is based on maximum length. In the above example, this is 250 bytes.

Normally, the length portion of the field is not displayed. However, when DISPLAY HEX is used, the length and the data are displayed. DISPLAY HEX displays the length and the full data field in hexadecimal and character format. The following example illustrates this:

```
Statements:
  DEFINE FLDA  W 7 A VALUE 'ABCD' VARYING
  JOB INPUT NULL
  DISPLAY FLDA
  DISPLAY HEX FLDA
  STOP

Produce:
  ABCD
  CHAR   ABCD
  ZONE 00CCCC4
  NUMB 0412340
```

Assigning and Moving Varying Length Fields

Assignments are based on the current length of the data and the rules of assignment. MOVEs default to the current length of the data. MOVE SPACES moves blanks according to the maximum possible length of the varying length field. The following shows some examples:

```
Statements:
  DEFINE SENDNULL W 10 A VARYING VALUE ''
  DEFINE SENDVAR W 10 A VARYING VALUE '12345678'
  DEFINE RECVVAR07 W  7 A VARYING
  DEFINE RECVVAR10 W 10 A VARYING
  JOB INPUT NULL
  RECVVAR10 = SENDNULL   * ASSIGN NULL TO A VARYING
  DISPLAY '1. VALUE=' RECVVAR10 +1 'LENGTH=' RECVVAR10:LENGTH
  RECVVAR10 = SENDVAR    * ASSIGN 10 BYTE VARYING TO 10 BYTE VARYING
  DISPLAY '2. VALUE=' RECVVAR10 +1 'LENGTH=' RECVVAR10:LENGTH
  RECVVAR07 = SENDVAR    * ASSIGN 10 BYTE VARYING TO 7 BYTE VARYING
  DISPLAY '3. VALUE=' RECVVAR07 +1 'LENGTH=' RECVVAR07:LENGTH
  RECVVAR10 = RECVVAR07   * ASSIGN 7 BYTE VARYING TO 10 BYTE VARYING
  DISPLAY '4. VALUE=' RECVVAR10 +1 'LENGTH=' RECVVAR10:LENGTH
  MOVE SPACES TO RECVVAR07   * MOVE SPACES TO 7 BYTE VARYING
  DISPLAY '5. VALUE=' RECVVAR07 +1 'LENGTH=' RECVVAR07:LENGTH
  MOVE SENDVAR TO RECVVAR07  * MOVE 10 BYTE VARYING TO 7 BYTE VARYING
  RECVVAR07:LENGTH = 2     * THEN OVERRIDE LENGTH TO 2
  DISPLAY '6. VALUE=' RECVVAR07 +1 'LENGTH=' RECVVAR07:LENGTH
  STOP

Produce:
  1. VALUE=          LENGTH=           
  2. VALUE=12345678  LENGTH=          8
  3. VALUE=12345       LENGTH=          5
  4. VALUE=12345       LENGTH=          5
  5. VALUE=           LENGTH=          5
  6. VALUE=12          LENGTH=          2
```
**Note:** If the sending field has a length of zero and the receiving field is a VARYING field, the receiving field will have a length of zero. If the sending field has a length of zero and the receiving field is not a VARYING field, the receiving field is filled with the fill character (blank for assigned, blank or specified fill character for MOVE).

### Alternate Report Headings

The default report heading for a field is the field name. Sometimes a field name can be cryptic or mismatched when compared to the width of the actual data. In these cases, you can use the HEADING option, as the following example illustrates:

```
FILE PERSNL FB(150 1800)
  EMP#   9  5 N  HEADING(’EMPLOYEE’ ’NUMBER’)
  PAY-NET 94  4 P 2  HEADING(’NET’ ’PAY’)
  PAY-GROSS 94  4 P 2  HEADING(’GROSS’ ’PAY’)
  WORK-FIELD  W  4 P 2  HEADING(’AMOUNT’ ’OF’ ’TAXES’)
JOB INPUT PERSNL NAME MYPROG
WORK-FIELD = PAY-GROSS - PAY-NET
PRINT REPORT1
  *
REPORT REPORT1
LINE EMP# PAY-GROSS PAY-NET WORK-FIELD
```

### Signed/Unsigned Rules

#### Signed

If you specify a numeric field with decimal positions (0 to 18), CA-Easytrieve/Plus considers it a signed (quantitative) field. The following rules apply to signed fields:

1. For binary numbers, CA-Easytrieve/Plus takes the high-order (left-most) bit as the sign, regardless of field length. In any manipulation, CA-Easytrieve/Plus shifts the field and propagates the high-order bit. For example, a one-byte binary field containing a hexadecimal FF has the numeric value -1.

2. For non-negative, zoned decimal numbers on the left side of an Assignment statement, CA-Easytrieve/Plus sets an F sign. Otherwise, it manipulates the number in packed decimal format.

3. Packed decimal numbers are manipulated in packed decimal format.

4. By definition, there is no sign in unsigned packed decimal numbers (U format). When you manipulate these numbers, CA-Easytrieve/Plus supplies an F sign.
**Unsigned**

If you specify a numeric field with no decimal positions, CA-Easytrieve/Plus considers that field unsigned (non-quantitative) and the following rules apply:

1. For binary numbers, the magnitude of the number must fit within 31 bits or less. The NUMERIC test is not true for a four-byte binary field with the high-order bit on. The high-order bit contributes to the magnitude of numbers in fields of one-byte to three-byte lengths. For example, a one-byte binary field containing a hexadecimal FF has a numeric value of 255.

2. Both zoned decimal and packed decimal fields follow the same rules. CA-Easytrieve/Plus packs all zoned decimal fields and handles them as packed decimal fields. CA-Easytrieve/Plus uses the actual storage value in the field, but it is your responsibility to maintain a positive sign. An F sign is placed in any unsigned field on the left side of an Assignment statement.

3. An unsigned packed decimal field (U format) is always unsigned. When you manipulate the field, CA-Easytrieve/Plus supplies an F sign.

**Edit Masks**

CA-Easytrieve/Plus edits field data only at the time of printing and according to a specified edit mask pattern. The MASK option of the DEFINE statement specifies the edit mask pattern. Each edit mask pattern is created by using combinations of the following characters:

- 9 — causes a digit to print.
- Z — causes a digit to print (except for leading zeros).
- * — causes an asterisk to replace leading zero digits.
- - — causes a minus sign to print prior to the first non-zero digit of a negative number.
- $ — causes a currency symbol to print prior to the first non-zero digit (the currency symbol is determined by the MONEY option from the Options Table described in Appendix C, “Options Table”).
- x — any character can be printed with the edited data when the digits are represented by 9s.
**Editing Rules**

Although the examples that follow illustrate edit masks and their use, it is important to understand the rules of editing that follow:

1. Each digit of the field must be designated in the mask by 9, Z, *, -, or $.
2. There is no implied relationship between the edit mask and the number of decimal digit attributes of the field.
3. Alphanumeric fields cannot be edited. (The exception is MASK HEX.)

**Leading Zeros**

CA-Easytrieve/Plus provides a number of methods for dealing with leading zeros by printing, suppressing, or replacing them.

**Printing**

When leading zeros are an important part of the number (for example, social security numbers, and part numbers), an edit mask that prints these zeros is essential. Here are several examples of edit masks that print leading zeros.

<table>
<thead>
<tr>
<th>Mask</th>
<th>Field Contents</th>
<th>Printed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>999-99-9999</td>
<td>053707163</td>
<td>053-70-7163</td>
</tr>
<tr>
<td>(99)-9999</td>
<td>006421</td>
<td>(00)-6421</td>
</tr>
</tbody>
</table>

**Suppressing**

In some instances, leading zeros add unnecessary information and can confuse the reader. In these cases, use an edit mask that suppresses leading zeros.

<table>
<thead>
<tr>
<th>Mask</th>
<th>Field Contents</th>
<th>Printed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>$$.$$$9</td>
<td>01234</td>
<td>$1.234</td>
</tr>
<tr>
<td>$$.$$$9</td>
<td>00008</td>
<td>$8</td>
</tr>
<tr>
<td>$$.$$$9.99</td>
<td>0123456</td>
<td>$1.234.56</td>
</tr>
<tr>
<td>ZZZ.ZZ9</td>
<td>000123</td>
<td>123</td>
</tr>
<tr>
<td>---.---9</td>
<td>+001234</td>
<td>1.234</td>
</tr>
<tr>
<td>---.---9</td>
<td>-001234</td>
<td>-1.234</td>
</tr>
</tbody>
</table>
Replacing

In cases where fields need to be protected (for example, check amounts), you can use edit masks that replace leading zeros with other characters.

<table>
<thead>
<tr>
<th>Mask</th>
<th>Field Contents</th>
<th>Printed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>**9</td>
<td>001</td>
<td>**1</td>
</tr>
<tr>
<td>**,**9</td>
<td>01234</td>
<td>*1,234</td>
</tr>
<tr>
<td>**,**9.99</td>
<td>0123456</td>
<td>*1,234.56</td>
</tr>
</tbody>
</table>

Negative Numbers

CA-Easytrieve/Plus prints the characters used as negative number indicators to the right of the last digit of the negative data that you edit. You can use any characters as negative number indicators, although the most typical indicators are the minus sign (-) and the credit indicator (CR). If the number is positive, CA-Easytrieve/Plus inhibits the printing of these character(s).

<table>
<thead>
<tr>
<th>Mask</th>
<th>Field Contents</th>
<th>Printed Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZZZ-</td>
<td>-123</td>
<td>123-</td>
</tr>
<tr>
<td>ZZZ-</td>
<td>+123</td>
<td>123</td>
</tr>
<tr>
<td>ZZZ CR</td>
<td>-123</td>
<td>123 CR</td>
</tr>
<tr>
<td>ZZZ CR</td>
<td>+123</td>
<td>123</td>
</tr>
<tr>
<td>ZZZ IS MINUS</td>
<td>-123</td>
<td>123 IS MINUS</td>
</tr>
</tbody>
</table>

COPY Statement

The COPY statement duplicates the field definitions of a named file or record. Its syntax is:

**Syntax**

```plaintext
COPY {file-name
    }
    
{(database-file-name:)record-name }
```

You can code an unlimited number of COPY statements for any one file. File-name or record-name is the name of a previously defined file or record whose fields you want to duplicate. CA-Easytrieve/Plus duplicates the fields as if they were coded at the place CA-Easytrieve/Plus encounters the COPY statement.

The same rules of field definition apply when using the COPY statement (for example, field names must be unique in a given file or record).
Example

```
FILE PERSNL FB(150 1800)
  NAME 17 20 A HEADING ('EMPLOYEE NAME')
  NAME-LAST NAME 8 A HEADING ('FIRST NAME')
  NAME-FIRST NAME +8 12 A HEADING ('LAST NAME')
FILE SORTWRK FB(150 1800) VIRTUAL
COPY PERSNL
SORT PERSNL TO SORTWRK USING +
  (NAME-LAST NAME-FIRST) NAME MYSORT
JOB INPUT SORTWRK NAME MYPROG
  PRINT REPORT1
  REPORT REPORT1
LINE NAME-FIRST NAME-LAST
```

COPY Rules for Database Files

The following rules apply to the COPY statement when used with database files.

If you code:

```
FILE file-name-1 DLI
  field-name ...
  RECORD ...
  field-name ...
FILE file-name-2 DLI
COPY file-name-1
```

the entire DLI file structure, including the PCB fields, are copied from file-name-1
to file-name-2.

If you code:

```
FILE file-name-1 DLI
  field-name ...
  RECORD ...
  field-name ...
  RECORD ...
FILE file-name-2 (sequential file)
COPY file-name-1
```

the DLI fields (no PCB fields are copied) are copied and adjusted to reflect their
locations as they are in the concatenated DLI record. (Because the record
structure is not copied to a sequential file, the field names must be unique. If this
is not the case, a duplicate name error occurs.)

COPY RECORD-NAME copies the fields from only the specified RECORD.
**Examples**

```plaintext
FILE DBASE IDMS(DEMOSS03)
RECORD CUSTOMER 104 KEY(CUST-NO)
  CUST-NO     1 10 A
  CUST-NAME   11 20 A
RECORD SALES  28
  SLS-CUST-NO 1 10 A
FILE DDBASE FB(28 280)
COPY SALES       (fields from RECORD SALES copied)
JOB INPUT (DBASE) NAME MYPROG
RETRIEVE DBASE +
  SELECT (CUSTOMER AREA 'CUSTOMER-REGION' +
            SALES ID 'SA' SET 'CUSTOMER-SALES')
  IF PATH-ID EQ 'SA'
    MOVE LIKE SALES TO DDBASE
  PUT DDBASE
ELSE
  GO TO JOB
END-IF

FILE DLIFILE DLI(DI21PART 1)
RECORD PARTROOT 50 KEY(PARTKEY 1 17)
  PARTKEY      1 17 A
  PART-NUMBER  1 17 A
  PART-DESC   27 24 A
FILE SEQFILE FB(50 500)
COPY DLIFILE         (record description from FILE DLIFILE copied)
JOB INPUT (DLIFILE) NAME MYPROG
RETRIEVE DLIFILE +
  SELECT (PARTROOT WHILE (PART-DESC ALPHABETIC))
  MOVE LIKE DLIFILE TO SEQFILE
  PUT SEQFILE
```

You can specify record-name(file-name) or file-name:record-name on the COPY statement.

**Note:** You cannot copy a database file to a Host Disk file.

**Data Reference**

Every data reference in your program must be unique. You can provide this uniqueness in one of two ways:

- Unique name
- Qualification.

**Unique Name**

A name is unique if no other file or work field has that name. For example, GROSS-PAY is unique if it appears as field-name-1 in only one DEFINE statement (and is never copied to another file).
Qualification

Qualification occurs when you prefix the optional qualifier file-name: to a field-name. CA-Easytrieve/Plus requires the use of the qualifier whenever the field-name alone cannot uniquely identify the data reference. The qualifier for file fields is the associated file name and/or record name. For working storage fields, the qualifier is the keyword WORK. Fields from non-synchronized, automatic input files do not require the optional file-name: qualifier.

Indexing

Indexing is data reference that results when CA-Easytrieve/Plus derives a displacement value to correspond to a particular occurrence in a field name defined with the OCCURS option. The formula for deriving the index value is: the number of the desired occurrence minus one, multiplied by the length of the occurring field element. For example, if an occurring field is defined as:

```
DEFINE MONTHWORD MONTH-TABLE 9 A + 
   OCCURS 12 INDEX MONTH-INDEX
```

MONTH-INDEX for the third occurrence is derived as follows:

```
MONTH-INDEX = (3 - 1) * 9
MONTH-INDEX = 18
```

Subscripts are an alternate method available where CA-Easytrieve/Plus performs the index computation.

System-Defined Fields

CA-Easytrieve/Plus automatically provides three categories of system defined fields:

- General
- File related
- Report related.

General Fields

SYSDATE is an eight-byte alphabetic field that contains the system date at the start of CA-Easytrieve/Plus execution. The DATE option of the Options Table (Appendix C), determines the format of the date. A slash (/) separates the month, day, and year components of the date, such as MM/DD/YY.
SYSDATE-LONG is a 10-byte alphabetic field that contains the system date, including the century, at the start of CA-Easytrieve/Plus execution. The DATE option of the Options Table (Appendix C), determines the format of the date. A slash (/) separates the month, day, and year components of the date, such as MM/DD/CCYY.

SYSTIME is an eight-byte alphabetic field that contains the system time at the start of CA-Easytrieve/Plus execution. A period (.) separates the data into hours, minutes, and seconds, such as HH.MM.SS.

PARM-REGISTER is a four-byte binary field which contains the contents of register 1 upon entry to CA-Easytrieve/Plus. Register 1 contains the address of a passed parameter list. This address is accessed by the subprogram EZTPX01 when passing EXEC statement PARM parameters.

RETURN-CODE is a four-byte binary field whose contents are returned to the operating system in register 15 when CA-Easytrieve/Plus terminates. RETURN-CODE is initialized to zero, but you can set it to any value.

KANJI-DAY is an 18-byte field that contains the system date at the start of CA-Easytrieve/Plus execution. The DATE option of the CA-Easytrieve/Plus Options module determines the format of the date. The DBCS Options module defines the KANJI characters that are merged with the year, month, and day components of the date. The DBCS code system of the date is the default Processing DBCS code system as established in the DBCS Options module. For more information, see the CA-Easytrieve/Plus Installation Guide.

KANJI-DAY is a 16-byte Kanji field that contains the system time at the start of CA-Easytrieve/Plus execution. This field is the same as the CA-Easytrieve/Plus system defined field called SYSTIME, except that the SYSTIME EBCDIC value has been converted into DBCS. The DBCS code system of the date is the default Processing DBCS code system as established in the DBCS Options module. For more information, see the CA-Easytrieve/Plus Installation Guide.

**File Fields**

**Report Fields**

The report related fields LEVEL and TALLY are described in Chapter 11, “Report Processing,” under the subject “Control Reports.”

LINE-COUNT is a field that contains the number of lines printed on the page.

PAGE-COUNT is a field that contains the number of pages printed.
Introduction

This chapter discusses CA-Easytrieve/Plus processing activities. These activities resemble the steps of a JOB, but they are not constrained by JOB Control Language (JCL) and associated operating system overhead. CA-Easytrieve/Plus executes activities one at a time, in top-to-bottom order.

The two processing activities of CA-Easytrieve/Plus are JOB and SORT. You can code JOB and SORT activities as often as necessary within any CA-Easytrieve/Plus program.

JOB Activities

JOB activities examine and manipulate data, write information to files, and initiate printed reports.

SORT Activities

SORT activities create a sequenced file that contains all or part of the records from another file.

JOB Statement

The JOB statement defines and initiates processing activities. Within these activities, statements can specify various processing tasks:

- Retrieval of input files
- Examination and manipulation of data
- Initiation of printed reports
- Production of output files.
The JOB statement also identifies the name of the automatic input file (which can be any file or database that can be processed sequentially). The following example illustrates the position of JOB activities within a CA-Easytrieve/Plus program:

```
Environment
...
Library
...
{JOB ...}...
{...}
JOB   <--------{ job procedures
and/or {...
SORT   { reports
Activities {...
...
```

### Syntax

Code the parameters of the JOB statement in any order. As shown, code multiple subparameters within parentheses. The syntax is:

```
JOB  +
[      {                                       }
[INPUT {{(file-name [KEY (field-name-1...)]...)}]  +
[      {   NULL                                }
[      {   SQL                                 }
[START proc-name-1]   +
[FINISH proc-name-2]  +
[NAME job-name]       +
[ENVIRONMENT {NONE }] +
[    {COBOL}]
[CHECKPOINT (literal [USING (field-name-2...)])) +
[RESTART proc-name-3]
```

### Parameters

**[INPUT]**

The optional INPUT parameter identifies the automatic input to the activity.

When you do not specify INPUT, CA-Easytrieve/Plus automatically provides an input file. If a SORT activity immediately precedes the current JOB activity, the default input is the output file from that SORT activity. Otherwise, the default input is the first file named in the library section.
{{(file-name)}}

File-name identifies the automatic input file(s). File-name identifies any file defined in the library section of the program eligible for sequential input processing.

{NULL}

Code NULL as the file-name to inhibit automatic input. Use this when no input is required or when input is retrieved by statements within the activity. When using NULL, a STOP statement must be executed within the JOB activity, otherwise the activity executes forever.

{SQL}

SQL is used to identify input coming from an SQL table. The selection criteria for the input is specified on the SQL SELECT statement that must immediately follow the JOB statement. For more details on SQL statements, see the appropriate database interface manual CA-Easytrieve/Plus SQL Interface Option Guide or ORACLE).

{{[KEY(field-name-1 ...)]}}

Code the KEY field-name subparameter for each file-name of a synchronized file input process. Keys can be any fields from the associated file. The only exceptions are varying length fields, which cannot be used as keys. For more detailed information on automatic input, including synchronized file processing, see Chapter 12, “File Processing.”

[START proc-name-1]

The optional START parameter identifies a procedure to be executed during the initiation of the JOB.

CA-Easytrieve/Plus performs the procedure coded in proc-name-1 before it retrieves the first automatic input record. A typical START procedure sets working storage fields to an initial value or positions an indexed or keyed sequentially processed file to a specific record. You cannot reference fields in automatic input files since no records have been retrieved at this stage of processing.

[FINISH proc-name-2]

The optional FINISH parameter identifies a procedure to be executed during the normal termination of the JOB. After CA-Easytrieve/Plus processes the last automatic input record, it performs the proc-name-2 procedure. A typical FINISH procedure displays control information accumulated during the activity.
The optional NAME parameter names the JOB activity. Job-name can be up to 40 characters long, can contain any character other than a delimiter, and begins with A-Z or 0-9; it cannot consist of all numeric characters. This parameter is used only for documentation purposes. If the LINK parameter of the PARM statement is in effect, job-name is generated into the object module as a character constant to assist in identifying the code for the job. If present, job-name is printed by the CA-Easytrieve/Plus debugging aid. If you use the LINK parameter, the first 8 characters (first 7 characters for VSE) of all job names must be unique.

ENVIRONMENT (OS/390 only) instructs CA-Easytrieve/Plus to establish the proper execution environment prior to calling any COBOL subprograms. The environment is established prior to each JOB activity that contains a CALL statement and is terminated after the activity for which it was established. When used on the JOB statement, it establishes the default (NONE or COBOL) used for a single JOB activity.

When this parameter is absent, the default for ENVIRONMENT depends on how the ENVIRON system option was set at installation and whether or not the ENVIRONMENT parameter of the PARM statement was specified. Using the subparameter NONE overrides an existing default of COBOL and using COBOL overrides a default of NONE. See Chapter 14, “Subprograms,” for more information on the ENVIRONMENT parameter.

CHECKPOINT and RESTART are IMS-only parameters for use with the Extended Checkpoint/Restart Facility. For details see the CA-Easytrieve/Plus IMS/DLI Interface Option manual.

Job Flow

Because of a JOB statement's particular relationship with CA-Easytrieve/Plus file processing, no examples of JOB statement usage have been provided in this chapter. Chapter 12, “File Processing,” discusses file processing techniques in detail.
It is important, however, to fully understand the flow of control within a JOB activity. The following exhibits illustrate the physical relationship between statements in a job activity, with the implied statements attributed to JOB.

<table>
<thead>
<tr>
<th>reset working storage</th>
<th>Step 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[PERFORM restart-proc]</td>
<td></td>
</tr>
<tr>
<td>[PERFORM start-proc]</td>
<td></td>
</tr>
<tr>
<td>JOB ... retrieve automatic input</td>
<td>Logic generated by JOB</td>
</tr>
<tr>
<td>IF EOF ...</td>
<td></td>
</tr>
<tr>
<td>reset working storage</td>
<td></td>
</tr>
<tr>
<td>[PERFORM finish-proc]</td>
<td></td>
</tr>
<tr>
<td>wrap-up REPORTs</td>
<td></td>
</tr>
<tr>
<td>go to the next JOB/SORT activity</td>
<td></td>
</tr>
<tr>
<td>END-IF</td>
<td></td>
</tr>
<tr>
<td>IF ...</td>
<td>Step 2</td>
</tr>
<tr>
<td>...</td>
<td>Data examination and</td>
</tr>
<tr>
<td>...</td>
<td>manipulation statements</td>
</tr>
<tr>
<td>END-IF</td>
<td></td>
</tr>
<tr>
<td>reset working storage</td>
<td>Step 3</td>
</tr>
<tr>
<td>GO TO JOB</td>
<td>Processing of the activity</td>
</tr>
<tr>
<td></td>
<td>is reinstated to get the</td>
</tr>
<tr>
<td></td>
<td>next automatic input record</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>proc-name. PROC</td>
<td>Step 4</td>
</tr>
<tr>
<td>...</td>
<td>Optional programmer-written</td>
</tr>
<tr>
<td>END-PROC</td>
<td>procedures and reports are</td>
</tr>
<tr>
<td>...</td>
<td>placed at the end of the</td>
</tr>
<tr>
<td>REPORT</td>
<td>activity</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>JOB/SORT</td>
<td></td>
</tr>
</tbody>
</table>
The following flow diagram illustrates the processing of a CA-Easytrieve/Plus JOB activity:
CA-Easytrieve/Plus processes input records one at a time. You can use any valid combination of CA-Easytrieve/Plus statements to examine and manipulate the input record. CA-Easytrieve/Plus repeats the processing activity until the input is exhausted or until you issue a STOP statement. RESET is a keyword on the DEFINE statement for resetting working storage fields.

**SORT Statement**

Using the SORT statement, CA-Easytrieve/Plus can sort any file that can be processed sequentially. The following example 5.5 shows the position of a SORT activity within a program:

```
Environment
...
Library
...
Activities
...
JOB
and/or {SORT ...
SORT <----------{ sort procedure
...
...
```

Your installation's sort program performs the actual sort process. CA-Easytrieve/Plus utilizes conventional sort interface techniques by invoking the sort program's E15 (input) and E35 (output) exits. For detailed information on the available options for sort program utilization, refer to your installation's sort program manual.

**Syntax**

Code the required parameters (file names and keys) immediately after the SORT parameter. Code the optional parameters in any order. The complete syntax is:

```
SORT file-name-1 TO file-name-2 +
   [D]
   USING (field-name-1 [IBM-sort-options] ... ) +
   [JEF-sort-options]
   [SIZE literal-1] +
   [WORK literal-2] +
   [BEFORE proc-name] +
   [NAME sort-name]
```
**Parameters**

**file-name-1**

File-name-1 is the name of the input file for the sort activity. File-name-1 must reference a FILE statement that defines a SAM, VSAM, ISAM, or VFM file. The record length of file-name-1 controls the length of records to be sorted, except when both files are fixed length. When this occurs, the length is from file-name-1 or file-name-2, whichever is shorter.

**TO file-name-2**

The TO parameter designates the name of the output file of the sort activity. File-name-2 must reference a FILE statement that defines a SAM, VSAM, Host Disk, or VFM file. If file-name-2 is the same as file-name-1, the SORTed output is written over the input file.

**USING (field-name-1 [D]...)**

The USING parameter identifies data fields in file-name-1 used as sort keys. You can code any number of fields up to your installation’s sort input program’s limit. Field-name-1 can be any field (less than 256 bytes long) in the sort input file. (The only exceptions are variable length fields, which cannot be used as keys.) Coding a D immediately after any field-name-1 causes that field to be sorted in descending order. If the D is not coded after field-name-1, by default the field is sorted in ascending order.

If no DBCS fields are identified as sort keys, the CA-Easytrieve/Plus system invokes the system sort that the CA-Easytrieve/Plus options module identifies. Refer to the *CA-Easytrieve/Plus Installation Guide* for more information on the CA-Easytrieve/Plus options module.

The rules for supporting DBCS and MIXED fields as sort keys are:

- All MIXED fields are defined to the sort as alphanumeric type fields.
- File-name-1 and file-name-2 must belong to the same DBCS code system. If the system for both files is not the same, the SORT statement generates an error.
- CA-Easytrieve/Plus supports only the IBM and FACOM (JEF) Kanji sorts. Therefore, if you code a DBCS field as a sort key and the DBCS code system of file-name-1 is not IBM or JEF, CA-Easytrieve/Plus uses the standard system sort and processes the DBCS fields as alphanumeric type fields.
- If the DBCS code system of file-name-1 is IBM or JEF but the applicable Kanji sort is not identified in the DBCS options module, CA-Easytrieve/Plus passes any DBCS fields coded as sort keys to the system sort as alphanumeric field types.
CA-Easytrieve/Plus supports the respective sort options when support for the IBM or FACOM sort is defined in the DBCS options module. You can code them for only DBCS fields. If you do not select a sort-option for a DBCS field, CA-Easytrieve/Plus defines the field to the sort as an alphanumeric field. If no DBCS fields are using the IBM or JEF sort-options, CA-Easytrieve/Plus uses the normal system sort. That is, CA-Easytrieve/Plus uses the applicable Kanji sort only when required.

\[
\text{USING (field-name-1 [ IBM-sort-options ] ... )}
\]

An IBM-sort-option defines the Kanji sequence technique to be applied to the DBCS field. For more specific information regarding the different techniques, consult the IBM Program Description and Operation Manual for the Kanji/Chinese Sort/Merge program product.

```
IBM-sort-options
field-name-1  +

[ BUSHU [D] ]
[ SOKAKU [D] ]

[ KOKUGO (field-name-2 [ SMAP field-name-3 ] [D] [B] [I])
  [ DMAP field-name-4 ] ]

[ DENWA (field-name-2 [ SMAP field-name-3 ] [D] [B] [I])
  [ DMAP field-name-4 ] ]
```

[ BUSHU [D] ]

This sort option invokes the Basic Radical Stroke-Count sequence for field-name-1. Coding a D immediately after BUSHU causes the field to be sorted in descending Radical Stroke-Count order. If you do not code the D, by default the field is sorted in ascending order.

[ SOKAKU [D] ]

This sort option invokes the Basic Total Stroke-Count sequence for field-name-1. Coding a D immediately after SOKAKU causes the field to be sorted in descending Total Stroke-Count order. If you do not code the D, by default the field is sorted in ascending order.
This sort option invokes one of two different Kanji sorting techniques depending upon whether or not you choose the SMAP or DMAP keyword. Both techniques require you to specify a phonetic syllabary field. Field-name-2 identifies a field defined in file-name-1. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is alphanumeric, the field must contain the phonetic reading of field-name-1 in Katakana. If field-name-2 is DBCS, it must contain the phonetic reading of field-name-1 represented by IBM DBCS Hiragana or Katakana.

If you do not specify SMAP or DMAP then CA-Easytrieve/Plus applies the Japanese Dictionary sequencing technique to field-name-1. This technique sequences the records based on the phonetic reading of the WHOLE Kanji field.

If you select SMAP or DMAP, then CA-Easytrieve/Plus applies Kanji Index type processing for field-name-1. This technique orders the records based on the phonetic reading of each Kanji character. To do this you must supply a reading map so the sort can relate phonetic syllabary characters to the appropriate Kanji characters. The reading map must be in the form of an alphanumeric field specified as field-name-3 or field-name-4. Field-name-3 specifies a field defined in file-name-1. The field contains a single-map. Field-name-4 specifies a field defined in file-name-1. The field contains a double-map. For more information on the definition and creation of these bit map fields, see the IBM Sort/Merge Program - Kanji/Chinese Manual.

Coding a D causes the field to be sorted in descending order. If the D is not coded, by default the field is sorted in ascending order. Coding a B causes CA-Easytrieve/Plus to apply the BUSHU (or Radical Stroke-Count) technique as a sub-sequence for those records that have the same phonetic reading. If you do not code B, by default the field is sub-sequenced using the SOKAKU (or Total Stroke-count). Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.
The DENWA sort option invokes one of two different forms of the Japanese Telephone Directory sequencing technique. The form of sequencing technique that you use depends upon whether or not you specify the SMAP or DMAP keyword. The Telephone Directory method requires you to specify a phonetic syllabary field. Field-name-2 identifies a field defined in file-name-1. It must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS data it must contain the phonetic reading of field-name-1 represented by IBM DBCS Hiragana or Katakana.

If you do not specify SMAP or DMAP, CA-Easytrieve/Plus applies the Simple form of the Japanese Telephone Directory order to field-name-1. This technique sequences the records based on the representative reading of the first Kanji character in the field. The representative reading of the first Kanji character is obtained from a special representative reading table based on the Kanji character itself and the voiceless sound of the first phonetic syllabary character in field-name-2.

If you specify SMAP or DMAP, CA-Easytrieve/Plus applies the All-Digit Japanese Telephone Directory sequence. This technique orders the records based on the representative reading of all the Kanji characters in the field. Sequencing is determined by applying representative reading to each Kanji character. For this, CA-Easytrieve/Plus requires a reading map field in the form of an alphanumeric field specified as field-name-3 or field-name-4. The reading map field identifies a bit map that allows the sort to relate phonetic syllabary characters to the appropriate Kanji characters. Field-name-3 specifies a field defined in file-name-1. The field contains a single-map. Field-name-4 specifies a field defined in file-name-1. The field contains a double-map. For more information on the definition and creation of these bit map fields, see the IBM Sort/Merge Program - Kanji/Chinese Manual.

Coding a D causes the field to be sorted in descending order. If you do not code D, by default the field is sorted in ascending order. Coding a B causes CA-Easytrieve/Plus to apply the BUSHU (or Radical Stroke-Count) technique as a sub-sequence for those records that have the same phonetic reading. If you do not code B, by default the field is sub-sequenced using the SOKAKU (or Total Stroke-count). Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.
SORT Statement

USING (field-name [JEF-sort-options] ...)

A JEF-sort-option defines the Kanji sequence technique to be applied to the DBCS field. For more specific information regarding the different techniques, consult the FACOM Sort/Merge Program Description Manual.

Each of these keywords identify a unique form of Kanji sequencing.

- **BUSHU**—ordered by the Radical Stroke count of the Kanji character.
- **SOKAKU**—ordered by the Total Stroke count of the Kanji character.
- **ON**—ordered by the 'ON' or Chinese pronunciation of the Kanji Character.
- **KUN**—ordered by the 'KUN' or Japanese pronunciation of the Kanji Character.

Specify one or more of these keywords to indicate the ordering technique that CA-Easytrieve/Plus is to apply to field-name-1. If you code more than one keyword, they must be coded in major to minor order. CA-Easytrieve/Plus uses each minor ordering technique to order those records that are the same after applying the preceding ordering technique. For example if you coded "USING (FIELD-ONE (SOKAKU ON KUN))" FIELD-ONE would be ordered using the Total Stroke Count of the Kanji. Those records having the same count would then be ordered by their ON reading. If there are records still the same, they would be ordered by their KUN or Japanese reading.

Each of the keywords can only be coded once after field-name-1.
[KOKUGO (field-name-2 [D] [I] )]

The KOKUGO sort option invokes the Japanese Dictionary sequencing technique for field-name-1. This technique sequences the records based on the phonetic reading of the WHOLE Kanji field. This technique requires you to specify a phonetic syllabary field. Field-name-2 identifies a field defined in file-name-1. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS format data, it must contain the phonetic reading of field-name-1 represented by JEF DBCS Hiragana or Katakana.

Coding a D causes the field to be sorted in descending order. If you do not code the D, by default the field is sorted in ascending order. Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.

[DENWA (field-name-2 [D] [I] [DLM] )]

The DENWA sort option invokes the Japanese Telephone Directory sequencing technique for field-name-1. This technique requires you to specify a phonetic syllabary field. Field-name-2 identifies a field defined in file-name-1. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS format data, it must contain the phonetic reading of field-name-1 represented by JEF DBCS Hiragana or Katakana.

Coding a D causes the field to be sorted in descending order. If you do not code the D, by default the field is sorted in ascending order. Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence. You can code the DLM subparameter if both field-name-1 and field-name-2 contain space characters that act as delimiters thus permitting the JEF Kanji SORT to associate the phonetic characters in field-name-2 with the Kanji characters in field-name-1. If no such delimiters exist, you should not code DLM. When you do not code the DLM subparameter, CA-Easytrieve/Plus assumes the NODLM sort option.

[SIZE literal-1]

Since CA-Easytrieve/Plus knows the number of records in files created by previous activities, it automatically supplies that information to the sort program. If the file was not created by a previous activity, you can enhance sort efficiency by supplying the approximate number of records as literal-1 on the optional SIZE parameter. The default file size is 1000 records.
The literal-2 value of the optional WORK parameter establishes the number of work data sets used by the sort program. This parameter overrides the NUMWORK options table entry.

For VSE, literal-2 must be one of the following:
- DA to indicate a multi-extent direct access type data set is used.
- A value from one to eight to specify the number of sequential disk type data sets used.

For OS/390, literal-2 must be one of the following:
- A zero to indicate that DD statements are supplied for work data sets.
- A value from 1 to 31 to indicate the number of work data sets that the sort program dynamically allocates.

The optional BEFORE parameter identifies a procedure that pre-screens, modifies, and selects input records for the sort. Proc-name indicates the PROC statement that identifies the sort procedure. If you do not specify BEFORE proc-name, CA-Easytrieve/Plus sorts all records in file-name-1 and outputs them onto file-name-2. See the SELECT statement description later in this chapter for selection of records to be sorted.

The optional NAME parameter names the SORT activity. Sort-name can be up to 40 characters long, can contain any character other than a delimiter, and begin with A-Z or 0-9; it cannot consist of all numeric characters. This parameter is used only for documentation purposes.

### Sorting Files

The following output file contains all of the records of the input file sorted into ascending sequence by the values of fields REGION and BRANCH.

```
FILE PERSNL FB(150 1800) %PERSNL
FILE SORTWRK FB(150 1800) VIRTUAL
COPY PERSNL
SORT PERSNL TO SORTWRK USING + (REGION, BRANCH) NAME MYSORT
JOB INPUT SORTWRK NAME MYPROG
* PRINT REPORT1
REPORT REPORT1
LINE REGION BRANCH NAME
```
Sort Procedures

CA-Easytrieve/Plus normally sorts all input records and outputs them onto the TO file. The output file usually has the same format and length as the input file. However, sometimes it is desirable to sort only certain records and/or to modify the contents. To do this, you must write a sort procedure which must immediately follow the SORT statement.

You can code any valid CA-Easytrieve/Plus statement in a sort procedure; however, you cannot code statements that generate input/output. Invalid statements are:

DISPLAY
DLI
GET
IDMS
POINT
PRINT
PUT
READ
SQL
WRITE

Note: For debugging purposes, you can DISPLAY to the system output device (SYSPRINT/SYSLST).

The only valid field references within a sort procedure are:

- Any field of the input file (file-name-I)
- Any working storage field
- System defined fields, such as SYSDATE and RECORD-LENGTH.
**SORT Statement**

---

**Sort Flow**

The following example illustrates the flow of a SORT activity. The basic format of the SORT statement, as shown earlier in this chapter, is:

```
SORT filea TO fileb USING fld1, ... [BEFORE proc-name]
```

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Retrieve first record from input file (filea)</td>
</tr>
<tr>
<td>2</td>
<td>DO WHILE NOT EOF filea</td>
</tr>
<tr>
<td>3</td>
<td>IF BEFORE was specified</td>
</tr>
<tr>
<td>4</td>
<td>IF RESET working storage fields are specified</td>
</tr>
<tr>
<td>5</td>
<td>Reset all RESET working storage fields</td>
</tr>
<tr>
<td>6</td>
<td>END-IF</td>
</tr>
<tr>
<td>7</td>
<td>PERFORM proc-name</td>
</tr>
<tr>
<td>8</td>
<td>IF SELECT statement was executed</td>
</tr>
<tr>
<td>9</td>
<td>pass record to SORT</td>
</tr>
<tr>
<td>10</td>
<td>END-IF</td>
</tr>
<tr>
<td>11</td>
<td>ELSE</td>
</tr>
<tr>
<td>12</td>
<td>pass record to SORT</td>
</tr>
<tr>
<td>13</td>
<td>END-IF</td>
</tr>
<tr>
<td>14</td>
<td>Retrieve next record from input file (filea)</td>
</tr>
<tr>
<td>15</td>
<td>END-DO</td>
</tr>
<tr>
<td>16</td>
<td>Perform SORT process (USING fld1, ...)</td>
</tr>
<tr>
<td>17</td>
<td>DO WHILE sorted records exist</td>
</tr>
<tr>
<td>18</td>
<td>Write sorted record to output file (fileb)</td>
</tr>
<tr>
<td>19</td>
<td>END-DO</td>
</tr>
<tr>
<td>20</td>
<td>proc-name. PROC</td>
</tr>
<tr>
<td>21</td>
<td>...</td>
</tr>
<tr>
<td>22</td>
<td>SELECT</td>
</tr>
<tr>
<td>23</td>
<td>...</td>
</tr>
<tr>
<td>24</td>
<td>END-PROC</td>
</tr>
<tr>
<td>25</td>
<td>JOB/SORT</td>
</tr>
</tbody>
</table>

---

**Sort Flow**

Logic generated by the SORT statement

IF BEFORE requested

Re-initialize RESET fields

Perform the user's proc

SELECT executed?

pass record to SORT

No BEFORE proc.

pass all to SORT

Get next record from input file

Actually SORT the records

Write sorted records to output file

Optional user-written procedure is placed after the SORT
This diagram shows the flow of a SORT activity using a flow diagram.

begin activity

retrieve input

end of input? NO

YES

execute SORT

write records to output file

terminate activity

BEFORE coded? NO

YES

execute RESET

perform BEFORE procedure

SELECT executed? YES

NO

pass record to SORT
SELECT Statement

CA-Easytrieve/Plus supplies input records to your sort procedure one at a time. If a BEFORE procedure is used, the SELECT statement must be executed for each record that you want to sort. The syntax of the SELECT statement is:

Syntax

```
SELECT
```

SELECT only sets a switch to cause record selection at a later time. If you SELECT a record twice, it only appears once on the SORTed file.

Sorting a Selected Portion of a File

The following example of a sort activity shows an output file that contains only a reordered subset of the input file. The output file contains only those records for which the SELECT statement is executed.

```
FILE PERSNL FB(150 1800)
%PERSNL
FILE SORTWRK FB(150 1800) VIRTUAL
COPY PERSNL
SORT PERSNL TO SORTWRK USING +
   (REGION, BRANCH, DEPT, +
    NAME-LAST, NAME-FIRST) +
   NAME MYSORT BEFORE SCREENER
* SCREENER. PROC
   IF MARITAL-STAT = 'S' AND SEX = 1
      SELECT
   END-IF
END-PROC
* * JOB INPUT SORTWRK NAME MYPROG
   PRINT REPORT1
* REPORT REPORT1
   LINE REGION BRANCH DEPT NAME-LAST NAME-FIRST
```
Introduction

Chapter 7 discusses the ways to move or manipulate data within your program using the Assignment, MOVE, and MOVE LIKE statements.

The Assignment statement establishes the value of a field as a result of simple data movements, an arithmetic expression, or logical bit manipulation.

An arithmetic expression produces a numeric value by adding, subtracting, multiplying, or dividing numeric quantities.

The MOVE statement transfers character strings from one storage location to another.

The MOVE LIKE statement copies fields with identical field names from one file to another.

Arithmetic Expressions

To fully understand how the Assignment statement establishes the value of a field as a result of an arithmetic expression, you need to know how arithmetic expressions work within CA-Easytrieve/Plus.

An arithmetic expression enables two or more numeric quantities to be combined to produce a single value. Arithmetic expressions can be used in Assignment statements and in field relational conditions.

Syntax

```
{[ ]} {*} {[ ]} {*[ ]} {[ ]} {[ ]} {*[ ]} {*[ ]} {*[ ]} {*[ ]} ... 
{[+] field-name-1} {[/]} {[+] field-name-2} 
{[-] literal-1} {[*]} {[-] literal-2} 
{[ ]} {[-] {[ ]}} 
```
Arithmetic Expressions

Operation

The CA-Easytrieve/Plus arithmetic operators are:
* multiplication
/ division
+ addition
- subtraction

All fields and literals in an arithmetic expression must be numeric.

CA-Easytrieve/Plus follows the standard mathematical order of operations when computing arithmetic expressions: multiplication and division are performed before addition and subtraction, in order from left to right.

The following exhibit illustrates how CA-Easytrieve/Plus evaluates arithmetic expressions:

\[
\begin{align*}
11 &+ 5 \times 8 - 48 / 16 + 4 & \text{Step 1} \\
11 &+ 40 - 48 / 16 + 4 & \text{Step 2} \\
11 &+ 40 - 3 + 4 & \text{Step 3} \\
51 &- 3 + 4 & \text{Step 4} \\
18 &+ 4 & \text{Step 5} \\
52 & & \\
\end{align*}
\]

Parentheses

You can use parentheses to override the normal order of evaluation. Any level of parenthesis nesting is permitted. CA-Easytrieve/Plus evaluates expressions within parentheses first. Evaluation proceeds from the innermost parenthesis level to the outermost.
The following exhibit illustrates how CA-Easytrieve/Plus evaluates parentheses found within arithmetic expressions:

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

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

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

\[
11 + 5 \times 1.5 \quad \text{Step 4}
\]

\[
11 + 7.5 \quad \text{Step 5}
\]

\[
18.5
\]

**Evaluations**

When evaluating an arithmetic expression, CA-Easytrieve/Plus maintains at most 30 decimal digits for each operation.

During the calculation of:

\[
\text{field-name-1} \quad \{=\} \quad \text{value-1} \quad \{/\} \quad \text{value-2}
\]

the length and number of decimal places maintained during the calculation (intermediate results) is determined for each operation according to the rules shown in the following table.

<table>
<thead>
<tr>
<th>If Operation is:</th>
<th>Number of Decimal Places Equals:</th>
</tr>
</thead>
</table>
| Addition or Subtraction | Decimal places:
                        | The larger of the number of decimal places in value-1 or value-2.                               |
                        | Length:
                        | The larger of the number of integer places in value-1 or value-2, plus the number of decimal places in result plus 1. |
| Multiplication        | Decimal places:
                        | The sum of the number of decimal places in value-1 and value-2.                                 |
                        | Length:
<pre><code>                    | The sum of the length of value-1 and value-2.                                                    |
</code></pre>
<table>
<thead>
<tr>
<th>If Operation is:</th>
<th>Number of Decimal Places Equals:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Division</td>
<td>Decimal places</td>
</tr>
<tr>
<td></td>
<td>The larger of:</td>
</tr>
<tr>
<td></td>
<td>a) The number of decimal places in value-1 minus the number of decimal places in value-2.</td>
</tr>
<tr>
<td></td>
<td>b) The number of decimal places in field-name-1 plus one.</td>
</tr>
<tr>
<td></td>
<td>c) 4 decimal places.</td>
</tr>
<tr>
<td>Length:</td>
<td>The number of integer places in value-1 plus the number of decimal places in the result.</td>
</tr>
</tbody>
</table>

If the length of the intermediate result has more than 30 digits, CA-Easytrieve/Plus must truncate the excess digits. For addition, subtraction, and division, the excess digits are always truncated from the left side of the result.

For multiplication, however, CA-Easytrieve/Plus first attempts to do the truncation on the right side of the result. The minimum number of decimal places to be maintained in the result is the number of decimal places in field-name-1 plus one. If the number of decimal places in the result is less than or equal to this minimum, no digits are truncated from the right side of the result. Otherwise, the number of digits truncated from the right is the smaller of a) the number of excess digits or b) the difference between the number of decimal places in the result and the minimum.

When truncation occurs on the right, both the length and number of decimal places in the result are reduced by the number of digits truncated. If there are still excess digits after right truncation, these excess digits are truncated from the left.
For example, assume that value-1 and value-2 both have a length of 18 digits and both have 4 decimal places. Then, according to the above table, the result is a length of 36 digits and 8 decimal places. In this case, the number of excess digits is 6. Then, for various values of the number of decimal places in field-name-1, the result is truncated as shown in the following table.

<table>
<thead>
<tr>
<th>Decimal places in field-name-1</th>
<th>Digits truncated on right</th>
<th>Digits truncated on left</th>
<th>Decimal places in result</th>
</tr>
</thead>
<tbody>
<tr>
<td>fewer than 2</td>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>more than 6</td>
<td>0</td>
<td>6</td>
<td>8</td>
</tr>
</tbody>
</table>

**Assignment Statement**

The Assignment statement establishes a value in a field. The value can be a copy of the data in another field or literal, or it can be the result of an arithmetic or logical expression evaluation. The two formats of the Assignment statement are:

**Format 1**

```plaintext
[ROUNDED] { = } {field-name-2
field-name-1 [INTEGER] [TRUNCATED] {EQ} {literal-1
[ ] { } {arithmetic expression}
```

**Format 2**

```plaintext
{ } {AND} {field-name-2 {OR } {field-name-3
{EQ} {XOR} {literal-2 }
```

Assignment Statement
Format 1 (Normal Assignment)

Format 1 sets the value of field-name-1 equal to the value of field-name-2, literal-1, or the arithmetic expression. The rules of the statement are shown in the following table.

<table>
<thead>
<tr>
<th>Specification Rule</th>
<th>Resulting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If field-name-1 is alphanumeric and the right-hand side is:</td>
<td></td>
</tr>
<tr>
<td>field-name-2</td>
<td>The resulting value of field-name-1 is padded on the right with spaces or truncated as necessary.</td>
</tr>
<tr>
<td>(alphabetic)</td>
<td></td>
</tr>
<tr>
<td>(numeric)</td>
<td>The resulting value of field-name-1 is the non-quantitative zoned decimal equivalent of field-name-2 with padding or truncation on the left as necessary.</td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>literal-1</td>
<td>The resulting value of field-name-1 is padded on the right with spaces as necessary.</td>
</tr>
<tr>
<td>(can be alphanumeric or hexadecimal)</td>
<td></td>
</tr>
<tr>
<td>When field-name-1 is numeric, use:</td>
<td></td>
</tr>
<tr>
<td>field-name-2</td>
<td>The result is padded on the left with zeros to fit the description of field-name-1. If the value of the assignment is too large to be stored into field-name-1, it is truncated as follows:</td>
</tr>
<tr>
<td>(must be numeric)</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>literal-1</td>
<td>- For Binary numbers (numbers expressed in Two's Complement form), the sign and high order bits are truncated from the left as necessary, and the remaining left-most bit becomes the new sign.</td>
</tr>
<tr>
<td>(must be numeric)</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td></td>
</tr>
<tr>
<td>arithmetic expression</td>
<td>- For Zoned Decimal, Packed Decimal, and Unsigned Packed Decimal numbers (numbers expressed in Sign-Magnitude form), the high order digits are truncated from the left as necessary.</td>
</tr>
<tr>
<td>(all elements must be numeric)</td>
<td></td>
</tr>
</tbody>
</table>
Use the INTEGER option after field-name-1 to ignore the fractional portion of the value being assigned. INTEGER causes only the numerals to the left of the decimal point to be transferred during the assignment.

Use ROUNDED or TRUNCATED when the receiving field (field-name-1) is too small to handle the fractional result of the assignment. TRUNCATED is the default.

Use the ROUNDED option after field-name-1 to round off the fractional result of the assignment statement. Rounding takes place in the commonly accepted manner. That is, the least significant digit of the result (receiving field) has its value increased by one when the most significant digit of the excess decimal digits is greater than or equal to five. For example, if 10.75 is the value of the sending field and the receiving field has one decimal place, ROUNDED causes the receiving field to be 10.8.

Use the TRUNCATED option to truncate the result of the assignment statement. Low order digits are truncated on the right as necessary when the result is moved to the receiving field.

If INTEGER is used with ROUNDED, the result is rounded to the nearest integer before the INTEGER function is performed. If INTEGER is used with TRUNCATED, then only the INTEGER function is performed.

Assume:

| SENDFLD | W 5 N 2 VALUE(10.75) |
| RCVFLD  | W 5 N 1               |

Then:

<table>
<thead>
<tr>
<th>Assignment Statement</th>
<th>RCVFLD Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCVFLD INTEGER ROUNDED = SENDFLD</td>
<td>11.0</td>
</tr>
<tr>
<td>RCVFLD INTEGER TRUNCATED = SENDFLD</td>
<td>10.0</td>
</tr>
<tr>
<td>RCVFLD INTEGER = SENDFLD</td>
<td>10.0</td>
</tr>
<tr>
<td>RCVFLD ROUNDED = SENDFLD</td>
<td>10.8</td>
</tr>
<tr>
<td>RCVFLD TRUNCATED = SENDFLD</td>
<td>10.7</td>
</tr>
<tr>
<td>RCVFLD = SENDFLD</td>
<td>10.7</td>
</tr>
</tbody>
</table>

INTEGER, ROUNDED, and TRUNCATED are valid only with numeric fields.
Format 2 (Logical Expression)

Format 2 of the Assignment statement sets the value of field-name-1 equal to the result of evaluating a logical expression. The value of field-name-2 is logically acted upon by the value of field-name-3 or literal-2. The lengths of all values must be the same and literal-2 must be hexadecimal. The logic operators are processed as follows:

AND zero bits in field-name-3 or literal-2 are carried forward to field-name-2 and the result is placed in field-name-1.

OR one bit in field-name-3 or literal-2 is carried forward to field-name-2 and the result is placed in field-name-1.

XOR corresponding bits of field-name-3 or literal-2, and field-name-2 must be opposite (zero and one) to result in a one bit in field-name-1.

Examples

The following examples of the Assignment statement illustrate its various rules:

Assignment Format 1 (Field-name-1 is Alphanumeric)

Format 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1A</td>
<td>4</td>
<td>A</td>
<td>VALUE 'A'</td>
</tr>
<tr>
<td>F2A1</td>
<td>1</td>
<td>A</td>
<td>VALUE 'A'</td>
</tr>
<tr>
<td>F2A2</td>
<td>6</td>
<td>A</td>
<td>VALUE 'ABCDEF'</td>
</tr>
<tr>
<td>F2N1</td>
<td>2</td>
<td>N</td>
<td>VALUE 12</td>
</tr>
<tr>
<td>F2N2</td>
<td>3</td>
<td>P 1</td>
<td>VALUE 1234.5</td>
</tr>
</tbody>
</table>

Resulting Value

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1A</td>
<td>'A'</td>
</tr>
<tr>
<td>F1A</td>
<td>'ABCD'</td>
</tr>
<tr>
<td>F1A</td>
<td>'0012'</td>
</tr>
<tr>
<td>F1A</td>
<td>'2345'</td>
</tr>
<tr>
<td>F1A</td>
<td>'X'FF'</td>
</tr>
</tbody>
</table>

Note: For an example using varying length alphanumeric fields, see “DEFINE Statement Examples” in Chapter 5.
Assignment Statement

Assignment Format 1 (Field-name-1 is Numeric)

Statements:

DEFINE F1N W 4 N 1
DEFINE F2N1 W 4 N 1 VALUE 1
DEFINE F2N2 W 4 N 1 VALUE 2
DEFINE F2N3 W 4 N 1 VALUE 3
JOB INPUT NULL NAME MYPROG
F1N = F2N1 + F2N2 + F2N3
DISPLAY SKIP 2 +
'F1N = F2N1 + F2N2 + F2N3 = ' F1N
F1N = F2N1 + F2N2 / F2N3
DISPLAY SKIP 2 +
'F1N = F2N1 + F2N2 / F2N3 = ' F1N
F1N = (F2N1 + F2N2) / F2N3
DISPLAY SKIP 2 +
'F1N = (F2N1 + F2N2) / F2N3 = ' F1N
F1N = ((F2N1 / F2N2) * 100) + .5
DISPLAY SKIP 2 +
'F1N = ((F2N1 / F2N2) * 100) + .5 = ' F1N

STOP

Produce:

Resulting
Value

F1N = F2N1 + F2N2 + F2N3 = 6.0
(1 + 2 + 3)

F1N = F2N1 + F2N2 / F2N3 = 1.6
(1 + 2 / 3)
(1 + 0.6666)

F1N = (F2N1 + F2N2) / F2N3 = 1.0
(( 1 + 2) / 3)
(3 / 3)

F1N = ((F2N1 / F2N2) * 100) + .5 = 50.5
(( 1 / 2) * 100) + .5
((0.5 * 100) + .5)
(50 + .5)

Assignment Format 2 (Logical Expression Evaluation)

Statements:

DEFINE F1 W 2 P MASK HEX
DEFINE F2 W 2 P VALUE X'123D'
JOB INPUT NULL NAME MYPROG
F1 = F2 AND X'FFFF'
DISPLAY SKIP 2 +
'F1 = F2 AND X''FFFF'' = ' F1
F1 = F2 OR X'000F'
DISPLAY SKIP 2 +
'F1 = F2 OR X''000F'' = ' F1
F1 = F2 XOR X'FFFF'
DISPLAY SKIP 2 +
'F1 = F2 XOR X''FFFF'' = ' F1
F1 = F2 XOR F2
DISPLAY SKIP 2 +
'F1 = F2 XOR F2 = ' F1

STOP
Assignment Statement

Produce:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Resulting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 = F2 AND X'FFFE'</td>
<td>= 123C</td>
</tr>
<tr>
<td>F1 = F2 OR X'000F'</td>
<td>= 123F</td>
</tr>
<tr>
<td>F1 = F2 XOR X'FFFF'</td>
<td>= EDC2</td>
</tr>
<tr>
<td>F1 = F2 XOR F2</td>
<td>= 0000</td>
</tr>
</tbody>
</table>

**EBCDIC To DBCS Conversion**

When conversion from EBCDIC format to DBCS format is required for the Assignment statement, CA-Easytrieve/Plus converts the EBCDIC data using the technique defined for Katakana conversion literals. That is, CA-Easytrieve/Plus converts:

- The lowercase EBCDIC values into the applicable DBCS Katakana characters
- The other valid EBCDIC characters into their equivalent DBCS English values
- Any non-valid EBCDIC values into DBCS spaces.
### Format 1 (Normal Assignment)

Format 1 sets the value of field-name-1 equal to the value of field-name-2, literal-2, or the arithmetic expression. The rules of the statement are shown below.

<table>
<thead>
<tr>
<th>Format 1 Specification Rule</th>
<th>Resulting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If field-name-1 is DBCS and the right side is:</td>
<td></td>
</tr>
<tr>
<td>field-name-2 (DBCS)</td>
<td>CA-Easytrieve/Plus converts field-name-2 into the DBCS code system of field-name-1. The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
<tr>
<td>(MIXED)</td>
<td>CA-Easytrieve/Plus converts each EBCDIC byte of field-name-2 into its equivalent DBCS value. CA-Easytrieve/Plus converts any DBCS data identified by shift codes to the DBCS code system of field-name-1. CA-Easytrieve/Plus then removes the shift codes. The resulting value of field-name-1 is padded on the right with DBCS spaces, or truncated on the right as necessary.</td>
</tr>
<tr>
<td>(alphabetic)</td>
<td>CA-Easytrieve/Plus converts each byte of field-name-2 into its equivalent DBCS value and stores the resulting value in field-name-1. The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
<tr>
<td>Format 1</td>
<td>Resulting Value</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Specification Rule</strong></td>
<td></td>
</tr>
<tr>
<td>If field-name-1 is DBCS and the right side is:</td>
<td></td>
</tr>
<tr>
<td>(numeric) or (packed) or (binary)</td>
<td>The resulting value of field-name-1 is the zoned decimal equivalent of field-name-2 with each byte converted into the DBCS equivalent. Before the conversion, the result is padded on the left with DBCS zeros, or truncated on the left.</td>
</tr>
<tr>
<td>literal-1 (DBCS)</td>
<td>The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
<tr>
<td>(MIXED)</td>
<td>CA-Easytrieve/Plus converts each EBCDIC byte into its equivalent DBCS value. EASYTRIEVE/PLUS converts any DBCS data identified by shift codes, into the DBCS code system of field-name-1. The shift codes identify the code system of this DBCS data. EASYTRIEVE/PLUS removes the shift codes during the conversion. The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
<tr>
<td>If field-name-2 is DBCS and the left side is:</td>
<td></td>
</tr>
<tr>
<td>(alphanumeric) or (hexadecimal)</td>
<td>CA-Easytrieve/Plus converts each byte of literal-1 into its equivalent DBCS value and stores the result in field-name-1. The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
<tr>
<td>field-name-1 (DBCS)</td>
<td>CA-Easytrieve/Plus converts field-name-2 into the DBCS code system of field-name-1. The resulting value of field-name-1 is padded on the right with DBCS spaces or truncated on the right as necessary.</td>
</tr>
</tbody>
</table>
**Format 1**

<table>
<thead>
<tr>
<th>Specification Rule</th>
<th>Resulting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If field-name-2 is DBCS and the left side is:</td>
<td></td>
</tr>
<tr>
<td>(MIXED)</td>
<td>CA-Easytrieve/Plus converts field-name-2 into the DBCS code system of field-name-1. The shift codes defined for the code system of field-name-1 are added and the resulting value of field-name-1 is padded on the right with EBCDIC spaces or truncated on the right as necessary. When truncation occurs DBCS characters are not split. Truncation is to the nearest double byte.</td>
</tr>
<tr>
<td>Field-name-2 (DBCS)</td>
<td>CA-Easytrieve/Plus converts field-name-2 into the DBCS code system of field-name-1. The shift codes defined for the code system of field-name-1 are added and the resulting value of field-name-1 is padded on the right with EBCDIC spaces or truncated on the right as necessary. When truncation occurs DBCS characters are not split. Truncation is to the nearest double byte.</td>
</tr>
<tr>
<td>If field-name-1 is MIXED and the right side is:</td>
<td></td>
</tr>
<tr>
<td>(MIXED)</td>
<td>The EBCDIC data in field-name-2 is moved unaltered to field-name-1. CA-Easytrieve/Plus converts the DBCS data identified by shift codes to the DBCS code system of field-name-1. CA-Easytrieve/Plus also converts the shift codes to meet the requirements of that code system. The resulting value of field-name-1 is padded on the right with EBCDIC spaces or truncated on the right as necessary. When truncation occurs within the DBCS portion of a field, DBCS characters are not split. Truncation is to the nearest double byte.</td>
</tr>
<tr>
<td>(alphabetic)</td>
<td>Each byte of field-name-2 is moved unaltered to field-name-1. The resulting value of field-name-1 is padded on the right with EBCDIC spaces or truncated on the right as necessary.</td>
</tr>
</tbody>
</table>
### Format 1 Specification Rule

<table>
<thead>
<tr>
<th>Specification Rule</th>
<th>Resulting Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>If field-name-1 is MIXED and the right side is:</td>
<td></td>
</tr>
<tr>
<td>(numeric) or (packed) or (binary)</td>
<td>The resulting value of field-name-1 is the zoned decimal equivalent of field-name-2 with padding or truncation on the left (if necessary).</td>
</tr>
<tr>
<td>literal-1 (DBCS)</td>
<td>CA-Easytrieve/Plus converts field-name-2 into the code system of field-name-1 and adds the correct shift codes. The result is padded on the right with EBCDIC spaces.</td>
</tr>
<tr>
<td>(MIXED)</td>
<td>Each byte is moved to field-name-1 and the result is padded on the right with EBCDIC spaces. CA-Easytrieve/Plus converts any DBCS data identified by shift codes to the DBCS code system of field-name-1. The shift codes are also changed to the valid values.</td>
</tr>
<tr>
<td>(alphanumeric) or (hexadecimal)</td>
<td>Each byte of literal-1 is moved to field-name-1 unaltered. The resulting value of field-name-1 is padded on the right with EBCDIC spaces.</td>
</tr>
<tr>
<td>field-name-1 (DBCS)</td>
<td>CA-Easytrieve/Plus converts each EBCDIC byte of field-name-2 into its equivalent DBCS value. CA-Easytrieve/Plus converts any DBCS bytes identified by shift codes to the DBCS code system of field-name-1. CA-Easytrieve/Plus removes the shift codes. The resulting value of field-name-1 is padded to the right with DBCS spaces or truncated to the right as necessary.</td>
</tr>
<tr>
<td>(MIXED)</td>
<td>The EBCDIC data in field-name-2 is moved unaltered to field-name-1. CA-Easytrieve/Plus converts the DBCS data identified by shift codes to the DBCS code system of field-name-1. CA-Easytrieve/Plus also converts the shift codes to meet the requirements of that code system. The resulting value of field-name-1 is padded on the right with EBCDIC spaces or truncated on the right as necessary. When truncation occurs within the DBCS portion of a field, DBCS characters are not split. Truncation is to the nearest double byte.</td>
</tr>
</tbody>
</table>
MOVE Statement

MOVE transfers character strings from one storage location to another. MOVE is especially useful for moving data without conversion and for moving variable length data strings. The formats of the MOVE statement syntax are:

Syntax

Format 1

{file-name-1} [ {record-name-1} [field-name-2] +
MOVE {field-name-1} [ literal-2 ]
{literal-1} [ ]

TO {file-name-2} [ ]
{record-name-2} [field-name-4] [FILL literal-4]
{field-name-3} [ literal-3 ]
{ } [ ]

Format 2

{SPACE}
{SPACES}
MOVE {ZERO} TO field-name-3 ...
{ZEROS}
{ZEROES}

Format 1

When you specify Format 1, data moves from left to right as if both areas were alphanumeric. The data moved is unconverted.

Format 2

When you specify Format 2, the receiving areas are set to the appropriate format of data, for example, packed zero for fields with a type of P.
Variables

The first parameter after the MOVE keyword (file-name-1, record-name-1, field-name-1, literal-1, SPACE, or ZERO) is the sending data area. The default length of file-name-1 is the current value of file-name-1:RECORD-LENGTH. The length of the sending field is optionally overridden with field-name-2’s current value or literal-2.

Field-name-3, file-name-2, or record-name-2 is the receiving data area. The default length of file-name-2 is the current value of file-name-2:RECORD-LENGTH. The length of the receiving field is optionally overridden with field-name-4’s current value or literal-3. In format 2, multiple field-names can be used for field-name-3.

When field-name-3 is a DBCS format field, the space character of the DBCS code system of field-name-3 is moved to the field. If field-name-3 is a MIXED format field, EBCDIC spaces are moved to the field.

CA-Easytrieve/Plus truncates longer sending fields on the right. Longer receiving fields are padded on the right with EBCDIC spaces or the optional literal-4.

When literal-1 is a non-numeric literal, it must be enclosed within apostrophes.

For EBCDIC fields, literal-4 must be one character. For DBCS fields, literal-4 must be two characters. Non-numeric literals must be enclosed within apostrophes. When literal-4 contains numeric characters, they are treated as a zoned decimal value.

File-name-1, record-name-1, file-name-2, and record-name-2 can be any file or database record with current data availability. When file-name-1 is an IMS/DLI or IDMS file, all records in the file are moved.
Example

The following exhibit demonstrates the use of the MOVE statement and shows the results:

Statements:

```
DEFINE ASTERISK-LINE W 10 A VALUE '=========='
DEFINE COUNTER-1  W 10 N VALUE 99
DEFINE COUNTER-2  W  2 N VALUE 66
JOB INPUT NULL NAME MYPROG
   DISPLAY COUNTER-1 +2 COUNTER-2
   MOVE ZEROS TO COUNTER-1 COUNTER-2
   DISPLAY COUNTER-1 +2 COUNTER-2
   DISPLAY ASTERISK-LINE
   MOVE '*' TO ASTERISK-LINE FILL '*'
   DISPLAY ASTERISK-LINE
   STOP
```

Produce:

```
0000000099   66
0000000000   00
==========
**********
```

MOVE LIKE Statement

MOVE LIKE moves the contents of fields with identical names from one file to another. Data movement and conversion follow the rules of the Assignment statement. The MOVE LIKE statement syntax is:

Syntax

```
{             }    {             }
MOVE LIKE {file-name-1} TO {file-name-2}
{record-name-1}    {record-name-2}
```

Operation

When you issue a MOVE LIKE statement, the contents of fields in file-name-1 or record-name-1 replace the contents of fields with identical names in file-name-2 or record-name-2. When file-name-2 or record-name-2 contain overlapping fields, the order in which the fields are defined is important. The moves occur starting with the last identically named field in file-name-2 or record-name-2 and ending with the first identically named field in the file. The Example, shown later, illustrates the use of the MOVE LIKE statement.
IDD Processing of MOVE LIKE (CA-IDMS)

In IDD processing, the fields of a file defined by an IDD statement are organized into group item structures. A group item is a field subdivided by smaller fields. The smaller fields can themselves be group items and therefore subdivided by even smaller fields. A group item is said to “own” its subdividing fields. A field without subdivision is called an “elementary” field.

In IDD processing, MOVE LIKE assigns a new value to the receiving field if all of the following conditions are met:

- The sending and receiving fields have matching names.
- The sending and receiving fields have matching qualifier (group item) names.
- Either the sending or receiving field is an elementary field.

Record name qualifiers do not participate in the process of matching qualifiers between two fields. For example, in a MOVE LIKE from a record to a file containing records, no matching is done between the record names. Therefore, fields can be matched to receiving fields regardless of which records the receiving fields are in. One source field can be matched to a field under one record and another source field can be matched to a field under a different record.

Example

FILE PERSNL FB(150 1800)
REGION     1   1   N
BRANCH     2   2   N
NAME      17  16   A
        NAME-LAST 17  8  A
        NAME-FIRST 25  8  A
FILE MYFILE FB(150 1800)
COPY PERSNL
JOB INPUT PERSNL NAME MYPROG
       MOVE LIKE PERSNL TO MYFILE
       PUT MYFILE

In the above example, MOVE LIKE generates the following Assignment statements:

MYFILE:NAME-FIRST = PERSNL:NAME-FIRST
MYFILE:NAME-LAST = PERSNL:NAME-LAST
MYFILE:NAME = PERSNL:NAME
MYFILE:BRANCH = PERSNL:BRANCH
MYFILE:REGION = PERSNL:REGION

Whatever values were in the fields of the file PERSNL are now found in the fields of the file MYFILE.
Chapter 8
Decision and Branching Logic

Introduction

CA-Easytrieve/Plus uses certain statements to control the execution of your program by means of decision and branching logic. These statements govern the flow of execution in your program depending on the truth value of the conditional expressions.

The following statements are associated with CA-Easytrieve/Plus decision and branching logic and are discussed in this chapter:

- IF, ELSE-IF, ELSE, and END-IF
- DO WHILE, DO UNTIL, and END-DO
- CASE
- GOTO
- PERFORM proc-name. PROC
- Statement Label
- STOP.

Conditional Expressions

Conditional expressions used as parameters of IF, ELSE-IF, and DO statements offer an alternative to the normal top to bottom execution of CA-Easytrieve/Plus statements. Conditional expressions are also used in the WHILE subparameter of the RETRIEVE statement in the IMS/DLI and CA-IDMS interfaces. The syntax of a conditional expression is:

{IF              }
{ELSE-IF         }            [ {   }           ]
{DO WHILE        }   condition[ {AND}  condition]...
{DO UNTIL        }            [ {OR }           ]
{RETRIEVE...WHILE}
CA-Easytrieve/Plus accepts seven different conditions:

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

The following are skeletal examples of each type of conditional expression used in an IF statement:

Field Relational     - IF field-1 = field-2
Field Series         - IF field-1 = field-2, field-3, field-4
Field Class          - IF field-1 ALPHABETIC
Field Bits           - IF field-1 ON X'0F4000'
File Presence        - IF EOF file-name
File Presence Series - IF MATCHED file-1, file-2, file-3
Record Relational    - IF DUPLICATE file-name

**Simple Conditions**

There are five 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
- Field Bits
- File Presence
- Record Relational.

**Extended Conditions**

The extended conditions are:

- Field Series
- File Presence Series.
**DBCS Considerations**

The following conditions provide support for DBCS and MIXED fields:
- Field Relational
- Field Series
- Field Class
- Field Bits.

As with the data equates, when conversion from EBCDIC to DBCS format is part of the conditional expression, CA-Easytrieve/Plus converts the EBCDIC data using the technique defined for Katakana conversion literals. That is, CA-Easytrieve/Plus converts:
- Lowercase EBCDIC values into the applicable DBCS Katakana characters
- Other valid EBCDIC characters into their equivalent English values
- Non-valid EBCDIC values into DBCS spaces.

**Varying Length Fields**

Comparisons of varying length fields (fields which use the VARYING option of the DEFINE statement) are based on the length of the data at the time of the comparison with the following exceptions.
- If the subject is a zero length VARYING field and the object is either null (' ') or another VARYING field of length zero, the comparison is true.
- If the subject is a zero length VARYING field and the object is not either null (' ') or another VARYING field of length zero, the comparison is false.

See Chapter 5 for more details on variable length fields.
Combined Conditions

Any of these conditions, simple and extended, can be combined using the logical connectors AND or OR in any combination.

In the case of combined conditions, those connected by AND are evaluated first. The connected condition is true only if all of the conditions are true. The conditions connected by OR are then evaluated. The combined condition is true if any of the connected conditions are true. You can use parentheses to override the normal AND/OR relationships. The exhibit below illustrates the results of combining conditions with AND, OR, and parentheses. The values x, y, and z represent any condition.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x OR y</th>
<th>x AND y</th>
<th>(x OR y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Field Relational Condition

The field relational condition compares fields with values. Its syntax is:

**Syntax**

<table>
<thead>
<tr>
<th>IF</th>
<th>ELSE-IF</th>
<th>DO WHILE</th>
<th>DO UNTIL</th>
<th>RETRIEVE ... WHILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>{IF  }</td>
<td>{ELSE-IF}</td>
<td>{DO WHILE}</td>
<td>{DO UNTIL}</td>
<td>{RETRIEVE ... WHILE}</td>
</tr>
<tr>
<td>Subject</td>
<td>Operator</td>
<td>Field-name-1</td>
<td>Literal</td>
<td>Arithmetic expression</td>
</tr>
<tr>
<td>Field-name-1</td>
<td>EQ =</td>
<td>LT &lt;</td>
<td>LE &lt;=</td>
<td>GT &gt;</td>
</tr>
<tr>
<td>Field-name-2</td>
<td>NE ¬=</td>
<td>LS</td>
<td>¬&lt;</td>
<td>GQ ¬&lt;</td>
</tr>
</tbody>
</table>

**Operation**

**Subject**

Field-name-1 is the subject of the comparison.
**Relational Operator**

Code any of the relational operators to control the condition's evaluation process.

**Object**

Code field-name-2, a literal, or an arithmetic expression to designate the object of the comparison. Note that alphanumeric literals must be enclosed in apostrophes. Refer to Chapter 7, “Assignments and Moves: Arithmetic Expressions,” for a complete description of how CA-Easytrieve/Plus evaluates arithmetic expressions.

**Alphanumeric Subjects**

When the condition subject is an alphanumeric field, the following evaluation rules apply:

1. The object must be either a field or an alphanumeric literal.
2. If necessary, numeric field objects are converted to zoned decimal.
3. The length of the object is adjusted, by truncation or padding, to match the length of the subject. Refer to Chapter 7, “Assignments and Moves,” for more information on data adjustment.
4. Comparison is logical (bit-by-bit).

**Numeric Subjects**

When the condition subject is a numeric field, the following evaluation rules apply:

1. The object must be either a numeric field, a numeric literal, or an arithmetic expression.
2. Comparison is arithmetic.

**Mixed Subjects**

When the condition subject is a MIXED field, the following evaluation rules apply:

1. CA-Easytrieve/Plus only supports the Equal (EQ =) and Not Equal (NE ≠ NQ) conditions. The use of any other conditional operators causes an error.
2. The object must be either a field, an alphanumeric literal, a MIXED literal, or a DBCS literal.
3. CA-Easytrieve/Plus does not perform a conversion if the object is an EBCDIC alphanumeric field or literal.

4. If the object is a MIXED field or literal, CA-Easytrieve/Plus converts the DBCS portion of data into the DBCS code system of the subject. Also, CA-Easytrieve/Plus converts the shift codes to the values defined for the DBCS code system of the subject in the DBCS Options module.

5. If the object is a DBCS field or literal, CA-Easytrieve/Plus converts the data into the DBCS code system of the subject. Once converted, the shift codes defined for the code system of the subject are added to the data.

6. CA-Easytrieve/Plus converts numeric field objects to zoned decimal (if necessary).

7. To match the length of the subject, CA-Easytrieve/Plus truncates or pads the object. Padding uses the EBCDIC space character. During truncation, no DBCS character is split. When truncation occurs within the DBCS portion of a field, the truncation is adjusted to the nearest double byte boundary.

8. Comparison is logical (bit-by-bit).

**DBCS Subjects**

When the condition subject is a DBCS field, the following evaluation rules apply:

1. CA-Easytrieve/Plus only supports the Equal (EQ =) and Not Equal (NE ≠ NQ) conditions. The use of any other conditional operators causes an error.

2. The object must be either a field, an alphanumeric literal, a MIXED literal, or a DBCS literal.

3. If the object is an EBCDIC alphanumeric field or literal, then CA-Easytrieve/Plus converts each character into the DBCS code system of field-name-1.

4. If the object is a MIXED field or literal, CA-Easytrieve/Plus converts the DBCS portion of data into the DBCS code system of the subject. CA-Easytrieve/Plus also converts the EBCDIC portion of data to its equivalent DBCS value based on the code system of the subject. CA-Easytrieve/Plus removes shift codes.

5. If the object is a DBCS field or literal, CA-Easytrieve/Plus converts the data into the DBCS code system of the subject.

6. If necessary, CA-Easytrieve/Plus converts numeric field objects to zoned decimal and then converts the EBCDIC result into the equivalent DBCS characters based on the code system of field-name-1.

7. To match the length of the subject, CA-Easytrieve/Plus truncates or pads the object. Padding uses the DBCS space character.

8. Comparison is logical (bit-by-bit).
Example

The following exhibit illustrates various field relational conditions:

```plaintext
FILE PERSNL FB(150 1800)
EMP#       9   5   N
NAME       17  20  A
  NAME-LAST NAME      8   A
  NAME-FIRST NAME +8  12  A
PAY-NET    90  4   P 2
PAY-GROSS  94  4   P 2
SEX        127 1   N
TOTAL-EMP# W 3 N VALUE 0
TOTAL-SEX  W 3 N VALUE 0
TOTAL-PAY  W 3 N VALUE 0
TOTAL-FIRST-NAME W 3 N VALUE 0
MALE       W 1 N VALUE 1
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
  IF EMP# GT 10000
      TOTAL-EMP# = TOTAL-EMP# + 1
  END-IF
  IF SEX NE MALE
      TOTAL-SEX = TOTAL-SEX + 1
  END-IF
  IF PAY-NET LT (PAY-GROSS / 2)
      TOTAL-PAY = TOTAL-PAY + 1
  END-IF
  IF NAME-FIRST EQ 'LINDA'
      TOTAL-FIRST-NAME = TOTAL-FIRST-NAME + 1
  END-IF
* FINISH-PROC. PROC
  DISPLAY TOTAL-EMP#
  DISPLAY TOTAL-SEX
  DISPLAY TOTAL-PAY
  DISPLAY TOTAL-FIRST-NAME
END-PROC
```

Field Series Condition

The field series condition compares a field and a series or a range of values. Its syntax is:

```
Relational

<table>
<thead>
<tr>
<th>Subject</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>}</td>
</tr>
<tr>
<td>ELSE-IF</td>
<td>{ }</td>
</tr>
<tr>
<td>DO WHILE</td>
<td>field-name-1{EQ } +</td>
</tr>
<tr>
<td>DO UNTIL</td>
<td>{NE ¬= NQ}</td>
</tr>
<tr>
<td>RETRIEVE...WHILE</td>
<td></td>
</tr>
</tbody>
</table>

Object

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[field-name-2</td>
<td>THRU{ field-name-3 ]....}</td>
<td></td>
</tr>
<tr>
<td>[literal-1</td>
<td>{ literal-2 ]</td>
<td></td>
</tr>
</tbody>
</table>
```
Field Series Condition

**Operation**

**Subject**

Field-name-1 is the subject of the comparison.

**Relational Operator**

Equal and not equal are the only valid relational operators for field series conditions.

**Object**

Code field-name-2 or a literal-1 as often as you need to indicate the series of comparison objects. Field-name-2 through field-name-3, field-name-2 through literal-2, literal-1 through field-name 3, or literal-1 through literal-2 designate a value range.

**Note:** Alphanumeric literals must be enclosed in apostrophes.

**Rules for Evaluation**

Evaluation rules for field series conditions are as follows:

1. Alphanumeric (including DBCS and MIXED format fields) and numeric fields are evaluated as in the field relational condition.

2. An equal (=) relational operator tests if the subject is equal to or within range of any of the series of values comprising the object.

3. A not equal (¬=) relational operator tests if the subject is unequal to or outside the range of all the series of values comprising the object.
Example

The following exhibit illustrates the field series condition:

```plaintext
FILE PERSNL FB(150 1800)
REGION       1 1 N
BRANCH       2 2 N
DEPT         98 3 N
MARITAL-STAT 128 1 A
*
TOTAL-REGION  W 3 N VALUE 0
TOTAL-BRANCH  W 3 N VALUE 0
TOTAL-DEPT    W 3 N VALUE 0
TOTAL-MARITAL W 3 N VALUE 0
WORK-REGION  W 2 N VALUE 04
*
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
  IF REGION = 0, 8, 9
    TOTAL-REGION = TOTAL-REGION + 1
  END-IF
  IF BRANCH NE 01, WORK-REGION
    TOTAL-BRANCH = TOTAL-BRANCH + 1
  END-IF
  IF DEPT EQ 940 THRU 950
    TOTAL-DEPT = TOTAL-DEPT + 1
  END-IF
  IF MARITAL-STAT NE 'M', 'S'
    TOTAL-MARITAL = TOTAL-MARITAL + 1
  END-IF
*
FINISH-PROC. PROC
  DISPLAY TOTAL-REGION
  DISPLAY TOTAL-BRANCH
  DISPLAY TOTAL-DEPT
  DISPLAY TOTAL-MARITAL
END-PROC
```

Field Class Condition

The field class condition determines if all positions of a field contain alphabetic, numeric, space, or zero characters. Its syntax is:

**Syntax**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ALPHABETIC}</td>
<td></td>
</tr>
<tr>
<td>{BREAK}</td>
<td></td>
</tr>
<tr>
<td>{KANJI}</td>
<td></td>
</tr>
<tr>
<td>{HIGHEST-BREAK}</td>
<td></td>
</tr>
<tr>
<td>{IF}</td>
<td></td>
</tr>
<tr>
<td>{ELSE-IF}</td>
<td></td>
</tr>
<tr>
<td>{DO WHILE}</td>
<td></td>
</tr>
<tr>
<td>{DO UNTIL}</td>
<td></td>
</tr>
<tr>
<td>{RETRIEVE ... WHILE}</td>
<td></td>
</tr>
<tr>
<td>field-name [NOT]</td>
<td></td>
</tr>
<tr>
<td>{NUMERIC}</td>
<td></td>
</tr>
<tr>
<td>{SPACE}</td>
<td></td>
</tr>
<tr>
<td>{SPACES}</td>
<td></td>
</tr>
<tr>
<td>{ZERO}</td>
<td></td>
</tr>
<tr>
<td>{ZEROS}</td>
<td></td>
</tr>
<tr>
<td>{ZEROES}</td>
<td></td>
</tr>
</tbody>
</table>
**Operation**

**Subject**

Field-name is the subject of the comparison. Each byte of the field must pass the test before the test is true.

**Object**

The object determines the class of data to be tested for.

The NOT parameter indicates that the condition test is reversed.

**Rules for Evaluation**

Evaluation rules for field class conditions are as follows:

**{ALPHABETIC}**

ALPHABETIC tests for the characters A through Z or a blank space in each byte of the subject field.

**{BREAK}**

BREAK tests whether this field is currently being processed as a CONTROL break field on a report. The BREAK test is an alternative to testing the field-name LEVEL for a specific numeric value. Field-name must be defined on a CONTROL statement or it must be the reserved word FINAL.

**{HIGHEST-BREAK}**

HIGHEST-BREAK tests whether this field caused the CONTROL break on a report. The HIGHEST-BREAK test is an alternative to testing the field-name BREAK-LEVEL for a specific numeric value. Field-name must be defined on a CONTROL statement or it must be the reserved word FINAL.

**{KANJI}**

KANJI tests for valid DBCS values applicable to the DBCS code system of the field. The DBCS Options Module Chapter of the *CA-Easytrieve/Plus Installation Guide* defines the valid values for each DBCS code system.
MIXED tests each DBCS character in the field for valid DBCS values applicable to the DBCS code system of the field. The DBCS data must be identified by the correct shift codes for the DBCS code system of the field. Appendix B, “Installing DBCS Option,” of the CA-Easytrieve/Plus Installation Guide defines the valid values for each DBCS code system. If the field contains only EBCDIC data, this test always proves to be correct.

NUMERIC tests for the digits 0 through 9 (in the correct format for the field's data type), and for a possible algebraic sign in the low-order position of type P fields or in the high-order position of type N fields.

SPACE and SPACES test for the character space in each byte of EBCDIC and MIXED format subjects and each double byte of DBCS subjects. The DBCS code system of the subject defines the value of the appropriate DBCS space character.

ZERO, ZEROS, and ZEROES test for the digit 0 (in the correct format for the field's data type), and for a possible algebraic sign in the low-order position of type P fields or in the high-order position of type N fields.
Example

The following exhibit illustrates the use of the field class condition:

```
FILE PERSNL FB(150 1800)
REGION       1   1 N
BRANCH       2   2 N
NAME         17  20 A
   NAME-LAST NAME   8 A
   NAME-FIRST NAME +8 12 A
* 
TOTAL-NUMERIC   W 3   N VALUE 0
TOTAL-NON-ZEROS W 3   N VALUE 0
TOTAL-ALPHABETIC W 3   N VALUE 0
* 
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
   IF REGION NUMERIC
       TOTAL-NUMERIC = TOTAL-NUMERIC + 1
   END-IF
   IF BRANCH NOT ZERO
       TOTAL-NON-ZEROS = TOTAL-NON-ZEROS + 1
   END-IF
   IF NAME ALPHABETIC
       TOTAL-ALPHABETIC = TOTAL-ALPHABETIC + 1
   END-IF
* 
FINISH-PROC. PROC
   DISPLAY TOTAL-NUMERIC
   DISPLAY TOTAL-NON-ZEROS
   DISPLAY TOTAL-ALPHABETIC
END-PROC
```
Field Bits Condition

The field bits condition compares selected bits of a field for on (1) or off (0) conditions. Its syntax is:

Syntax

<table>
<thead>
<tr>
<th>Subject</th>
<th>Relational Operator</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>{IF}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ELSE-IF }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{DO WHILE }</td>
<td>field-name-1 [NOT]</td>
<td>{ ON} {field-name-2}</td>
</tr>
<tr>
<td>{DO UNTIL }</td>
<td></td>
<td>{OFF} {literal   }</td>
</tr>
<tr>
<td>{RETRIEVE...WHILE}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Operation

Subject

Field-name-1 is the subject of the comparison. It can be any field type.

Relational Operator

The relational operators ON and OFF test for bit values of one or zero respectively.

The NOT parameter indicates the condition test is reversed.

Object

Field-name-2 or a literal establish the bits mask to be tested. CA-Easytrieve/Plus tests only those bits that correspond to one (1) bits in the mask. The length of the object must equal the length of the subject. When you code literal as the object, it must be a hexadecimal literal. Indicate a hexadecimal literal by preceding it with an X.
File Presence Condition

Example

The following exhibit illustrates the use of the field bits condition:

```plaintext
DEFINE FIELD-1 W 1 B VALUE X'20'
DEFINE FIELD-NUM W 4 B VALUE X'FF00FF00'
DEFINE PATTERN-8 W 1 B VALUE X'80'
DEFINE LOWER-CASE W 1 A VALUE X'81'

* 
JOB INPUT NULL NAME MYPROG
   IF FIELD-1 ON PATTERN-8
      DISPLAY 'PERFORM CODE FOR PATTERN 8'
   END-IF
   IF LOWER-CASE OFF X'40'
      DISPLAY 'THIS LETTER IS LOWER CASE'
   END-IF
   IF FIELD-NUM ON X'FF000000'
      DISPLAY '1ST BYTE HIGH VALUES'
   END-IF
STOP
```

File Presence Condition

The file presence condition determines whether a record of the file is currently available for processing. Its syntax is:

Syntax

```plaintext
{IF} {ELSE-IF} {NOT} {EOF} {file-name}
{DO WHILE} {DO UNTIL} {file-name}
{file-name} {file-name}
```

Operation

Subject

File-name, PRIMARY, and SECONDARY designate the subject of the test. PRIMARY and SECONDARY are special names for the first and second files in a synchronized file input JOB.
Object

The object of the test is simply the availability of the record for processing.

The optional EOF parameter causes the test to be true when the subject is at end-of-file. This test can never be true for automatic input files that are not involved in synchronized file processing.

The optional NOT parameter reverses the condition test.

The file is available if the last GET or READ operation was successful and there is a record that can be accessed.

Example

The following exhibit illustrates the use of the file presence condition:

```plaintext
FILE PERSNL FB(150 1800)  
%PERSNL
FILE INVENT FB(200 3200)  
%INVMSTR
FILE SORT1  FB(150 1800) VIRTUAL
COPY PERSNL
FILE SORT2  FB(200 3200) VIRTUAL
COPY INVENT
COUNT-1   W 3 N VALUE 0
COUNT-2   W 3 N VALUE 0
*
SORT PERSNL TO SORT1 USING (ADDR-STATE) NAME MYSORT1
SORT INVENT TO SORT2 USING (LOCATION-STATE) NAME MYSORT2
*
JOB INPUT (SORT1 KEY (ADDR-STATE), +
    SORT2 KEY (LOCATION-STATE)) +
    NAME MYPROG FINISH FINISH-PROC
    IF EOF SECONDARY
        DISPLAY 'EOF ON SECONDARY'
    STOP
    END-IF
    IF NOT PRIMARY
        DISPLAY 'NO PERSONNEL RECORD- ' LOCATION-STATE
    END-IF
    IF NOT SECONDARY
        DISPLAY 'NO INVENTORY RECORD- ' ADDR-STATE
    END-IF
    IF SORT1
    * HOW MANY PERSONNEL RECORDS RETURNED
        COUNT-1 = COUNT-1 + 1
    END-IF
    IF SORT2
    * HOW MANY INVENT RECORDS RETURNED
        COUNT-2 = COUNT-2 + 1
    END-IF
    *
FINISH-PROC. PROC
    DISPLAY COUNT-1
    DISPLAY COUNT-2
END-PROC
```
File Presence Series Condition

The file presence series condition determines file presence and record matching for more than one file in JOBs with synchronized file input (See Chapter 12, “File Processing”). Its syntax is:

Syntax

Subject

{IF} {file-name}
{ELSE-IF} [NOT] MATCHED [PRIMARY] ...
{DO WHILE} [SECONDARY]
{DO UNTIL}

Operation

Subject

The optional file-name, PRIMARY, and SECONDARY parameters identify the files to be tested. When you do not code this parameter, the condition is true only if all input files have matching records.

The optional NOT parameter reverses the condition test.

Example

The following exhibit illustrates the use of the file presence series condition:

FILE PERSNL FB(150 1800)
&PERSNL
FILE INVENT FB(200 3200)
%INVMSTR
FILE SORT1 FB(150 1800) VIRTUAL
COPY PERSNL
FILE SORT2 FB(200 3200) VIRTUAL
COPY INVENT
COUNT-1 W 3 N VALUE 0
* SORT PERSNL TO SORT1 USING (ADDR-STATE) NAME MYSORT1
SORT INVENT TO SORT2 USING (LOCATION-STATE) NAME MYSORT2
* JOB INPUT (SORT1 KEY (ADDR-STATE), +
      SORT2 KEY (LOCATION-STATE)) +
      NAME MYPROG FINISH FINISH-PROC
      IF MATCHED
        COUNT-1 = COUNT-1 + 1
      END-IF
* FINISH-PROC. PROC
    DISPLAY COUNT-1
    END-PROC
Record Relational Condition

The record relational condition determines the relationship of the current record of a file to the previous and next records of the same file. This test is valid only for synchronized file processing (see Chapter 12, “File Processing”). The condition syntax is:

Syntax

```
{IF      }       {         }     {         }
{ELSE-IF } [NOT] {DUPLICATE}     {file-name}
{DO WHILE}       {FIRST-DUP}     {PRIMARY  }
{DO UNTIL}       {LAST-DUP }     {SECONDARY}
```

Operation

{DUPLICATE}

DUPLICATE is true when the previous or next record has the same key as the current record.

{FIRST-DUP}

FIRST-DUP is true for the first of two or more records with the same key.

{LAST-DUP}

LAST-DUP is true for the last of two or more records with the same key.

Subject

The file-name, PRIMARY, and SECONDARY parameters identify the file to be tested.

The optional NOT parameter reverses the condition.
Example

The following exhibit illustrates the use of the record relational condition:

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
FILE INVENT FB(200 3200)
%INVMSTR
FILE SORT1 FB(150 1800) VIRTUAL
COPY PERSNL
FILE SORT2 FB(200 3200) VIRTUAL
COPY INVENT
COUNT-1 W 3 N VALUE 0
COUNT-2 W 3 N VALUE 0
*  
SORT PERSNL TO SORT1 USING (ADDR-STATE) NAME MYSORT1
SORT INVENT TO SORT2 USING (LOCATION-STATE) NAME MYSORT2  
*  
JOB INPUT (SORT1 KEY (ADDR-STATE) +
  SORT2 KEY (LOCATION-STATE)) +
  NAME MYPROG FINISH FINISH-PROC
  IF DUPLICATE PRIMARY
    COUNT-1 = COUNT-1 + 1
  END-IF
  IF DUPLICATE SORT2
    COUNT-2 = COUNT-2 + 1
  END-IF
*  
FINISH-PROC. PROC
  DISPLAY COUNT-1
  DISPLAY COUNT-2
END-PROC
```

IF, ELSE-IF, ELSE, and END-IF Statements

The IF statement controls the execution of its associated statements. Associated statements are those that are coded between IF and END-IF. The syntax and logic of the IF, ELSE-IF, ELSE, and END-IF are:

Syntax

```plaintext
IF  conditional-expression-1
  [statement-1]

  [ELSE-IF  conditional-expression-2]  [ . . . ]
  [   [statement-2]   ]  [ . . . ]

  [ELSE
    [   [statement-3]   ]
    [     ]

END-IF
```
Operation

The truth value of the conditional-expression-1 determines whether statement-1 is executed. CA-Easytrieve/Plus executes statements designated by statement-1 when conditional-expression-1 is true. When conditional-expression-1 is false, CA-Easytrieve/Plus tests conditional-expression-2 if the ELSE-IF is specified.

If ELSE-IF is specified, the truth value of conditional-expression-2 determines whether statement-2 is executed. CA-Easytrieve/Plus executes statements designated by statement-2 when conditional-expression-2 is true. When conditional-expression-2 is false, CA-Easytrieve/Plus tests conditional-expression of the ELSE-IF, if specified. If the last ELSE-IF's conditional-expression is also false, CA-Easytrieve/Plus executes statements designated by statement-3. Nest as many ELSE-IFs within the IF as necessary; terminate the IF with a single END-IF.

If ELSE-IF is not specified and conditional-expression-1 is false, CA-Easytrieve/Plus executes statements designated by statement-3.

If the ELSE statement is not specified and the conditional expression is false, no statements are executed and control passes to the statement following the END-IF.

Statement-1, statement-2, and statement-3 each represent any number of CA-Easytrieve/Plus statements. Whenever one or more of these statements is an IF statement, the IFs are considered to be nested. The format of nested IFs is simply that statement-1, statement-2, and statement-3 of any IF can be an IF statement.
ELSE-IF

ELSE-IF is optional and identifies a conditional expression to be tested when the previous conditional expression is false. ELSE-IFs enable multiple conditions to be nested without requiring an END-IF for each condition. You can code as many ELSE-IFs as necessary.

ELSE

ELSE is optional and identifies the statements to be executed when conditions are false. When the conditions of the preceding IF are not satisfied and ELSE-IF is not specified, CA-Easytrieve/Plus bypasses the statements between IF and ELSE and continues execution with the statement following ELSE.

END-IF

END-IF terminates the logic associated with the previous IF statement. An END-IF statement must be specified after each IF statement and its associated statements. You must not specify an END-IF for an ELSE-IF.

Example

The next three exhibits illustrate IF statement usage. In each of the illustrated cases, the field XMAS-BONUS is computed to be either three or five percent over PAY-GROSS. When the field PAY-GROSS is non-numeric, a warning message is issued and the record is bypassed from further processing.

Without Nested IFs

```
FILE PERSNL FB(150 1800)
XPERSNL
XMAS-BONUS       W 4 P 2 VALUE 0
TOT-XMAS-BONUS   W 6 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
   IF PAY-GROSS NOT NUMERIC
      DISPLAY EMP# ' PERSONNEL RECORD IS DAMAGED'
      GO TO JOB
   END-IF
   IF PAY-GROSS > 500.00
      XMAS-BONUS = PAY-GROSS * 1.03
   ELSE
      XMAS-BONUS = PAY-GROSS * 1.05
   END-IF
   TOT-XMAS-BONUS = TOT-XMAS-BONUS +
                 + XMAS-BONUS
   PRINT MYREPORT
   *
FINISH-PROC. PROC
DISPLAY
DISPLAY 'TOTAL $ SPENT IN BONUS ' +
       ' MONEY ====> ' TOT-XMAS-BONUS
END-PROC
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS
```
With Nested IFs

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
XMAS-BONUS W 4 P 2 VALUE 0
TOT-XMAS-BONUS W 6 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
  IF PAY-GROSS NOT NUMERIC
    DISPLAY EMP# ' PERSONNEL RECORD IS DAMAGED'
    GOTO JOB
  ELSE
    IF PAY-GROSS > 500.00
      XMAS-BONUS = PAY-GROSS * 1.03
    ELSE
      XMAS-BONUS = PAY-GROSS * 1.05
    END-IF
    END-IF
    TOT-XMAS-BONUS = TOT-XMAS-BONUS + XMAS-BONUS
    PRINT MYREPORT
  *
FINISH-PROC. PROC
DISPLAY
DISPLAY 'TOTAL $ SPENT IN BONUS ' +
'MONEY =====> ' TOT-XMAS-BONUS
END-PROC
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS
```

With ELSE-IFs

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
XMAS-BONUS W 4 P 2 VALUE 0
TOT-XMAS-BONUS W 6 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
  IF PAY-GROSS NOT NUMERIC
    DISPLAY EMP# ' PERSONNEL RECORD IS DAMAGED'
    GOTO JOB
  ELSE-IF PAY-GROSS > 500.00
    XMAS-BONUS = PAY-GROSS * 1.03
  ELSE
    XMAS-BONUS = PAY-GROSS * 1.05
  END-IF
  END-IF
  TOT-XMAS-BONUS = TOT-XMAS-BONUS + XMAS-BONUS
  PRINT MYREPORT
  *
FINISH-PROC. PROC
DISPLAY
DISPLAY 'TOTAL $ SPENT IN BONUS ' +
'MONEY =====> ' TOT-XMAS-BONUS
END-PROC
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS
```
DO and END-DO Statements

The loop control statements DO and END-DO control and delimit repetitive program logic. Their syntax and logic is:

Syntax

```
{WHILE}
DO {UNTIL} conditional-expression
     
statement-1
... 
statement-n
END-DO
```

Operation

The DO statements of DO WHILE and DO UNTIL operate in conjunction with the END-DO statement.
DO WHILE and END-DO

The truth value of the conditional expression determines whether statement-1 through statement-n are executed. When the conditional expression is true, the statements are executed and the program branches back to test the conditional expression. The program continues to loop for as long as the conditional expression is true. When the conditional expression is false, the program branches to the statement following the END-DO.

Statement-1 and statement-n represent any number of CA-Easytrieve/Plus statements.

DO UNTIL and END-DO

Statement-1 through statement-n are executed. The truth value of the conditional expression determines whether the group of statements are executed again. When the conditional expression is true, the program branches to the statement following the END-DO. When the conditional expression is false, the program branches back to execute the statements. The program continues to loop until the conditional expression is true.

Statement-1 and statement-n represent any number of CA-Easytrieve/Plus statements.

Nesting

You can nest DO loops, that is, any of the statements within your loop can be a DO. The statement must satisfy conditions within inner loops before the program proceeds to an outer loop.
Example

In the exhibit below, each continuous line represents an entire DO loop:

```
DEFINE COUNT-1 W 3 N VALUE 0
DEFINE COUNT-2 W 3 N VALUE 0
DEFINE RESULT W 3 N VALUE 0
*
JOB INPUT NULL NAME MYPROG
DO WHILE COUNT-1 LT 10
  COUNT-1 = COUNT-1 + 1
  COUNT-2 = 0
  DO WHILE COUNT-2 < 10
    COUNT-2 = COUNT-2 + 1
    RESULT = COUNT-1 * COUNT-2
    DISPLAY 'COUNT-1= ' COUNT-1 '  COUNT-2= ' COUNT-2 '  RESULT= ' RESULT
  END-DO
END-DO
END-DO
STOP
```

Proper Nesting (lines do not cross)

```
DO WHILE
  DO WHILE
    END-DO
END-DO
```

Improper Nesting (lines cross)

```
DO WHILE
  END-DO
  DO WHILE
    END-DO
```

Refer to Chapter 13, “Table and Array Processing,” for more examples of loop control processing.
CASE Statement

The CASE statement conditionally executes one of several groups of statements based on the value of a specified field. The syntax and logic are:

Syntax

```
CASE   field-name

WHEN compare-literal-1 [THRU range-literal-1] [...]  
statement-1

WHEN compare-literal-n [THRU range-literal-n] [...]  
statement-n

[OTHERWISE   ]  
[  statement-n+1]

END-CASE
```

Operation

field-name

Field-name is the subject of the evaluation. The contents of field-name are compared to the values represented by literal-1 and literal-2. Field-name can be a field of any type but cannot be a varying length alphanumeric field. If field-name is alphanumeric, it must be 254 or fewer bytes in length. If field-name is numeric, it must have zero or no decimal places.
**CASE Statement**

**WHEN**

You specify one or more WHEN conditions. You cannot code any statements between the CASE and the first WHEN. Each WHEN must supply a unique set of values to be compared with field-name.

`compare-literal [THRU range-literal] ...`

Compare-literal is the value to be compared with field-name. You can specify a single literal, a series of literals, or a range of literals. A range is represented by compare-literal THRU range-literal. A range is satisfied when field-name is greater than or equal to lesser of compare-literal and range-literal and is less than or equal to the greater of compare-literal and range-literal.

When field-name is alphanumeric, compare-literal and range-literal must also be alphanumeric and must be equal in length to field-name. When field-name is numeric, compare-literal and range-literal must also be numeric and must not have any decimal places. Numeric literals must be equal in length to field-name.

The set of literal values specified for a given WHEN, including the unspecified values implied by a range, must be unique as compared to the literal values of any other WHEN for the same CASE.

`statement-1 
statement-n`

Statement-1 and statement-n represent any number of CA-Easytrieve/Plus statements executed when the WHEN comparison is satisfied. Whenever one or more of these statements is a CASE statement, the CASE statements are considered to be nested.

**OTHERWISE**

OTHERWISE is an optional statement that specifies a group of statements to be executed if no WHEN comparison was satisfied. If OTHERWISE is not specified and field-name does not equal any of the specified WHENs, execution continues with the statement following END-CASE.

`statement-n+1`

Statement-n+1 represents any number of CA-Easytrieve/Plus statements executed when no WHEN comparisons are equal. Whenever one or more of these statements is a CASE statement, the CASE statements are considered to be nested.

**END-CASE**

END-CASE terminates the body of the CASE. An END-CASE must be specified after each CASE and its associated statements.
Example

The following exhibit illustrates CASE statement usage. In it, the field XMAS-BONUS is computed to be a certain percent over PAY-GROSS.

FILE PERSNL
NAME-LAST 17 8 A
PAY-GROSS 94 4 P 2
WORK-GROSS W 4 P 0
XMAS-BONUS    W 4 P 2 VALUE 0
TOT-XMAS-BONUS W 6 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG FINISH FINISH-PROC
WORK-GROSS = PAY-GROSS
CASE WORK-GROSS
  WHEN 0 THRU 1000
    XMAS-BONUS = PAY-GROSS * 1.03
  WHEN 1001 THRU 5000
    XMAS-BONUS = PAY-GROSS * 1.05
  OTHERWISE
    XMAS-BONUS = PAY-GROSS * 1.10
END-CASE
TOT-XMAS-BONUS = TOT-XMAS-BONUS + XMAS-BONUS
PRINT MYREPORT
*
FINISH-PROC. PROC
DISPLAY
DISPLAY 'TOTAL $ SPENT IN BONUS ' +
      ' MONEY ====> ' TOT-XMAS-BONUS
END-PROC
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS

GOTO Statement

The GOTO statement allows you to modify the natural top to bottom logic flow of statement execution. Its syntax is:

Syntax

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

Operation

{label}

GOTO immediately transfers execution control to the first statement following the associated label. Processing then continues in a top-to-bottom sequence. The label must be contained in the same activity or procedure. Statement labels are discussed in detail later in this chapter.
GOTO JOB causes an immediate branch to the top of the current JOB activity.

Example

The following exhibit illustrates the use of GOTO in a program. The arrows indicate that control is passed to the first executable statement of the activity or procedure.

```
FILE PERSNL  FB(150 1800)
%PERSNL
XMAS-BONUS  U 4 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG
→ IF PAY-GROSS < 300.00
    GOTO SPECIAL-BONUS
ELSE
    XMAS-BONUS = PAY-GROSS * 1.03
END-IF
PRINT-LABEL
→ PRINT MYREPORT
    GOTO JOB
*
SPECIAL-BONUS
    XMAS-BONUS = PAY-GROSS * 1.10
GOTO PRINT-LABEL
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS
```

Statement Label

Use the statement label with the GOTO statement. When a GOTO is executed, execution transfers to the first statement following the statement label named on the GOTO statement. The syntax of a statement label is:

Syntax

```
GOTO label
...
...
label
...
```
Operation

label

Label can be up to 40 alphanumeric characters long, can contain any character other than a delimiter, and begin with A-Z or 0-9; it cannot consist of all numeric characters.

A statement label is a complete CA-Easytrieve/Plus statement that you can code prior to the following statements:

<table>
<thead>
<tr>
<th>Assignment Statement</th>
<th>Statement label</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>STOP</td>
</tr>
<tr>
<td>CASE</td>
<td>WRITE</td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td></td>
</tr>
<tr>
<td>DLI</td>
<td></td>
</tr>
<tr>
<td>DO UNTIL</td>
<td></td>
</tr>
<tr>
<td>DO WHILE</td>
<td></td>
</tr>
<tr>
<td>ELSE-IF</td>
<td></td>
</tr>
<tr>
<td>END-DO</td>
<td></td>
</tr>
<tr>
<td>END-IF</td>
<td></td>
</tr>
<tr>
<td>END-PROC</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td></td>
</tr>
<tr>
<td>GOTO</td>
<td></td>
</tr>
<tr>
<td>IDMSIF</td>
<td></td>
</tr>
<tr>
<td>MOVE</td>
<td></td>
</tr>
<tr>
<td>MOVE</td>
<td></td>
</tr>
<tr>
<td>LIKE</td>
<td></td>
</tr>
<tr>
<td>PERFORM</td>
<td></td>
</tr>
<tr>
<td>POINT</td>
<td></td>
</tr>
<tr>
<td>PRINT</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td></td>
</tr>
<tr>
<td>SEARCH</td>
<td></td>
</tr>
<tr>
<td>SELECT</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td></td>
</tr>
</tbody>
</table>
Example

The following exhibit illustrates the use of a statement label with the GOTO statement:

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
XMAS-BONUS W 4 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG
  IF PAY-NET < 100.99
    GOTO MAJOR-INC
  ELSE
    GOTO REGULAR-INC
  END-IF
*
MAJOR-INC
  XMAS-BONUS = PAY-GROSS * 1.20
  PRINT MYREPORT
  GO TO JOB
*
REGULAR-INC
  XMAS-BONUS = PAY-GROSS * 1.05
  PRINT MYREPORT
  GO TO JOB
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS
```
PERFORM Statement

PERFORM transfers control to a procedure and, after the procedure has been executed, returns control to the next executable statement after the PERFORM statement. Its syntax is:

Syntax

PERFORM proc-name

Operation

proc-name

Proc-name is the name of the procedure to be executed.

When CA-Easytrieve/Plus encounters the PERFORM statement, it immediately branches to the named routine. After exiting from the procedure, execution continues with the statement following the PERFORM statement.

Example

The following exhibit illustrates the use of the PERFORM statement in executing a user procedure:

FILE PERSNL FB(150 1800)
%PERSNL
XMAS-BONUS W 4 P 2 VALUE 0
*
JOB INPUT PERSNL NAME MYPROG
  IF PAY-GROSS < 300.99
    PERFORM SPECIAL-BONUS
  ELSE
    PERFORM STANDARD-BONUS
  END-IF
  PRINT MYREPORT
*
SPECIAL-BONUS. PROC
  XMAS-BONUS = PAY-GROSS * 1.20
END-PROC
*
STANDARD-BONUS. PROC
  XMAS-BONUS = PAY-GROSS * 1.05
END-PROC
*
REPORT MYREPORT
LINE NAME-LAST XMAS-BONUS

Refer to Chapter 10, “Procedure Processing,” for more information on processing procedures using the PERFORM statement.
The STOP statement terminates JOB and SORT activities. In CA-Easytrieve/Plus, activities with automatic file input automatically terminate when all input records have been processed. You can terminate activities prematurely, however, with a STOP statement. You must use STOP to terminate JOB activities without automatic file input (for example, JOB INPUT NULL). The STOP statement syntax is:

**Syntax**

```
STOP  [EXECUTE]
```

**Operation**

[EXECUTE]

The EXECUTE option immediately terminates all CA-Easytrieve/Plus execution. Spooled data or data in buffers is not printed. The default terminates the current activity only.

**Example**

The following exhibit illustrates STOP in a SORT activity to limit the number of records being sorted. In this example, only the first 50 records from PERSNL are sorted since the STOP statement simulates end-of-file on PERSNL.

```
FILE PERSNL  FB(150 1800)
&PERSNL
FILE SORTOUT FB(150 1800)
COPY PERSNL
*
SORT PERSNL TO SORTOUT  +
   USING (PAY-GROSS D)  +
   NAME MYSORT BEFORE SORT1-PROC
*
SORT1-PROC. PROC
   IF PERSNL:RECORD-COUNT GT 50
      STOP
   ELSE
      SELECT
      END-IF
   END-PROC
```

**Conditional Execution**

The normal flow of CA-Easytrieve/Plus executes each activity in top-to-bottom order. Whenever you want to alter that flow and effectively inhibit activity execution, you can conditionally execute the STOP statement in JOB START proc-name procedures.
Example

The following exhibit illustrates conditional activity execution:

FILE PERSNL FB(150 1800)
%PERSNL
TOTAL-GROSS W 10 P 2 VALUE 0
JOB-CODE W 4 A VALUE '    '
*
JOB INPUT PERSNL NAME MYPROG1 FINISH FINISH-PROC
  TOTAL-GROSS = TOTAL-GROSS + PAY-GROSS
*
FINISH-PROC. PROC
  IF TOTAL-GROSS > 50000.00
    JOB-CODE = 'JOB2'
  ELSE
    JOB-CODE = 'JOB3'
  END-IF
  DISPLAY 'JOB CODE= ' JOB-CODE
END-PROC
*
JOB INPUT NULL START (START-JOB2) NAME MYPROG3
  DISPLAY 'JOB2 PROCESSING'
STOP
*
START-JOB2. PROC
  IF JOB-CODE NE 'JOB2'
    STOP
  END-IF
END-PROC
*
JOB INPUT NULL START (START-JOB3) NAME MYPROG4
  DISPLAY 'JOB3 PROCESSING'
STOP
*
START-JOB3. PROC
  IF JOB-CODE NE 'JOB3'
    STOP
  END-IF
END-PROC

Termination

Under certain circumstances, you might want to completely terminate all activities.
Example

The following exhibit illustrates termination of activities:

FILE INVENT FB(200 3200)
%INVMSTR
FILE SORTWRK FB(200 3200)
COPY INVENT *

JOB INPUT INVENT NAME MYPROG1 FINISH FINISH-PROC
   PRINT MYREPORT *

FINISH-PROC. PROC
   IF RECORD-COUNT = 0
       DISPLAY 'INPUT FILE NOT AVAILABLE'
       DISPLAY 'HALTING EXECUTION...'
       STOP EXECUTE
   END-IF
END-PROC *

REPORT MYREPORT
LINE PART-NUMBER PART-DESCRIPTION *

SORT INVENT TO SORTWRK USING +
   (VENDOR-LOCATION-STATE, +
    VENDOR-LOCATION-CITY) NAME MYSORT *

JOB INPUT SORTWRK NAME MYPROG2
   PRINT MYREPORT *

REPORT MYREPORT
LINE PART-NUMBER VENDOR-LOCATION-CITY +
   VENDOR-LOCATION-STATE
Chapter 9 presents the CA-Easytrieve/Plus statements used for controlled input and output processing. These statements are:

- DISPLAY
- GET
- POINT
- PRINT
- PUT
- READ
- WRITE

These statements cause an input or output action to occur.

### STATUS Parameter

The STATUS parameter can be optionally specified on the GET, POINT, and READ statements for ISAM or VSAM files and also on the PUT and WRITE statements for VSAM files.

Use the STATUS option whenever the possibility exists for an unsatisfactory completion of the input/output request. Code STATUS to check the status of the input/output processing, that is, to see if it was performed properly. Then you can perform an appropriate action based on that status. STATUS causes the file's FILE-STATUS field to be set with the return code from the operating system's data management control program. You can determine the meaning of the contents of FILE-STATUS by referring to the appropriate IBM manual; however, normally a 0 or nonzero test is sufficient. If you do not code STATUS and the operating system returns a nonzero status, CA-Easytrieve/Plus issues an appropriate diagnostic message.

Refer to Chapter 12, “File Processing,” for a more extensive explanation of automatic and controlled file processing.
DISPLAY Statement

VSAM FILE-STATUS Codes

<table>
<thead>
<tr>
<th>FILE-STATUS Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Operation performed successfully</td>
</tr>
<tr>
<td>4</td>
<td>End-of-file during GET</td>
</tr>
<tr>
<td>8</td>
<td>(a) Duplicate key for a record being output with PUT or WRITE</td>
</tr>
<tr>
<td></td>
<td>(b) Additional records with the same key exist in the alternate index during a GET operation</td>
</tr>
<tr>
<td>12</td>
<td>Keys not in sequence during PUT operation</td>
</tr>
<tr>
<td>16</td>
<td>Record not found during READ operation</td>
</tr>
</tbody>
</table>

DISPLAY Statement

DISPLAY formats and transfers data to the system output device or to a named file. You can code DISPLAY to transfer printed data to the system output device (SYSPRINT/SYSLST). You can optionally code a file-name after DISPLAY to cause data to be printed to the named file. The DISPLAY syntax has three formats:

Syntax

Format 1

```
DISPLAY [file-name-1] [NEWPAGE] [+integer-2] ... [SYSPRINT] [SKIP integer-1] [-integer-2]
```

Format 2

```
DISPLAY [file-name-1] [NEWPAGE] [field-name-1] [field-name-2]
```
Format 3

```
DISPLAY [file-name-1] [CONTROL literal-3]
[SYSPRINT ]
[ ]
```

Format 1

```
[ ]
[file-name-1]
[SYS PRINT ]
[ ]
```

When you specify file-name-1, CA-Easytrieve/Plus prints data to the named file. If you do not specify file-name-1, the default is SYS PRINT/SYSLST.

**Note:** The named file (file-name-1) should be a PRINTER file or unpredictable results could occur.

[NEWPAGE]

The NEWPAGE option specifies that a skip to a new page occurs before the data is printed.

[SKIP integer-1]

The SKIP option specifies that integer-1 number of lines are skipped before the data is printed. When integer-1 is zero (0), the current line being DISPLAYed overlays the previous line output to file-name-1.

[CONTROL literal-1]

The CONTROL option sets the print carriage control character for the print line. Valid alphanumeric values for literal-1 are 0 through 9, +, -, A, B, or C (not valid for use in report procedures). When file-name-1 is associated with an extended reporting printer, the printer must support ANSI or Machine carriage controls. (See the *CA-Easytrieve/Plus Extended Reporting Facility Guide* for more information about extended reporting.)

```
[literal-2 ]
[field-name-1]
[ ]
```

Code literal-2 or field-name-1 in the order you want them to appear on the printed line. Note that if literal-2 or field-name-1 contain DBCS format data, the output device that you code after the DISPLAY keyword must be associated with an extended reporting printer (see Chapter 5, “Data Definition: FILE Statement”). This applies to both MIXED and DBCS fields and literals. The entries are invalid if SYSPRINT or file-name-1 are not associated with an extended reporting printer that supports DBCS data.
In addition, the extended reporting options module defines the DBCS code system that the printer supports. Should the code system of any of the MIXED/DBCS fields or literals coded on the DISPLAY statement not match the system code of the printer, the DBCS data in the field/literal is converted into the code system supported by that printer. Their data format does not change. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

The space adjustment option +integer-2 or -integer-2 modifies the horizontal spacing between display items. Integer-2 is any amount that does not extend beyond the end of the line.

The COL integer-3 option specifies the print column number where CA-Easytrieve/Plus places the next display item. Integer-3 is any value that does not extend beyond the end of the line.

When using an extended reporting printer that is not a standard line printer, an error results if two or more fields and/or literals overlap. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

**Note:** When coding a DISPLAY statement with POS or COL, be sure to include the filename after the POS or COL option.

For example, **Display COL 1 filename**

The POS integer-4 option on DISPLAY statements within report procedures causes the next display item to be positioned under the corresponding integer-4 item on the LINE 01 statement.

When using an extended reporting printer that is not a standard line printer, an error results if two or more fields and/or literals overlap. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

**Note:** When coding a DISPLAY statement with POS or COL, be sure to include the filename after the POS or COL option.

For example, **Display POS 4 filename**
The font index option identifies the font that CA-Easytrieve/Plus uses for the next display item. You can only specify this option if file-name-1 has been associated with an extended reporting printer. Integer-5 identifies the font number of a font defined for the extended reporting printer assigned to receive the print output. If you do not code the font index, then the next display item uses the default font for the assigned extended reporting printer.

If you code space adjustment or COL options before the font index option, integer-2 or integer-3 refers to horizontal sizes based on the default width of the assigned extended reporting printer. If you code the font index before either of these options, then integer-2 or integer-3 refers to horizontal sizes based on the width of the font identified by integer-5. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.

**Format 2**

In Format 2, CA-Easytrieve/Plus produces a hexadecimal and character dump of the current record of file-name-2 or of the specified field-name. The parameters, other than HEX, operate the same as in Format 1. For IMS/DLI and CA-IDMS files, record-name refers to any record (segment), while file-name-2 refers to all records (segments). DISPLAY HEX file-name-2 cannot be used in REPORT procedures.

**DISPLAY HEX**

Unless you specify relative or absolute positioning, the first data entry of each DISPLAY statement begins in column 1 of the print line. Each data entry that follows is printed next to a preceding entry. For HEX displays, the output is five lines per 100 bytes of the record or field.

**Double Byte Characters**

Data entries coded on the DISPLAY statement containing DBCS format data must be associated with an extended reporting printer that supports DBCS data via the SYSPRINT parameter or file-name-1. You must use the EXTENDED parameter of the FILE statement to associate the file with an extended reporting printer. (See Chapter 5, “Data Definition.”) This applies to both MIXED and DBCS fields and literals. DBCS and MIXED format entries are invalid if SYSPRINT or file-name-1 do not use this option, or if the printer does not support DBCS data.
If the DBCS Option is active and SYSPRINT or file-name-1 support DBCS format data, then the DISPLAY HEX command produces the five line hexadecimal display in Double Byte format. The character representation of the contents of the specified record or field is displayed in the first line of the five-line display. CA-Easytrieve/Plus converts all EBCDIC characters in the record or field to their Double Byte equivalent. CA-Easytrieve/Plus converts any DBCS characters in the field to the DBCS code system of the assigned extended reporting printer prior to being formatted onto the first line of the display.

CA-Easytrieve/Plus identifies EBCDIC and DBCS components in a field through one of two methods:

1. Examines the manner in which the field was defined in the CA-Easytrieve/Plus Data Definition or Library section (see Chapter 5, “Data Definition,” for details). If you redefined the field using the field-name-2 option of the DEFINE statement, the data format of these redefining fields is used to identify the EBCDIC and DBCS components. If you do not redefine the field in this manner, CA-Easytrieve/Plus uses method 2.

2. Examines all of your CA-Easytrieve/Plus program fields that redefine the area occupied by field-name-1. CA-Easytrieve/Plus uses the data format of these redefining fields to identify the EBCDIC and DBCS components of field-name-1. If there are no fields redefining the same area that field-name-1 occupies, then the format of field-name-1 is used to determine the format of the entire field.

If you code file-name or record-name on the DISPLAY HEX statement, CA-Easytrieve/Plus uses all fields defined for that file or record to identify the EBCDIC and DBCS components of the record. CA-Easytrieve/Plus then formats these components onto the first line of the hexadecimal display described earlier.

When you use DISPLAY in report procedures, output is always in the appropriate place in that report. However, when you use DISPLAY within a JOB activity, the DISPLAYed output can be interspersed with the first unSEQUENCED report (see Chapter 11, “Report Processing”).

**Format 3**

Format 3 pertains to you only if you are using the Extended Reporting Facility of CA-Easytrieve/Plus.

Some printing systems support print control records which are used to alter the printing environment. The updated environment, as altered by the print control record, is used by the printing system to format subsequent CA-Easytrieve/Plus output. Use Format 3 to output print control records.

Literal-3 can be an alphanumeric or hexadecimal literal that CA-Easytrieve/Plus outputs to the print file without paper control information.
You can only use this format of the DISPLAY statement when file-name-1 is associated with an extended reporting printer. CA-Easytrieve/Plus outputs a syntax error if the file-name-1 is not an extended reporting printer. See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.

Example

The following two exhibits illustrate the use of DISPLAY:

Statements:

FILE BADKEYS FB(150 1800) PRINTER
FILE PERSNL VS
SPERSNL
FILE INKEYS CARD
WHO * 5 N
JOB INPUT INKEYS NAME MYPROG
   READ PERSNL KEY WHO
      IF NOT PERSNL
         DISPLAY BADKEYS 'BAD KEY =' +1 WHO
         GOTO JOB
      END-IF
   DISPLAY SKIP 2 HEX PERSNL
END

01963

GET Statement

GET places the next sequential record of the named file into the file input area. Its syntax is:

Syntax

GET file-name [STATUS]
**Operation**

file-name

File-name identifies the input file. If file-name is an SQL file, see the appropriate database interface manual (*CA-Easytrieve SQL Interface Option Guide* or *ORACLE*).

[STATUS]

For information on the STATUS option, refer to STATUS Parameter at the beginning of this chapter.

**Example**

The following exhibit illustrates the use of GET:

```plaintext
FILE PERSNL VS
%PERSNL
JOB INPUT NULL NAME MYPROG
  GET PERSNL STATUS
  IF PERSNL:FILE-STATUS NE 0
    DISPLAY PERSNL:FILE-STATUS
  ELSE
    DISPLAY HEX PERSNL
  END-IF
STOP
```

Refer to Chapter 12, “File Processing,” for more detailed examples on the use of GET in file processing.

**POINT Statement**

The POINT statement enables you to establish the position within an ISAM, VSAM relative record (RRDS), or VSAM indexed (KSDS) file from which subsequent data is sequentially retrieved. Data in the file become available only after the next successful sequential retrieval by either automatic file input or a GET statement. The syntax of the POINT statement is:

**Syntax**

```
{= }
{EQ} {          }
POINT  file-name  {GE} {field-name} [STATUS]
{GQ} {literal   }
{>=}
```
**Operation**

file-name

File-name must be the same as on a FILE statement that describes an indexed, keyed, or relative-record file.

\{= \}  
\{(EQ\}  
\{(GE\}  
\{(GQ\}  
\{(\geq\}

Equal operators (\(=\) and EQ) initiate a file position search based upon an exact match between the file's keys and the search value. The 'greater than' operators (GE, GQ, and \(\geq\)) initiate a file position search based upon a file's key being equal to or greater than the search value.

\{ \}
\{field-name\}
\{literal\}

The search value can be any valid field-name or literal.

**Note:** Alphanumeric (EBCDIC, MIXED, or DBCS) literals must be enclosed within apostrophes as shown in the exhibit below.

CA-Easytrieve/Plus converts any DBCS data that field-name or literal contains into the DBCS code system of file-name. CA-Easytrieve/Plus does not change the data format of field-name or literal. The search value must have the same length as the key for the file.

[STATUS]

For information on the STATUS option, refer to STATUS Parameter at the beginning of this chapter. In addition to FILE-STATUS, IF EOF file-name is true for VSAM when the search value is greater than the highest key in the file.
Example

The following exhibit illustrates the use of POINT:

FILE PERSNL VS
%PERSNL
JOB INPUT NULL NAME MYPROG
  POINT PERSNL GE '01963' STATUS
  IF FILE-STATUS NE 0 OR EOF PERSNL
    DISPLAY 'BAD POINT...FILE STATUS= ' FILE-STATUS
    STOP
  END-IF
  GET PERSNL STATUS
  IF FILE-STATUS NE 0
    DISPLAY 'BAD GET...FILE STATUS= ' FILE-STATUS
  ELSE
    DISPLAY HEX PERSNL
  END-IF
STOP

Refer to Chapter 12, “File Processing,” for more detailed examples on the use of POINT in file processing.

PRINT Statement

The PRINT statement requests report output. Issue the PRINT statement to initiate a printed line. Its syntax is:

Syntax

PRINT [report-name]

Operation

[report-name]

Report-name is the name of the report as specified on a REPORT statement. If not given, it is assumed to be the first (un-named) report in the JOB activity.

In general, report output is not written directly to a report's printer file as with DISPLAY, but is scheduled for deferred formatting and writing to the report's printer file, perhaps following re-sequencing of an intermediate file.

Refer to Chapter 11, “Report Processing,” for detailed examples on the use of PRINT in report processing.
When you require an intermediate file (referred to as a report work file) for a report, executing PRINT causes fixed format records (called spool records) to be output to the work file. CA-Easytrieve/Plus determines the format of these records, but it includes all the fields required to produce the report except those in S-type working storage. When the report is to contain DBCS format data, CA-Easytrieve/Plus converts all the MIXED/DBCS fields into the code system defined as the Processing DBCS Code system.

For more information regarding the Processing code system, see the DBCS Options module description in the CA-Easytrieve/Plus Installation Guide and also the DBCSCODE keyword of the PARM statement.

This conversion occurs upon execution of the PRINT statement. Therefore, the spool record is built so that all of the MIXED/DBCS fields in that spool record belong to the same DBCS code system: the Processing DBCS Code system.

The exception to this assumption occurs when you specify the FILE option of the REPORT statement. The DBCS code system assigned to this FILE is the DBCS code system of the report work file. Upon execution of the PRINT statement CA-Easytrieve/Plus converts the MIXED/DBCS fields contained on the spool record to this DBCS code system. If the DBCS code system of the report work file is not the same as the DBCS code system of the extended reporting printer assigned to report-name, then CA-Easytrieve/Plus converts work file records to the DBCS code system applicable to report-name as the report is actually printed.

When you do not require a report work file and the report is output to the printer file immediately, CA-Easytrieve/Plus executes a PRINT statement. The conversion of the MIXED/DBCS fields still occurs. CA-Easytrieve/Plus performs the conversion before the data is moved into the print line(s).

PUT Statement

PUT performs sequential file output. PUT outputs records to SAM, VSAM, HOSTDISK, and VFM files and also adds consecutive records (mass sequential insertion) to a VSAM file. The syntax of the PUT statement is:

Syntax

```plaintext
PUT file-name-1 [FROM {file-name-2}] [STATUS]
[   {record-name}]
```
READ Statement

**Operation**

file-name-1

The file-name-1 parameter identifies the output file.

[  {          }]
[FROM  {file-name-2}]
[     {record-name}]

The FROM option identifies the file or record from which the current record is copied onto file-name-1. Using this parameter is equivalent to coding a MOVE statement prior to a PUT statement without the FROM option. When the length of file-name-2 or record-name is less than the length of file-name-1, uninitialized storage can be moved.

The length of the output data is taken from file-name-1:RECORD-LENGTH. For more information about RECORD-LENGTH, see Chapter 12, “File Processing.”

[STATUS]

For information on the STATUS option, refer to STATUS Parameter at the beginning of this chapter.

Refer to Chapter 12, “File Processing,” for detailed examples of PUT statement usage.

**READ Statement**

READ provides for random access to keyed and relative-record VSAM and ISAM files. Its syntax is:

**Syntax**

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

{literal}
```

**Operation**

file-name

The file-name parameter identifies the VSAM or ISAM file to be randomly accessed.
The KEY field-name parameter provides the key to the desired record. CA-Easytrieve/Plus uses the contents of field-name or 'literal' in a search for a corresponding record in the file.

If field-name contains DBCS data, CA-Easytrieve/Plus converts the DBCS data into the code system of file-name before performing the READ. CA-Easytrieve/Plus does not change the data format.

[STATUS]

The STATUS parameter is useful when randomly retrieving records from a VSAM data set. The exhibit below describes how to test whether a record is present in the file. For more information on the STATUS option, refer to the STATUS parameter description at the beginning of this chapter.

Example

FILE PERSNL VS
%PERSNL
FILE INKEYS CARD
WHO * 5 N
TOTAL-NET  W 6 P 2 VALUE 0
JOB INPUT INKEYS NAME MYPROG FINISH DISPLAY-TOTAL
  READ PERSNL KEY WHO STATUS
  IF PERSNL:FILE-STATUS NE 0
    DISPLAY 'UNSUCCESSFUL READ +
    PERFORMED ON FILE PERSNL' +
    +2 'KEY= ' WHO
  ELSE
    TOTAL-NET = TOTAL-NET + PAY-NET
  END-IF
*
DISPLAY-TOTAL. PROC
  DISPLAY 'TOTAL NET SALARY FOR ' +
  WHO ' = ' TOTAL-NET
END-PROC
END
11710
01895

Refer to Chapter 12, “File Processing,” for detailed examples of READ statement usage.
WRITE Statement

WRITE is used in the maintenance of keyed, entry sequenced, and relative-record VSAM files. During random processing of these files, WRITE updates and deletes existing records and adds new records. With entry sequenced files, only UPDATE is permitted. Its syntax has two formats:

Syntax

Format 1

```
WRITE  file-name-1  [UPDATE]   [FROM  {file-name-2}]  [STATUS]
[ADD   ]   [      {record-name}]
```

Format 2

```
WRITE  file-name-1  DELETE    [STATUS]
```

Format 1

Format 1 of the WRITE statement updates an existing record or adds a new record to the file. When updating, which is the default, the updated record is the current active record for the file.

Format 2

Format 2 of the WRITE statement deletes the current active record for the file.

file-name-1

File-name-1 is the name of the keyed or relative record VSAM file to be modified. You must also have coded the UPDATE subparameter on the FILE statement for file-name-1.

[UPDATE]
[ADD ]
[DELETE]
[

UPDATE (default), ADD, or DELETE designates the type of file maintenance activity to be performed.
The FROM file-name-2 record-name option identifies an alternate data source for file UPDATE and ADD operations. Using this parameter is equivalent to coding a MOVE statement prior to a WRITE statement. The current value of file-name-1:RECORD-LENGTH is the length of the output data.

[STATUS]

For more information on the STATUS option, refer to STATUS Parameter at the beginning of this chapter.

Example

The following exhibit illustrates the use of WRITE:

FILE PERSNL VS(UPDATE)
%PERSNL
JOB INPUT NULL NAME MYPROG
READ PERSNL KEY '05807' STATUS
IF PERSNL:FILE-STATUS NE 0
    DISPLAY 'FILE-STATUS= ', PERSNL:FILE-STATUS
    DISPLAY 'UNSUCCESSFUL READ ON PERSNL FILE'
ELSE
    DISPLAY HEX PERSNL
    MOVE '3125059599' TO TELEPHONE
    WRITE PERSNL UPDATE
    IF PERSNL:FILE-STATUS NE 0
        DISPLAY 'FILE-STATUS= ', PERSNL:FILE-STATUS
        DISPLAY 'UNSUCCESSFUL UPDATE ON PERSNL FILE'
    END-IF
END-IF
STOP

Refer to Chapter 12, “File Processing,” for more detailed examples on the use of WRITE in file processing.
Introduction

CA-Easytrieve/Plus provides all of the functions necessary to perform basic input/output, data examination, and data manipulation. However, there can be occasions when you want to incorporate your own routines into the main program. CA-Easytrieve/Plus permits you to do this through the use of procedures.

Chapter 10 describes procedures and shows you how to invoke them in your program using:
- PERFORM statement
- START and FINISH parameters of the JOB statement
- BEFORE parameter of the SORT statement
- Special-name procedures at the end of a REPORT activity.

Chapter 10 also discusses placement of procedures within your program.

Procedure Syntax

A procedure is a group of user-written CA-Easytrieve/Plus statements designed to accomplish a particular objective. Procedures are identified with the PROC and END-PROC keywords. The syntax of a procedure has two formats as illustrated in the exhibit below:
Syntax

Format 1
proc-name. PROC
  statement-1
  ...
  statement-n
END-PROC

Format 2
proc-name
PROC
  statement-1
  ...
  statement-n
END-PROC

Operation

proc-name

Proc-name identifies the procedure. The name can be from 1 to 40 characters long, can contain any character other than a delimiter, and begin with A-Z or 0-9; it cannot consist of all numeric characters. Proc-name must be followed by the keyword PROC as a separate statement (see the exhibit above).

statement-1...n

Statement-1 through statement-n are the statements which accomplish your procedure's task. In most cases, you can code any statements within a procedure. However, you cannot code certain input/output statements (for example, GET, PUT) in procedures invoked during SORT or REPORT processing.

PERFORM statements within a procedure can invoke other procedures; this is called procedure nesting. However, recursion is not permitted. That is, procedure A can invoke procedure B, but procedure B cannot then invoke procedure A.

END-PROC

END-PROC delimits the statements contained in the procedure.
Invoking Procedures

You can invoke procedures at specific points during the execution of your program. When a procedure is invoked, the statements within the procedure are executed until control passes to an END-PROC statement. At that time, CA-Easytrieve/Plus returns control to the point in your program where you invoked the procedure.

You can invoke a procedure by:

- Coding PERFORM in the statements following a JOB statement or in the statements within any procedure.
- Specifying proc-names with the START and FINISH parameters of a JOB statement.
- Specifying a proc-name with the BEFORE parameter of a SORT statement.
- Coding special-name procedures at the end of a REPORT activity.

**Note:** You cannot invoke other procedures from special-name procedures.

PERFORM Statement

The PERFORM statement transfers control to a procedure. After the procedure is executed, control returns to the next executable statement following the PERFORM. The syntax is:

**Syntax**

PERFORM  proc-name

Refer to Chapter 8, “Decision and Branching Logic: Perform Statement,” for a detailed explanation and examples of using PERFORM to invoke procedures.
START and FINISH Parameters (JOB)

START proc-name-1 and FINISH proc-name-2 are optional parameters of the JOB statement that incorporate procedures into their processing activities.

START

START identifies a procedure to be executed during the initiation of a JOB activity. CA-Easytrieve/Plus performs the procedure coded in proc-name-1 before it retrieves the first automatic input record.

FINISH

FINISH identifies a procedure to be executed during the termination of a JOB activity. After CA-Easytrieve/Plus processes the last automatic input record, it performs the procedure identified by proc-name-2.

Refer to Chapter 6, “Processing Activities: JOB Statement,” for detailed examples on how to invoke a procedure with the START and FINISH parameters of the JOB statement.

BEFORE Parameter (SORT)

BEFORE proc-name

Sometimes you want to sort only certain records and/or change their format. To do this, you must write a sort procedure that immediately follows the SORT statement.

The optional BEFORE parameter of the SORT statement identifies a procedure that screens, modifies, or selects records for the sort. Proc-name indicates the PROC statement that identifies your SORT procedure.

Refer to Chapter 6, “Processing Activities: SORT Statement,” for detailed examples on the use of the BEFORE parameter to invoke a sort procedure.
Special-name Report Procedures

CA-Easytrieve/Plus automatically invokes certain special-name procedures as they occur within REPORT processing. These are:

- REPORT-INPUT
- BEFORE-LINE
- AFTER-LINE
- BEFORE-BREAK
- AFTER-BREAK
- ENDPAGE
- TERMINATION


Procedure Placement

You must code procedures immediately after their associated activity (JOB or SORT) or subactivity (REPORT), as illustrated in the exhibit below:

**JOB**

```
JOB statements
... START proc-name-1
... FINISH proc-name-2
... procedures PERFORMed by the above procedures and statements
```

**REPORT**

```
REPORT declarative statements
... special-name procedures
```

**SORT**

```
BEFORE proc-name
... procedures PERFORMed by the BEFORE proc
```
Example

The exhibit below illustrates the use and placement of procedures within a program:

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
FILE SORT1 FB(150 1800) VIRTUAL
COPY PERSNL
TOTAL-COMPANY-SALARY-WOMEN W 6 P 2
TOTAL-COMPANY-SALARY-MEN W 6 P 2
*
SORT PERSNL TO SORT1 USING (NAME-LAST) +
  NAME MYSORT1 BEFORE SCREENER
*
SCREENER. PROC
  IF SEX = 1 OR SEX = 2
    SELECT
  END-IF
END-PROC
*
JOB INPUT (SORT1) NAME MYPROG +
  START (START-PROC) FINISH (FINISH-PROC)
  PERFORM CALC-AND-PRINT
*
START-PROC. PROC
  MOVE ZEROS TO TOTAL-COMPANY-SALARY-WOMEN +
    TOTAL-COMPANY-SALARY-MEN
END-PROC
*
FINISH-PROC. PROC
  PRINT MYREPORT1
END-PROC
*
CALC-AND-PRINT. PROC
  IF SEX = 1
    TOTAL-COMPANY-SALARY-WOMEN = +
    TOTAL-COMPANY-SALARY-WOMEN + PAY-GROSS
  PRINT MYREPORT2
  END-IF
  IF SEX = 2
    TOTAL-COMPANY-SALARY-MEN = +
    TOTAL-COMPANY-SALARY-MEN + PAY-GROSS
  PRINT MYREPORT3
  END-IF
END-PROC
*
REPORT MYREPORT1
LINE TOTAL-COMPANY-SALARY-WOMEN TOTAL-COMPANY-SALARY-MEN
*
REPORT MYREPORT2
LINE NAME-LAST NAME-FIRST PAY-GROSS
*
ENDPAGE. PROC
  DISPLAY SKIP 2 'TOTAL = ' TOTAL-COMPANY-SALARY-WOMEN
END-PROC
*
REPORT MYREPORT3
LINE NAME-LAST NAME-FIRST PAY-GROSS
*
ENDPAGE. PROC
  DISPLAY SKIP 2 'TOTAL = ' TOTAL-COMPANY-SALARY-MEN
END-PROC
```
Introduction

A major function of many CA-Easytrieve/Plus programs is to produce printed reports. The non-procedural nature of CA-Easytrieve/Plus report syntax is readily adaptable to the production of basic and extremely complex reports, both with minimum programming effort.

Two statements generate printed output:

- The PRINT statement initiates the basic declarative report facility.
- The DISPLAY statement produces single print lines on print files.

PRINT is the preferred method because of its many automatic facilities. This chapter, therefore, discusses report processing using the PRINT statement. For a complete discussion of the DISPLAY statement, refer to Chapter 9, “Input/Output Specification.”
Basic Report Structure

The CA-Easytrieve/Plus report facility is basically declarative; you only need to define the format and content of the report and CA-Easytrieve/Plus creates the necessary instructions to produce the report. The exhibit below illustrates the basic structure of a CA-Easytrieve/Plus job with report processing; you can define one or more reports for each activity.

PRINT Statement Processing

The PRINT statement activates the report logic defined by REPORT declarations. CA-Easytrieve/Plus extracts the data required for the requested report, formats it in the specified manner, and outputs it to the printer.

The immediate result of a PRINT statement is either one of the following:

- Data output to a print file.
- Data output to a work file.
CA-Easytrieve/Plus automatically creates work file records when:

- The report is SEQUENCEd.
- Another report is already using the associated print file (this happens when you have multiple reports in a single JOB activity).

**PRINT Workfile Processing**

The termination process of each JOB activity, illustrated in the exhibit below, includes the processing of any print work files created during the JOB activity.
Report Formats

There are two basic report formats:
- Standard Format, called a report (see the exhibit below).
- Label Format, called a label report (see the exhibit below).

Standard Format

The CA-Easytrieve/Plus default report format is the standard format illustrated in the exhibit below:

![Diagram of Standard Format]

Top Margin

The top margin is the space between the physical top of the form and the point to which the printer positions 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 optional title area consists of 1 to 99 title lines plus a title margin between the last title line and the first heading line.
Heading Area

The optional heading area consists of 1 to 99 heading lines plus a heading margin between the last heading line and the report body.

Report Body

The report body consists of one or more line groups. Each line group is 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.

Label Format

The second report format is used to print labels. The exhibit below illustrates the basic label report page format:

```
  1   2   3   4
  5   6   7   8
```

A label line consists of one or more labels positioned across the label page. In the above exhibit, labels one through four compose a label line. A single line group composes each label. Therefore, CA-Easytrieve/Plus produces a label for each PRINT statement execution. CA-Easytrieve/Plus formats the labels on the page in the order illustrated in the above exhibit. DOWN and SIZE indicate the dimensions of each label.

DBCS Reporting Restrictions

CA-Easytrieve/Plus only supports DBCS format data in a report definition if the report is directed to a CA-Easytrieve/Plus printer file associated with an extended reporting option printer. For more details on this option, see the CA-Easytrieve/Plus Extended Reporting Facility Guide.
The extended reporting printer that has been associated with the printer file to which the report is being directed must also support DBCS output. This support is indicated in the Extended Printer Options module.

If the printer does not support a Double Byte Character Set, any MIXED or DBCS type fields or literals that you code on output-producing statements in the report (for example, TITLE, HEADING, and LINE statements) cannot be processed correctly.

This also applies to EBCDIC fields that have the DBCS mask option defined for them. This does not apply to those MIXED or DBCS fields that have the mask option of HEX.

These fields are output as a hexadecimal EBCDIC character string and therefore do not fall into the category of normal DBCS fields.

These restrictions apply to all printer files. If the report uses the CA-Easytrieve/Plus system output file (SYSPRINT/SYSLST) then you must associate that file with an extended reporting type printer that supports DBCS data.

If the report is directed to another printer file (with the PRINTER option on the REPORT statement) then you must associate that file with an extended reporting type file.

**Report Definition Statements**

Code a report definition by defining a REPORT statement followed by a series of Report Declaratives that define the content of the report.
Structure of Report Definition

A set of report definition statements defines every CA-Easytrieve/Plus report. The statements define the report type, format, sequence, and data content. Report definition statements are the last statements coded in a JOB activity. These statements must be coded in the order illustrated in the exhibit below. You can code report procedures in any order and can define any number of reports for each JOB activity.

REPORT Statement

Code the REPORT statement first in a report declaration. This statement establishes the type and characteristics of the report. Although you can specify a large number of REPORT statement parameters, most reports are produced by using default parameter values. REPORT statement parameters provide a simple way to define tailored reports. Appendix C, “Options Table,” contains a list of default parameter values.

REPORT statement parameters fall into five basic groups:

- Format determination parameters
- File directing parameters
- Spacing control parameters
- Testing aid parameters
- IMS – only parameters.
Syntax

The following exhibit shows the syntax of the REPORT statement.

```plaintext
REPORT [report-name] +

[SUMMARY] +   
[SUMFILE file-name-1] +
[SUMSPACE literal-1] +
[TALLYSIZE literal-2] +

[EVERY] +
[FIRST] +
(NONE) +

{} {
 {ALL }
}

{} {
 {HIAR}
{DTLCOPY}
}

{} {
 {TAG}
}

{} {
 }

{} {
 SPREAD
}

{} {
 NOADJUST
}

{} {
 NODATE/NOKDATE/LONGDATE/SHORTDATE
}

{} {
 NOPAGE/NOKPAGE
}

{} {
 NOHEADING
}

{} {
 LIMIT literal-12
}

{} {
 EVERY literal-13
}

{} {
 CHECKPOINT (literal-14 [USING (field-name ...)])
}
```

file directing parameters

format determination parameters

spacing control parameters

testing aid parameters

IMS-only parameter
REPORT Statement Parameters

Format Determination Parameters

[report-name]

The report-name parameter identifies the report. It is optional when there is only one report in a JOB. With multiple reports, the first report can remain unnamed but all others must be named. The report-name must be unique in each JOB and that report-name must be coded on at least one PRINT report-name statement. For unnamed reports, code the PRINT statement without a report-name parameter.

Report-name can be up to 40 characters long, can contain any character other than a delimiter, and can begin with A-Z or 0-9; it cannot consist of all numeric characters.

[SUMMARY]

On control reports, the SUMMARY option inhibits printing of detail data. Only control totals are printed. You cannot use the SUMMARY option in conjunction with the LABELS option.

[SUMFILE file-name-1]

The optional SUMFILE parameter generates a summary file. File-name-1 identifies the file that contains the data. The following files are illegal for use as summary files: DLI, IDMS, ISAM, SQL, and files used as input or output files in the job activity. For more information, refer to the Control Reports subject later in this chapter.

You can create a VSAM ESDS, KSDS, or RRDS SUMFILE. Be certain that the SUMFILE produced is compatible with your VSAM file attributes. SUMFILE records are PUT to a VSAM file. PUT adds consecutive records (mass sequential insertion) to a VSAM SUMFILE. For an ESDS SUMFILE, records are appended to the end of the file. For a KSDS SUMFILE, you must ensure that no SUMFILE record has a key duplicating an existing record’s key. Also, you must produce SUMFILE records in ascending key order. Keep in mind that a VSAM KSDS file has a maximum key length of 255 bytes. For an RRDS SUMFILE, slots are filled sequentially, starting with slot 1. You must ensure that all required slots are available.
The SUMSPACE option establishes the print size for total fields on a control report. Literal-1 is added to the length (in digits) of the field to establish its print size. This expansion is necessary to prevent the loss of significant data due to overflow conditions. The resulting print length is limited to a maximum of 18 digits. Valid values for literal-1 are 0 through 9. No additional numeric edit characters are included in the resulting edit mask. For example, totals such as 55555,555.55 can appear.

The TALLYSIZE option sets the print size for the field TALLY. Valid values for literal-2 are 1 through 18. The actual number of digits used for TALLY on a summary line are the summed value of TALLYSIZE and SUMSPACE.

Optional parameters, DTLCTL and SUMCTL, establish the method for printing control field values on control reports. DTLCTL establishes detail line printing characteristics and SUMCTL establishes total line printing characteristics.

Valid subparameters for DTLCTL are EVERY, FIRST, and NONE.

EVERY specifies that the value of all control fields is printed on every detail line.

FIRST specifies that the value of all control fields is printed on the first detail line of a page and on the first detail line after each control break. Control field values are not printed on all other detail lines.

NONE inhibits printing of control field values on detail lines.

Valid subparameters for SUMCTL are ALL, HIAR, NONE, TAG, DTLCOPY.

ALL causes control field values to be printed on every total line.

HIAR causes control field values to be printed in a hierarchical fashion on total lines. That is, only values for control fields on the same hierarchical level, or higher than the breaking control field, are printed on the associated total line.
[NONE]

NONE inhibits printing of control field values on total lines.

[TAG]

TAG causes control-field-name TOTAL to be printed as annotation to the left of the associated total line where control-field-name is the field-name for the breaking control field. There must be sufficient unused space on the left side of the total line for this annotation.

[DTLCOPY]

The DTLCOPY subparameter causes detail information to be printed on total lines. Normally, only control field values and associated totals are printed on total lines. Coding this subparameter causes the detail field contents, prior to the break, to be printed on the total line. These fields are printed only when LEVEL is one (1).

[DTLCOPYALL]

The DTLCOPYALL subparameter operates the same as the DTLCOPY subparameter except that the fields are printed for all control breaks.

[LABELS]

The LABELS option specifies that the report is a label report. The NOHEADING and NOADJUST options are automatically activated when the LABELS option is specified; therefore you cannot specify TITLE and HEADING statements. The subparameters ACROSS, DOWN, SIZE, and NEWPAGE control the spacing of label reports. You cannot use the LABELS option in conjunction with the SUMMARY option.

[ACROSS literal-3]

The ACROSS subparameter specifies the number of labels printed across the print line. Literal-3 is the number of labels printed side-by-side.

[DOWN literal-4]

The DOWN subparameter specifies the number of lines in a label. The value of literal-4 is the number of lines reserved for each label. The value range for literal-4 is 1 through 'nn', where 'nn' is at least as large as the largest corresponding 'LINE nn' value.

[SIZE literal-5]

The SIZE subparameter sets the length of each label. The value of literal-5 is the number of print positions on a label. Literal-5 has a value range from 1 to 'nnn', where 'nnn' is the length of the label.
If you direct the report to an extended reporting printer, CA-Easytrieve/Plus multiplies literal-5 by the default width defined for the assigned extended reporting printer. This result determines the width of each label. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

[NEWPAGE]

The NEWPAGE subparameter controls the printing of the first line (LINE 01) of each label. When coded, NEWPAGE associates a printer top-of-form request with the first line of each label.

The following algorithm constrains the overall size of labels:

\[
\text{LINESIZE} \geq (\text{ACROSS} - 1) \times \text{SIZE} + (\text{actual number of bytes printed on the last label})
\]

**File Directing Parameters**

[FILE file-name-2]

The optional FILE parameter identifies the work file used for very large reports. Code this parameter when the default VFM work file is too small to contain the report data. File-name-2 identifies the FILE that receives the work file data. The following file types are illegal for use as FILE parameter work files: DLI, IDMS, ISAM, SQL, HOSTDISK, and files being used in the JOB activity.

[PRINTER file-name-3]

The optional PRINTER parameter directs the report's printed output. File-name-3 identifies the FILE that receives the report. This file must have the PRINTER attribute specified. The default is the CA-Easytrieve/Plus standard print output file: SYSPRINT for OS/390, SYSLST for VSE.

If the system print output file or file-name-3 has been associated with an extended reporting printer, then CA-Easytrieve/Plus automatically formats the report to satisfy the requirements defined for that extended reporting printer. CA-Easytrieve/Plus restricts the support of extended reporting facilities to those reports that are output to printer files that have been associated with an extended reporting printer. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

**Spacing Control Parameters**

Each of the following parameters modifies the default spacing of a report page. You normally do not use these parameters; however, they are available to support unique report spacing requirements.
The PAGESIZE option establishes the length of each printed page. Literal-6a specifies the page length for LINE statements. Literal-6b specifies the page length for REPORT procedure DISPLAY statements.

The value of literal-6a sets the number of lines per page. Literal-6a has a valid range of 1 to 'nnn', where 'nnn' is at least as large as the sum of:

- 'nnn' of the last 'TITLE nnn'
- Literal-10 of TITLESKIP
- Number of HEADING lines plus one
- 'nnn' of the last 'LINE nnn'
- Literal-8 of SKIP.

In other words, at least one line group must fit on a report page.

If the report is directed to an extended reporting printer, and that printer does not support a Forms Control Block (FCB), then CA-Easytrieve/Plus multiplies the PAGESIZE value by the default height of the assigned extended reporting printer. This permits CA-Easytrieve/Plus to compare PAGESIZE with the heights of fonts used on the report. The value of literal-6 multiplied by the default height of the assigned extended reporting printer cannot exceed the maximum page length of that extended reporting printer. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

The value of literal-6b, the display-page-size, sets the number of lines per page for REPORT procedure DISPLAY statements. Literal-6b has a valid range from 0 to 32767. If literal-6b is zero, then a DISPLAY statement considers the report page infinite in size and never attempts to determine the end-of-page condition. That is, a DISPLAY statement does not compare literal-6b to the current line count.

However, if literal-6b is greater than zero, then a DISPLAY statement does compare the current line count to literal-6b. But only if the DISPLAY statement is in a REPORT-INPUT, BEFORE-LINE, AFTER-LINE, BEFORE-BREAK, or AFTER-BREAK report procedure. A DISPLAY statement in an ENDPAGE or a TERMINATION report procedure never attempts to determine the end-of-page condition, even when literal-6b is greater than zero.
A DISPLAY statement that determines the end-of-page condition, performs three actions:

1. It determines the end-of-page condition, by comparing the current line count to literal-6b, display-page-size.
2. If end-of-page is detected, then DISPLAY performs ENDPAGE and produces TITLEs and HEADINGS.
3. The DISPLAY formats and prints its own data.

If the current line count is greater than literal-6b, then the DISPLAY turns on the 'TITLEs requested' indicator, it does not print the TITLEs now. The actual printing of the TITLEs is left to the next LINE statement or the next DISPLAY statement that determines the end-of-page condition.

If literal-6b is not given, then the REPORT statement retrieves its value from the Options Table.

[LINESIZE literal-7]

The LINESIZE option determines the length of each line on a page. Literal-7 is the number of print positions on each report line. The value of literal-7 must be 1 to 'nnn', where 'nnn' is one less than the physical length (record size) of the printer file receiving the report. The first character in a PRINTER file contains the ASA carriage control information.

If the report is assigned to an extended reporting printer that is not a standard Line Printer, the maximum value of LINESIZE is not dependent upon the record size of the print data set. The insertion of Overprint and Function Codes into print records plus the support of different fonts on the same print line all impact the relationship between LINESIZE and the print data set record size. CA-Easytrieve/Plus supports any LINESIZE, provided literal-7 multiplied by the value of the assigned extended reporting printer's default width does not exceed the maximum page width of that extended reporting printer.

Literal-7 overrides the value defined in the CA-Easytrieve/Plus system's options module. If the report is directed to an extended reporting printer, CA-Easytrieve/Plus multiplies the LINESIZE value by the default width of the assigned extended reporting printer. This value defines the width of the print line. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

[SKIP literal-8]

The SKIP option determines the number of blank lines to be inserted between line groups (between the last 'LINE nnn' and the next LINE 01). Literal-8 has a valid range of 0 to 'nnn', where 'nnn' allows for the printing of at least one line group on each page. When you specify a value of 0, a line group containing multiple lines can be spanned across a page break. A non-zero value inhibits this spanning.
If the report is directed to an extended reporting printer that does not support a Forms Control Block (FCB), the default height of the assigned extended reporting printer defines the height of each line. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

[SPACE literal-9]

The SPACE option adjusts the default number of blanks (space characters) inserted between fields on TITLE and LINE statement items. The value of literal-9 has a valid range of 0 to 'nnn' (default is 3), where 'nnn' does not cause line overflow. The SPREAD option overrides this parameter.

If the report is directed to an extended reporting printer, CA-Easytrieve/Plus multiplies the SPACE factor by the default width of the assigned extended reporting printer. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

[TITLESKIP literal-10]

The TITLESKIP option inserts blank lines between the last title line and the first heading line (or LINE 01) of a report. The value of literal-10 has a valid range of 0 to 'nnn', where 'nnn' allows for the printing of at least one line group on each page.

If the report is directed to an extended reporting printer that does not support a Forms Control Block (FCB), the height of each line is defined by the default height of the assigned extended reporting printer. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

[CONTROLSKIP literal-11]

Specify CONTROLSKIP to define the number of blank lines to be inserted following CONTROL total lines and the next detail line. Literal-11 must be between 0 and 32767. If CONTROLSKIP is not specified, one blank line plus the SKIP value is inserted after the CONTROL total line.

[SPREAD ]
[NOADJ  ]

The SPREAD option requests that the maximum number of spaces be inserted between each column of a report. SPREAD overrides the SPACE parameter. NOSPREAD deactivates the SPREAD option when it is the default. SPREAD and NOADJUST are mutually exclusive. Refer to the SPREAD Option, later in this chapter, for more information and examples of this parameter.

[NOADJUST]

The NOADJUST option requests that the title lines and report be left-justified on the page. The default is for the report to be centered on the page. SPREAD and NOADJUST are mutually exclusive.
The parameters NODATE, NOKDATE, NOPAGE, LONGDATE, SHORTDATE, NOKPAGE and NOHEADING modify the information normally printed on the first TITLE line at the top of each report page.

[NODATE/NOKDATE/LONGDATE/SHORTDATE]

The NODATE option inhibits the printing of the SYSDATE value on the first title line (TITLE 01).

Normally, when a report is being generated for an extended reporting printer that supports DBCS data, the KANJI-DATE value is placed in positions one through 18 of the first title line (TITLE 01). You can inhibit this by coding the keyword NOKDATE on the REPORT statement.

When you use the NOKDATE option or the report is being generated for a printer that does not support DBCS data, CA-Easytrieve/Plus places the SYSDATE value in position one through eight of the first title line.

The LONGDATE option displays SYSDATE-LONG on the first title line.

The SHORTDATE option displays SYSDATE on the first title line.

If LONGDATE or SHORTDATE is not specified, the date to be displayed is determined from the Options Table.

[NOPAGE/NOKPAGE]

The NOPAGE option inhibits the printing of the PAGEWRD option and the current page number in the report title.

Normally, when a report is being generated for an extended reporting printer that supports DBCS data, CA-Easytrieve/Plus prints the Kanji page number at the end of the first title line. The Kanji page number consists of the current page count followed by the Kanji character for page. CA-Easytrieve/Plus edits the current page number and converts it into the appropriate DBCS value. This DBCS value is then combined with the Kanji page character and printed at the end of the first title line (TITLE 01). You can inhibit this by coding the keyword NOKPAGE on the REPORT statement.

When you use the NOKPAGE option or the report is being generated for a printer that does not support DBCS data, CA-Easytrieve/Plus prints the value of the PAGEWRD option and the current page number at the end of the first title line (TITLE 01).

[NOHEADING]

The NOHEADING option inhibits the printing of column headings. Normally, each field's HEADING value is printed as a column heading.
Testing Aid Parameters

The LIMIT and EVERY options are available as a testing aid for report development. These parameters control the amount of data output on a report.

[LIMIT literal-12]

The LIMIT option limits the number of records processed by the report. The value of literal-12 has a valid range of 1 to 32,767.

[EVERY literal-13]

The EVERY option specifies that only every Nth line is printed in the report. The value of literal-13 has a valid range of 1 to 32,767.

IMS-only Parameter

[CHECKPOINT (literal-14 [USING (field-name...))])]

CHECKPOINT is an IMS-only parameter for use with the extended Checkpoint/Restart Facility. For details see the CA-Easytrieve/Plus IMS/DLI Interface Option Guide.

DBCS/MIXED Considerations

When CA-Easytrieve/Plus cannot print a report directly as the PRINT statements are processed, it builds a report work file. Each record of the work file contains all the data required to produce the report at a later point in the CA-Easytrieve/Plus processing cycle. When MIXED/DBCS fields are specified as part of the report, it can be necessary to convert from one DBCS code system to another. The following situations define the conversion processing that is performed.

When you require a report work file and you do not code the FILE option of the REPORT statement, the code system of the report work file is the processing DBCS code system. CA-Easytrieve/Plus performs the conversion when the report work file record is built (that is as part of the PRINT statement operation). For more information regarding the Processing DBCS code system, see the DBCS Options module description in the CA-Easytrieve/Plus Installation Guide and also the DBCSCODE keyword of the PARM statement.
When you require a report work file and you have coded the FILE option of the REPORT statement, the code system of the report work file is that associated with the file-name specified to receive the report work file records. When the record is built, CA-Easytrieve/Plus processes any MIXED/DBCS fields that must be converted. In addition, when the report work file is then input for report printing, CA-Easytrieve/Plus converts all the MIXED/DBCS fields from the DBCS code system of the report work file to the DBCS code system defined for the extended reporting printer associated with the report.

Varying Length Field Considerations

The data window for fields with the VARYING option specified on the DEFINE statement is based on the maximum length of the field. The window is padded to the right with blanks for VARYING fields less than the maximum.

Report Definition Statements

The REPORT statement and its associated parameters define the physical attributes of your report. However, to define the content of your report, code the following statements:

SEQUENCE
CONTROL
SUM
TITLE
HEADING
LINE

You must code these statements (in the order shown above) immediately after the REPORT statement.

SEQUENCE Statement

The SEQUENCE statement optionally specifies the order of the report. You can order any report based on the content of one or more fields. The fields do not have to be part of the printed report. The SEQUENCE statement syntax is:

Syntax

SEQUENCE field-name-1 [D          ]            ...
          [IBM-sort-options]
          [JEF-sort-options]
**Operation**

field-name-1

The field-name identifies a field on which a report is ordered. You can specify multiple field-names. The fields must be in an active file or W type working storage. Each field must be less than 256 bytes. The fields specified are used as sort keys processed in major to minor order.

If there are no DBCS fields identified as sort keys, the system sort identified in the CA-Easytrieve/Plus options module is invoked. The rules for supporting DBCS and MIXED fields as sort keys are as follows:

1. All mixed fields are defined to the sort as alphanumeric MIXED fields.
2. CA-Easytrieve/Plus supports only the IBM and FACOM Kanji sorts. If the DBCS code system of the report work file is not IBM or JEF, then CA-Easytrieve/Plus uses the standard system sort and processes the DBCS fields as alphanumeric field types.
3. If the DBCS code system of the report work file is IBM or JEF but your system installation person has not identified the applicable Kanji sort in the DBCS Options module, then CA-Easytrieve/Plus passes any DBCS fields coded as sort keys to the system sort as alphanumeric field types.
4. When support for the IBM or FACOM sort is defined in the DBCS Options module, the respective sort-options are supported. You can only code them for DBCS fields. If you do not select a sort-option for a DBCS field, then CA-Easytrieve/Plus defines the field to the sort as an alphanumeric field. If there are no DBCS fields using any of the IBM or JEF sort-options, then CA-Easytrieve/Plus uses the normal system sort. That is, CA-Easytrieve/Plus only uses the applicable Kanji supportive sort when required.

**Note:** Varying length fields cannot be specified on a SEQUENCE statement.

An optional D following a field-name indicates that the field is sequenced into descending order. If you do not code D after a field-name, by default the field is sorted in ascending order.
An IBM-sort-option defines the Kanji sequence technique to be applied to the
DBCS field. For more specific information regarding the different techniques,
consult the IBM Program Description and Operation Manual for the Kanji/Chinese
Sort/Merge program product.

```
<table>
<thead>
<tr>
<th>IBM-sort-options</th>
</tr>
</thead>
<tbody>
<tr>
<td>field-name-1</td>
</tr>
<tr>
<td>[BUSHU [D]       ]</td>
</tr>
<tr>
<td>[SOKAKU [D]      ]</td>
</tr>
<tr>
<td>[KOKUGO field-name-2 [SMAP field-name-3] [D] [B] [I] ]</td>
</tr>
<tr>
<td>[          [DMAP field-name-4]         ]</td>
</tr>
<tr>
<td>[          [       ]                   ]</td>
</tr>
<tr>
<td>[DENWA field-name-2 [SMAP field-name-3] [D] [B] [I] ]</td>
</tr>
<tr>
<td>[          [DMAP field-name-4]         ]</td>
</tr>
<tr>
<td>[          [       ]                   ]</td>
</tr>
</tbody>
</table>
```

[ BUSHU [D] ]

This sort option invokes the Basic Radical Stroke-Count sequence for
field-name-1. Coding a D immediately after BUSHU causes the field to be sorted
in descending Radical Stroke-Count order. If you do not code the D, by default
the field is sorted in ascending order.

[ SOKAKU [D] ]

This sort option invokes the Basic Total Stroke-Count sequence for field-name-1.
Coding a D immediately after SOKAKU causes the field to be sorted in
descending Total Stroke-Count order. If you do not code the D, by default the
field is sorted in ascending order.

[ KOKUGO field-name-2 [SMAP field-name-3] [D] [B] [I] ]
[          [DMAP field-name-4]         ]
[          [       ]                   ]

This sort option invokes one of two different Kanji sorting techniques depending
upon whether or not you choose the SMAP or DMAP keyword. Both techniques
require you to specify a phonetic syllabary field – field-name-2. Field-name-2
must be either an alphanumeric (type A) or DBCS (type K) field type. If
field-name-2 is an alphanumeric field type, the field must contain the phonetic
reading of field-name-1 in Katakana. If field-name-2 contains DBCS format data,
it must contain the phonetic reading of field-name-1 represented by IBM DBCS
Hiragana or Katakana.
If you do not specify SMAP or DMAP then CA-Easytrieve/Plus applies the Japanese Dictionary sequencing technique to field-name-1. This technique sequences the records based on the phonetic reading of the WHOLE Kanji field.

If you select SMAP or DMAP, then CA-Easytrieve/Plus applies Kanji Index type processing for field-name-1. This technique orders the records based on the phonetic reading of each Kanji character. To do this you must supply a reading map so that the sort can relate phonetic syllabary characters to the appropriate Kanji characters. The reading map must be in the form of an alphanumeric field specified as field-name-3 or field-name-4. Field-name-3 specifies a field that contains a single-map. Field-name-4 specifies a field that contains a double-map. For more information on the definition and creation of these bit map fields, see the IBM Sort/Merge Program - Kanji/Chinese Manual.

Coding a D causes the field to be sorted in descending order. If the D is not coded, by default the field is sorted in ascending order. Coding a B causes CA-Easytrieve/Plus to apply the BUSHU (or Radical Stroke-Count) technique as a sub-sequence for those records that have the same phonetic reading. If you do not code B, by default the field is sub-sequenced using the SOKAKU (or Total Stroke-count). Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.

\[\text{DENWA field-name-2 \{ SMAP field-name-3 \} \{D\} \{B\} \{I\} } \]\n
\[\text{\{ DMAP field-name-4 \} } \]\n
The DENWA sort option invokes one of two different forms of the Japanese Telephone Directory sequencing technique. The form of sequencing technique that you use depends upon whether or not you specify the SMAP or DMAP keyword. The Telephone Directory method requires you to specify a phonetic syllabary field - field-name-2. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS format data, it must contain the phonetic reading of field-name-1 represented by IBM DBCS Hiragana or Katakana.

If you do not specify SMAP or DMAP, CA-Easytrieve/Plus applies the Simple form of the Japanese Telephone Directory order to field-name-1. This technique sequences the records based on the representative reading of the first Kanji character in the field. The representative reading of the first Kanji character is obtained from a special representative reading table based on the Kanji character itself and the voiceless sound of the first phonetic syllabary character in field-name-2.
If you specify SMAP or PMAP, CA-Easytrieve/Plus applies the All-Digit Japanese Telephone Directory sequence. This technique orders the records based on the representative reading of all the Kanji characters in the field. Sequencing is determined by applying representative reading to each Kanji character. For this, CA-Easytrieve/Plus requires a reading map field in the form of an alphanumeric field specified as field-name-3 or field-name-4. The reading map field identifies a bit map that enables the sort to relate phonetic syllabary characters to the appropriate Kanji characters.

Field-name-3 specifies a field that contains a single-map. Field-name-4 specifies a field defined in file-name-1. The field contains a double-map. For more information on the definition and creation of these bit map fields, see the *IBM Sort/Merge Program - Kanji/Chinese Manual*.

Coding a D causes the field to be sorted in descending order. If you do not code D, by default the field is sorted in ascending order. Coding a B causes CA-Easytrieve/Plus to apply the BUSHU (or Radical Stroke-Count) technique as a sub-sequence for those records that have the same phonetic reading. If you do not code B, by default the field is sub-sequenced using the SOKAKU (or Total Stroke-count). Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.

field-name-1 [JEF-sort-options] ...

A JEF-sort-option defines the Kanji sequence technique to be applied to the DBCS field. For more specific information regarding the different techniques, consult the *FACOM Sort/Merge Program Description Manual*.

<table>
<thead>
<tr>
<th>JEF-sort-options</th>
</tr>
</thead>
<tbody>
<tr>
<td>field-name-1</td>
</tr>
<tr>
<td>+</td>
</tr>
<tr>
<td>[ (BUSHU ) ]</td>
</tr>
<tr>
<td>[ ({SOKAKU} ...) ]</td>
</tr>
<tr>
<td>[ {ON } ]</td>
</tr>
<tr>
<td>[ {KUN } ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[KOKUGO (field-name-2 [D] [I] ) ]</td>
</tr>
<tr>
<td>[ ]</td>
</tr>
<tr>
<td>[DENWA (field-name-2 [D] [I] [DLM]) ]</td>
</tr>
</tbody>
</table>
[ {BUSHU} ]
[ {{SOKAKU} ...} ]
[ {ON} ]
[ {KUN} ]

Each of these words identify a unique form of Kanji sequencing.

**BUSHU** - ordered by the Radical Stroke count of the Kanji character.

**SOKAKU** - ordered by the Total Stroke count of the Kanji character.

**ON** - ordered by the 'ON' or Chinese pronunciation of the Kanji Character.

**KUN** - ordered by the 'KUN' or Japanese pronunciation of the Kanji Character.

Specify one or more of these keywords to indicate the ordering technique that CA-Easytrieve/Plus is to apply to field-name-1. If you code more than one keyword, they must be coded in major to minor order. CA-Easytrieve/Plus uses each minor ordering technique to order those records that are the same after applying the preceding ordering technique. For example if you coded “USING (FIELD-ONE SOKAKU ON KUN)” FIELD-ONE is ordered, using the Total Stroke Count of the Kanji. Those records having the same count are then ordered by their ON reading. If there are records still the same, they are ordered by their KUN or Japanese reading.

Each of the keywords can only be coded once after field-name-1.

[KOKUGO (field-name-2 [D] [I])]

The KOKUGO sort option invokes the Japanese Dictionary sequencing technique for field-name-1. This technique sequences the records based on the phonetic reading of the WHOLE Kanji field. This technique requires you to specify a phonetic syllabary field - field-name-2. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS format data, it must contain the phonetic reading of field-name-1 represented by JEF DBCS Hiragana or Katakana.

Coding a D causes the field to be sorted in descending order. If you do not code the D, by default the field is sorted in ascending order. Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence.
The DENWA sort option invokes the Japanese Telephone Directory sequencing technique for field-name-1. This technique requires the specification of a phonetic syllabary field - field-name-2. Field-name-2 must be either an alphanumeric (type A) or DBCS (type K) field type. If field-name-2 is an alphanumeric field type, the field must contain the phonetic reading of field-name-1 in Katakana. When the field contains DBCS format data, it must contain the phonetic reading of field-name-1 represented by JEF DBCS Hiragana or Katakana.

Coding a D causes the field to be sorted in descending order. If you do not code the D, by default the field is sorted in ascending order. Coding an I causes the phonetic reading order to be the IROHAON sequence. If you do not code I, by default the phonetic order is the Japanese Dictionary or GOJUON sequence. You can code the DLM subparameter if both field-name-1 and field-name-2 contain space characters that act as delimiters thus permitting the JEF Kanji Sort to associate the phonetic characters in field-name-2 with the Kanji characters in field-name-2. If no such delimiters exist, you should not code DLM. When you do not code the DLM subparameter, CA-Easytrieve/Plus assumes the NODLM sort option.

**CONTROL Statement**

The CONTROL statement identifies control fields used for a control report. You can code DBCS and MIXED fields on this statement in a report regardless of the type of printer that is to receive the output records. A control break occurs whenever the value of any control field changes or end-of-report occurs. The control break at end-of-report is equivalent to the final break. Specify control fields in major to minor order. A break level, discussed later in this chapter, is also assigned to each control field. Comparison of control fields is a logical compare. The syntax of the CONTROL statement is:

**Syntax**

```
[CONTROL [field-name] [NEWPAGE] [NOPRINT] ...
[FINAL] [RENUM] ]
```

**Operation**

You can specify one or more control breaks. If you do not specify any, a final break is implied.
Prior to the first field-name, if any, you can code FINAL to specify options for the control break at end-of-report. Field-name specifies any non-quantitative field located in an active file or in a W-type working storage field.

**Note:** Varying length and quantitative fields cannot be specified on a CONTROL statement.

Three options alter normal processing of a control break:

- **[NEWPAGE]**
  
  NEWPAGE causes a skip to top-of-page after control break processing is complete for the specified field. NEWPAGE specified with FINAL causes a skip to top-of-page before control-break processing occurs at end-of-report.

- **[RENUM]**
  
  RENUM performs the same function as NEWPAGE, and also resets the page number to 1 on the page following the control break.

- **[NOPRINT]**
  
  NOPRINT suppresses printing the summary line group for the specified control break. All other control break processing for the specified control break is performed as usual.

Refer to “Label Reports” and “Control Reports” later in this chapter for detailed examples of the CONTROL statement.

**SUM Statement**

The SUM statement specifies the quantitative fields which are totaled for a control report. Its syntax is:

**Syntax**

```
SUM  field-name  ...
```
Operation

*field-name*

Field-name is any quantitative field contained in an active file or W storage. You can specify multiple fields.

Normally, CA-Easytrieve/Plus automatically totals all quantitative fields specified on LINE statements. The SUM statement overrides this process; only the fields specified on the SUM statement are totaled. The fields specified on a SUM statement do not have to be specified on a LINE statement. The SUM statement is only valid within a Control Report.

TITLE Statement

One or more TITLE statements define the optional report title. The TITLE statement defines the title items and their position on the title line.

Coding DBCS and MIXED fields and literals on these control cards is only supported for reports assigned to an extended reporting printer that supports DBCS data. For a full description of extended reporting printers, please refer to the CA-Easytrieve/Plus Extended Reporting Facilities Guide.

CA-Easytrieve/Plus automatically positions the system date and current page count on title line one. This can be overridden by the options on the REPORT statement (NODATE/NOKDATE/LONGDATE/SHORTDATE and NOPAGE/NOKPAGE).

Those fields and literals coded on a TITLE statement containing DBCS data, convert the DBCS data into the DBCS code supported by the extended reporting printer associated with the report. The data format of the literal remains unchanged. TITLE statement syntax is:

Syntax

```plaintext
TITLE   [literal-1]
{field-name   }
{\'literal-2'  }
{+integer-1   } ...
{-integer-1   }
{COL integer-2}
{#integer-3    }
```
**Report Definition Statements**

**Operation**

[literal-1]

Specify the number of a title with literal-1. The title number must be from 1 to 99 (default is 1). The title number specifies the position of the title line within the title area. You must specify title numbers in ascending order with no duplicates. The title number of the first TITLE statement must be 1 or unspecified.

You must code at least one title item, specified by field-name or literal-2, on each TITLE statement.

{field-name}

The field-name entry specifies a field in any active file, working storage field, or system-defined field.

{literal-2}

Literal-2 specifies a character string for a title item. Specify the literal within quotes.

By default, each title line is formatted as a list of title items that are separated by the number of spaces defined by the SPACE parameter of the REPORT statement. The +, - and COL parameters can modify this positioning.

{          }
{+integer-1}
{-integer-1}

The space adjustment parameters, +integer-1 or -integer-1, modify the normal spacing between title items. Integer-1 is added to or subtracted from SPACE to get the absolute space between title items. The absolute space value can range from zero to any amount which still enables the title line to fit within the current LINESIZE value.

{COL integer-2}

The COL parameter specifies the print column number where the next title item is placed. The value of integer-2 has a valid range of 1 to 'nnn', where 'nnn' cannot force the following title item beyond the end of the title line LINESIZE. COL is permitted only when you specify the NOADJUST parameter of the REPORT statement. If the report is associated with an extended reporting printer which is not a standard Line Printer, an error results if two or more fields and/or literals overlap. (See the *CA-Easytrieve/Plus Extended Reporting Facility Guide* for more information about extended reporting.)
Each title line is centered within the title area of the report unless you specify NOADJUST. TITLE 01 receives extra consideration as follows:

1. If the report is to be printed on an extended reporting printer that supports DBCS data, the system-defined field KANJI-DATE is automatically placed in positions one through eighteen of the title area. If the NOKDATE parameter was specified on the REPORT statement or the report is directed to a printer that does not support DBCS data, the system-defined field, the current date is automatically placed, starting in position one of the title area. If you code the NODATE parameter on the REPORT statement, neither system-defined field is placed on the title.

2. If the report is to be printed on an extended reporting printer that supports DBCS data, the current page number and the Kanji character for page are placed at the right-hand end of the title area. The current page number is automatically edited and converted into the appropriate DBCS values before being combined before the Kanji character for page. If the NOKPAGE parameter was specified on the REPORT statement or the report is directed to a printer that does not support DBCS data, the value in the PAGEWRD option (see Appendix C, “Options Table”) and the current page count are placed at the right-hand end of the title area. If you code the NOPAGE parameter on the REPORT statement, neither format of the page number is placed on the title.

[#integer-3]

The #integer-3 value defines a font index. The value of integer-3 identifies a font whose specifications are to be used for the next display item. You can only specify this option if the report has been associated with an extended reporting printer. Integer-3 identifies the font number of a font defined for the associated extended reporting printer. If you do not code the font number, then the next display item uses the default font for the assigned extended reporting printer.

If you code space adjustment or COL options before the font index option, integer-1 or integer-2 refers to horizontal sizes based on the default width of the assigned extended reporting printer. If you code integer-3 before either of these options, integer-1 or integer-2 refers to horizontal sizes based on the width of the font that integer-3 identified. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)
**HEADING Statement**

The HEADING statement optionally defines an alternate heading for a field. When defining the field, you can specify the default heading. The HEADING statement in a report enables you to override the default for that report. Should there be a DBCS or MIXED literal in any of the lines of the heading that is to be used for a report, then the heading is only valid if the report is directed to an extended reporting printer that supports DBCS data. Otherwise, CA-Easytrieve/Plus generates a syntax error when the field is used on the LINE statement.

If the heading of the field contains DBCS or MIXED data, CA-Easytrieve/Plus converts the literal to the DBCS code system of the extended reporting Printer associated with the report. The data format of the literal remains unchanged. The syntax of the HEADING statement is:

**Syntax**

```
HEADING field-name ([integer-1] 'literal' ...)
```

**Operation**

*field-name*

The field-name specifies a field coded on the LINE 01 statement.

*integer-1*

Integer-1 defines the font number of a font that CA-Easytrieve/Plus uses to format 'literal-1' in the heading area of a report. You can only specify integer-1 if you direct the report to an extended reporting printer. If you direct the report to a normal printer, CA-Easytrieve/Plus outputs a syntax error when you code integer-1. Each 'literal-1' can use a unique font index by coding the # sign and a value for integer-1 before 'literal-1'. Any 'literal-1' that does not have a font index assigned uses the default font for the assigned extended reporting printer. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

*('literal' ...)*

One or more alphanumeric literal values specify the content of the alternate heading. Multiple literals within the parentheses are stacked vertically over the field when printed.

The HEADING statement overrides the field heading defined in the library chapter. The HEADING statement also provides alternate heading capabilities for system-defined fields such as TALLY and LEVEL.
LINE Statement

The LINE statement defines the content of a report line. One or more field values or literals can be contained on a report line; each one is a line item. The support for DBCS and MIXED fields and literals is limited to those reports directed to extended reporting printers supporting DBCS data.

Those fields and literals that contain DBCS data have their DBCS data converted into the DBCS code supported by the extended reporting printer associated with the report. The data format of the field or literal remains unchanged. The syntax of the LINE statement is:

**Syntax**

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

**Operation**

**[literal-1]**

Specify the optional line number with literal-1. The line number specifies the position of the line within the line group. The value must be from 1 to 99; the default is 1. You can omit literal-1 of the first LINE. You must specify the line numbers for multiple LINE statements in ascending order with no duplicates. Specify at least one data item (field-name or 'literal-2') on each LINE statement.

**{field-name}**

Field-name can specify any field contained in an active file or in working storage. If the field is contained in a file or W storage, data is transferred to the print line at the time the PRINT statement is executed. If the field is contained in S storage, data is transferred to the print line at the time the line is printed.

**{'literal-2'}**

'Literal-2' defines a static value for a line item. It must be an alphanumeric literal enclosed within single quotes.
The space adjustment parameters, +integer-1 or -integer-1, modify the spacing between line items. The integer-1 value is added to or subtracted from the SPACE value to give the absolute spacing between line items. The absolute space value can range from zero to any amount which still lets the next line item fit within the line defined by LINESIZE.

COL specifies the column number where the next line item is placed. You must specify the NOADJUST option of the REPORT statement to use the COL option. The value of integer-2 has a valid range of 1 to 'nnn', where 'nnn' cannot be so large that the following line item extends beyond the end of the line defined by LINESIZE.

When the report is associated with an extended reporting printer which is not a standard line printer, an error results if two or more fields and/or literals overlap. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)

The POS parameter enables you to position line items on lines 2 through 99 so that they line up under particular line items on the first line. Integer-3 corresponds to the line item number of LINE 01 under which the line item is placed.

For control reports, any quantitative field on the LINE statement is automatically totaled on each summary line. This feature can be overridden on the SUM statement.

The font index option identifies the font specifications to be used for the next display item. You can only specify this option if the report has been associated with an extended reporting printer. Integer-4 identifies the number of a font defined for the associated extended reporting printer. If you do not code the font, the next display item uses the default font for the assigned extended reporting printer.

If you code space adjustment or COL options before the font index option, integer-1 refers to horizontal sizes based on the default width of the assigned extended reporting printer. If you code the font index before either of these options, then integer-1 refers to horizontal sizes based on the width of the font that integer-4 identifies. (See the CA-Easytrieve/Plus Extended Reporting Facility Guide for more information about extended reporting.)
Standard Reports

The CA-Easytrieve/Plus report facility includes all of the functions necessary to produce most reports very easily. Using CA-Easytrieve/Plus report options, you can produce almost any report format. Most reports, however, are variations of what is termed the standard report. This topic describes the standard report.

Titles

The title is the first item printed on each report page. You can specify the report title with up to 99 TITLE statements. The exhibit below displays the title area of a report.

```
<table>
<thead>
<tr>
<th>date</th>
<th>LINESIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE 01 items</td>
<td>PAGE</td>
</tr>
<tr>
<td>TITLE 02 items</td>
<td>ZZ,ZZ9</td>
</tr>
<tr>
<td>TITLE 04 items</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>TITLE 99 items</td>
<td></td>
</tr>
</tbody>
</table>
```

**Note:** There is a maximum of 32,767 for page count. This field is internally a two byte signed binary field.

The following is a list of particular points to remember about standard report titles:

- TITLE 01 items are printed at top-of-form.
- The current date and page count are placed at either end of the TITLE 01 line.
- Title lines are centered within the space indicated by the LINESIZE parameter of the REPORT statement.
- The title line number controls the vertical spacing of titles relative to the first title.
- The SPACE parameter controls the number of blank characters (spaces) between title items.
The following are title statement examples and their resulting titles:

Statements:

FILE PERSNL FB(150 1800)
%PERSNL
JOB INPUT PERSNL NAME MYPROG
PRINT REPORT1
REPORT REPORT1 LINESIZE 50
TITLE 01 'TEMPORARY EMPLOYEES'
TITLE 03 'IN DEPARTMENT' DEPT
LINE 01 ' '

Produce:

2/09/87 TEMPORARY EMPLOYEES PAGE 1
IN DEPARTMENT 903

You can override the automatic (default) functions associated with title contents and spacing to produce any desired report title. This can be necessary to produce reports which use pre-printed forms as the output medium. You can use the following parameters to produce non-standard title content and spacing:

- NOADJUST causes each title line to be left-justified on the page. Use of the COL positioning parameter requires NOADJUST. NOADJUST can cause line items to overlay the tags printed for SUMCTL TAG. COL positioning is necessary for tag to appear.
- NODATE and NOPAGE inhibit current date and page count information from being placed on the first title line.

The current date overlays the left-most positions of TITLE 1 when NOADJUST is specified. Either use NODATE or reserve an area by specifying COL 10 or COL 12.
The following examples of title statements use title content and space adjustment parameters. The report title that results from the statements is also illustrated.

Statements:

```
FILE PERSNL FB(150 1800)
&PERSNL
JOB INPUT PERSNL NAME MYPROG
PRINT REPORT1
REPORT REPORT1 NOADJUST NODATE NOPAGE
TITLE 01 COL 20 SSN
TITLE 02 SYSDATE COL 20 NAME
TITLE 03 COL 20 ADDR-STREET
TITLE 04 COL 20 ADDR-CITY -3 ',,' +
       -2 ADDR-STATE +5 ADDR-ZIP
LINE  01 ' '
```

Produce:

```
column              column
 0                  2
 1                  0

11/19/86            025-30-5228
WIMN                GLORIA
430 M ST SW 107     BOSTON    , MA     02005
```

**Headings**

A report heading is normally printed for line items specified on LINE 01. Each heading is centered over its associated line item. The following rules control the heading; the order in which they are listed indicates the hierarchy of override:

1. The NOHEADING parameter of REPORT inhibits the printing of any headings.
2. The HEADING statement sets the item heading content.
3. The HEADING parameter of DEFINE sets the item heading content.
4. Field-name-1 of DEFINE sets the item heading content.
5. Line items which are literals do not have headings.
6. Only LINE 01 items have headings.
The exhibit below illustrates the positioning of headings in a typical report. Line items do not always have the same number of heading entries. In this case, the corresponding heading line area is blank for those items with missing headings.

<table>
<thead>
<tr>
<th>T I T L E A R E A</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLESKIP space</td>
</tr>
<tr>
<td>Heading area</td>
</tr>
<tr>
<td>report body</td>
</tr>
</tbody>
</table>

The following exhibit illustrates various heading options:

Statements:

FILE PERSNL FB(150 1800)
  SSN 4 5 P MASK '999-99-9999' +
  NAME 17 20 A
  NAME-LAST NAME 8 A
  NAME-FIRST NAME 8 12 A
  PAY-NET 90 4 P 2
  JOB INPUT PERSNL NAME MYPROG
  PRINT REPORT1
*  REPORT REPORT1 LINESIZE 65
  HEADING PAY-NET ('NET', 'PAY')
  LINE NAME SSN '* NO OVERTIME *' PAY-NET

Produce:

<table>
<thead>
<tr>
<th>SOCIAL SECURITY</th>
<th>NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>PAY</td>
</tr>
<tr>
<td>WIMN GLORIA</td>
<td>025-30-5228 * NO OVERTIME * 251.65</td>
</tr>
<tr>
<td>BERG NANCY</td>
<td>121-16-6413 * NO OVERTIME * 547.88</td>
</tr>
</tbody>
</table>

**Line Group**

Lines compose the body of a report. The lines of a report are output in groups for each PRINT statement issued. All of the LINE statements of the report make up a line group, which is also called a logical report line.

LINE ... } line or logical report
LINE 02 ...} group line
LINE 03 ...} group line

...
Line Item Positioning

Line item positioning follows three rules:

- LINE 01 items and their associated headings are centered in an area whose length is controlled by the longer of the following:
  a. The line item
  b. The longest heading entry

  The resulting value is called the item length.

- The first line item other than on LINE 01 (that is, LINE 02 through LINE 99) is positioned under the first item of LINE 01. The data is left-justified under the LINE01 data regardless of the heading size.

- Blank characters (spaces) separate all line items according to the value of the SPACE parameter of the REPORT statement.

When CA-Easytrieve/Plus analyzes a LINE statement according to the above rules, the total number of characters on that line must not exceed LINESIZE.

The following exhibit illustrates line item positioning:

FILE PERSNL FB(150 1800)
SSN            4 5 P       MASK '999-99-9999' +
HEADING('SOCIAL' 'SECURITY' 'NUMBER')
NAME           17 20 A     HEADING 'EMPLOYEE NAME'
NAME-LAST NAME  8 A        HEADING 'LAST NAME'
NAME-FIRST NAME +8 12 A   HEADING 'FIRST NAME'
ADDRESS        37 39 A    HEADING 'STREET'
ADDR-STREET    37 20 A    HEADING 'STREET'
ADDR-CITY      57 12 A    HEADING 'CITY'
ADDR-STATE     69 2 A     HEADING 'STATE'
ADDR-ZIP       71 5 N     HEADING 'ZIP CODE'
SEX            127 1 N    HEADING 'SEX CODE'
JOB INPUT PERSNL NAME MYPROG
PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65
LINE NAME SSN SEX
LINE 02 ADDR-STREET POS 2 ADDR-CITY

<table>
<thead>
<tr>
<th>line area</th>
<th>item area</th>
<th>item area</th>
<th>item area</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 10 20</td>
<td>1 5 10 14</td>
<td>...........</td>
<td>...........</td>
</tr>
</tbody>
</table>

SOCIAL SECURITY SEX heading

EMPLOYEE NAME

WIMN  GLORIA 025-30-5228 1 line group
**Special Positioning**

Sometimes the standard positioning of line items on a report is unsuitable for producing the desired result, as in the case of aligning numeric fields on LINE 02 with the decimal point of corresponding fields on LINE 01. The 'POS nnn' line item adjusting parameter left-justifies the corresponding fields, but when the LINE 02 field is not as long as the LINE 01 field, the two fields are unaligned. If that happens, use the '+nnn' or '-nnn' line item adjustment parameter after the 'POS nnn' parameter to adjust the data's position. The following exhibit illustrates poor and good decimal point alignment.

**Statements:**

```
DEFINE FLD1  W 4 P 2 VALUE 123.45
DEFINE FLD2  W 3 P 2 VALUE 678.90
JOB INPUT NULL NAME MYPROG
   PRINT REPORT1
   STOP
*
REPORT REPORT1 LINESIZE 40
   LINE 01 FLD1          FLD1
   LINE 02 FLD2 POS 2 +3 FLD2
```

**Produce:**

<table>
<thead>
<tr>
<th>poor column</th>
<th>good column</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 5 10 15</td>
<td>1 5 10 15</td>
</tr>
<tr>
<td>............</td>
<td>............</td>
</tr>
<tr>
<td>FLD1</td>
<td>FLD1</td>
</tr>
<tr>
<td>123.45</td>
<td>123.45 line 01</td>
</tr>
<tr>
<td>678.90</td>
<td>678.90 line 02</td>
</tr>
</tbody>
</table>
Pre-printed Form Production

Pre-printed form production is another instance when standard line item positioning must be overridden. A very simple example of this override is W-2 form printing in a payroll application. The exhibit below depicts the report declarative statements necessary to print a hypothetical W-2 form:

```
REPORT PAGESIZE 20 NOADJUST NOHEADING SPACE 1
LINE COL 7 'YOUR COMPANY NAME' COL 33 '903' +
         COL 39 '12-3456789'
LINE 82 COL 7 'YOUR COMPANY STREET'
LINE 83 COL 7 'YOUR COMPANY CITY STATE ZIP'
LINE 10 COL 7 SSN COL 23 YTD-FEDTAX +
         COL 39 YTD-WAGES +
         COL 54 YTD-FICA
LINE 12 COL 7 EMP-NAME COL 39 YTD-WAGES
LINE 14 COL 7 EMP-STREET
LINE 15 COL 7 EMP-CITY EMP-STATE EMP-ZIP
```

![W-2 Form Example](image-url)
**SPREAD Option**

The SPREAD option of the REPORT statement offers a unique opportunity for line item spacing. When you use reports as work sheets, it is often desirable to space line items as far apart as possible. SPREAD overrides the SPACE parameter of the REPORT statement and creates report lines with the maximum number of spaces between each item, as the exhibit below illustrates:

Statements:

```
DEFINE FLD1  W 4 P 2 VALUE 123.45
DEFINE FLD2  W 3 P 2 VALUE 678.90
DEFINE FLD3  W 4 P 2 VALUE 1129.59
JOB INPUT NULL NAME MYPROG
  PRINT REPORT1
STOP
* REPORT REPORT1 SPREAD LINESIZE 65
  TITLE 'S P R E A D   EXAMPLE'
  LINE FLD1 FLD2 FLD3
```

Produce:

```
11/19/86             S P R E A D   EXAMPLE         PAGE      1
FLD1               FLD2               FLD3
123.45            678.90            1,129.59
```
Label Reports

You can use the CA-Easytrieve/Plus label report capability to print mailing labels and other applications that require inserting variable data in a repetitious format.

A label report is different from a standard report in the following ways:

- Label reports do not have titles and headings.
- Multiple labels can be printed side-by-side.
- Controlled label reports permit control breaks, but do not automatically permit total quantitative fields. Totals, however, can be specified on a SUM statement and processed in BEFORE-BREAK and AFTER-BREAK procedures.

You can use the label report function whenever a complete logical print page is produced by each PRINT statement. Consider the W-2 form printing example; print time can be reduced substantially by having 2-up forms. You can then modify report declaration statements as follows:

```
REPORT LBLS LABELS (ACROSS 2 DOWN 15 SIZE 65 NEWPAGE) SPACE 1
LINE 01 COL 7 'YOUR COMPANY NAME' COL 33 '903' +
   COL 39 '12-3456789'
LINE 02 COL 7 'YOUR COMPANY STREET'
LINE 03 COL 7 'YOUR COMPANY CITY STATE ZIP'
LINE 10 COL 7 SSN COL 23 YTD-FEDTAX +
   COL 39 YTD-WAGES +
   COL 54 YTD-FICA
LINE 12 COL 7 EMP-NAME COL 39 YTD-WAGES
LINE 14 COL 7 EMP-STREET
LINE 15 COL 7 EMP-CITY EMP-STATE EMP-ZIP
```
**CONTROL Statement**

You can use the CONTROL statement with label reports to truncate a group of labels. Truncating makes it easy to separate labels after they are printed. The following exhibit demonstrates how a new label page starts when the control field changes.

Statements:

```plaintext
FILE PERSNL FB(150 1800)
%PERSNL
FILE SORTWRK FB(150 1800) VIRTUAL
COPY PERSNL
SORT PERSNL TO SORTWRK +
USING (ADDR-STATE, ADDR-ZIP) NAME MYSORT
JOB INPUT SORTWRK NAME MYPROG
PRINT REPORT1
*
REPORT REPORT1 LABELS
CONTROL ADDR-STATE NEWPAGE
LINE 01 NAME
LINE 02 ADDR-STREET
LINE 03 ADDR-CITY, ADDR-STATE, ADDR-ZIP
```

Produce:

```
  XXX  XXX  XXX  XXX
  XXX  XXX  XXX  XXX
  XXX  XXX  XXX  XXX
  XXX  XXX  XXX  XXX
  XXX  XXX  XXX  XXX
  XXX  XXX  XXX  XXX
```

*(goes to new page when ADDR-STATE changes)*

```
  VVV  VVV
  ...  VVV
```

**Note:** Any labels remaining on a line are left unused. The optional NEWPAGE parameter causes a top-of-page for the next print line.

---

**Sequenced Reports**

**SEQUENCE Statement**

Report sequence is controlled either by the order in which PRINT statements are issued or by the SEQUENCE statement. You can print both standard and label reports in any sequence.
Report definitions which contain SEQUENCE statements cause the report data to be spooled to a temporary work file. Work file usage is transparent.

The SEQUENCE function is performed by invoking your installation's sort program. The temporary work file is input to the sort program (through its E15 exit). When the sort is complete, the work file data is retrieved (through its E35 exit) and the report is produced.

Only those data elements used in the report are sorted. The sorted output is directly printed from the E35 exit. These attributes combine to make the SEQUENCE facility of CA-Easytrieve/Plus extremely efficient.

CONTROL Reports

The CONTROL statement specifies that a report automatically accumulates and prints totals of quantitative report information. The report accumulates information according to the hierarchy indicated on the CONTROL statement.

Terminology

The following terms are used throughout the discussion on control reports:

- A control field is any field named on a CONTROL statement to establish the hierarchy of a control report.
- A control break occurs whenever any field of the control hierarchy changes value.
A total line is a line group in a report body which contains control totals. Total lines are normally annotated on the left with the value of control fields according to the SUMCTL parameter of REPORT. This is done by listing the control fields first on the LINE statement.

A detail line is the same line data as in a standard report body line (not a total line). Detail lines are normally annotated on the left, with the value of control fields specified according to the DTLCTL parameter of REPORT. The SUMMARY parameter of REPORT inhibits the printing of detail lines on a control report.

Accumulators are system-created fields which contain totals. Accumulators are created for:
- All fields designated on the SUM statement
- All active file or W storage quantitative (signed) fields designated on the line group (LINE nn) statements, if a SUM statement is not specified. (Quantitative fields are numeric fields with a decimal point designation of 0 through 18.)

SUMFILE data are the contents of control fields and accumulators at each minor control break.

**Data Reference**

In general, report statements and procedures can reference any field of an active file or working storage. (Some report procedures have minor restrictions which are described with the associated procedure.)

Statements and procedures can directly reference data for detail lines in non-control reports. Data reference for total lines actually accesses SUMFILE data. This data includes all control fields and ten-byte (18 digit) packed fields for all accumulators. (See Summary File, later in this chapter.)

**TALLY**

TALLY is a system-defined field for control reports. TALLY contains the number of detail records that comprise a control break. You can use TALLY on a LINE statement or you can use it in calculations within report procedures. TALLY is commonly used to determine averages for a control level.

TALLY is a ten-byte packed decimal field with zero decimal places. This definition is used for calculations contained within report procedures. REPORT TALLYSIZE defines the number of digits which are printed for TALLY. A TALLY accumulator is created for each control break level.
LEVEL

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.

--- SUMFILE data ---

--- control fields --- accumulators --- LEVEL ---

<table>
<thead>
<tr>
<th>control</th>
<th>control</th>
<th>control</th>
<th>SUM</th>
<th>SUM</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>field-n</td>
<td>..</td>
<td>field-2</td>
<td>TALLY</td>
<td>field-1</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>field-1</td>
<td>TALLY</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TALLY</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUM</td>
<td>SUM</td>
<td>n + 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TALLY</td>
<td>field-1</td>
<td>...</td>
</tr>
</tbody>
</table>

See “BREAK-LEVEL” on the following page for an example of testing LEVEL and BREAK-LEVEL.

BREAK-LEVEL

BREAK-LEVEL is a system-defined field whose value indicates the highest control break level. The following exhibit illustrates using BREAK-LEVEL to display an appropriate message in a BEFORE-BREAK procedure:

REPORT RPT
  SEQUENCE REGION BRANCH
  CONTROL REGION BRANCH
  LINE REGION BRANCH NAME PAY-GROSS
  BEFORE-BREAK, PROC
  IF LEVEL = 1 . * processing lowest break
  IF BREAK-LEVEL = 1 . * only branch is breaking
    DISPLAY ‘*** BRANCH TOTALS’
  ELSE-IF BREAK-LEVEL = 2 . * region is breaking too
    DISPLAY ‘*** BRANCH AND REGION TOTALS’
  ELSE-IF BREAK-LEVEL = 3 . * final report totals
    DISPLAY ‘*** BRANCH, REGION, AND FINAL TOTALS’
  END-IF
  END-IF
END-PROC
CA-Easytrieve/Plus invokes the BEFORE-BREAK procedure before printing summary lines. See “Report Procedures” later in this chapter for more information.

In the above example, LEVEL and BREAK-LEVEL fields are used to determine the appropriate message displayed before the summary lines are printed. Testing for LEVEL 1 tells us that CA-Easytrieve/Plus is going to print the first summary line next (BRANCH totals). When BREAK-LEVEL is 1, only the BRANCH field is breaking. Therefore, we want to display a message stating this. When BREAK-LEVEL is 2, the REGION field is breaking. This causes both BRANCH and REGION summary lines to print. When BREAK-LEVEL is 3, CA-Easytrieve prints BRANCH, REGION, and final summary lines.

**IF BREAK/IF HIGHEST BREAK Processing**

An alternative to testing LEVEL and BREAK-LEVEL is to use IF field-name BREAK and IF field-name HIGHEST-BREAK processing. Using the example in the exhibit below, you can code the following for the same result:

```
REPORT RPT
  SEQUENCE REGION BRANCH
  CONTROL REGION BRANCH
  LINE REGION BRANCH NAME PAY-GROSS
  BEFORE-BREAK. PROC
    IF BRANCH BREAK                . * processing lowest break
      IF BRANCH HIGHEST-BREAK      . * only branch is breaking
        DISPLAY '*** BRANCH TOTALS ***'
      ELSE-IF REGION HIGHEST-BREAK . * region is breaking also
        DISPLAY '*** BRANCH AND REGION TOTALS ***'
      ELSE-IF FINAL  HIGHEST-BREAK . * final report totals
        DISPLAY '*** BRANCH, REGION AND FINAL TOTALS ***'
    END-IF
  END-IF
END-PROC
```

Coding IF field-name BREAK is equivalent to coding IF LEVEL = x, where x is the break level assigned to field-name. IF HIGHEST-BREAK performs the same function against the BREAK-LEVEL field. IF BREAK and IF HIGHEST-BREAK have the advantage of dynamically changing the LEVEL value if fields are added to or removed from the CONTROL statement.

**Control Report Contents**

The report body contains the only difference between standard and control report contents. Control reports print total lines in addition to detail lines (optional). The following examples use two control fields (STATE and ZIP) which contain data that is two and five bytes long, respectively, and one quantitative field (PAY-NET) which contains numeric data.
The standard control report contains standard report data plus total data. The following exhibit illustrates the report body of such a report. Detail and total lines are shown, with the totals illustrating the hierarchy of the report data.

Statements:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65
SEQUENCE STATE ZIP LAST-NAME
CONTROL STATE ZIP
LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWN IL 60076 123.45
BROWN IL 60076 678.90
BROWN IL 60076 802.35
JONES IL 60077 543.21
JONES IL 60077 98.76
JONES IL 60077 641.97
SMITH TX 75218 666.66
SMITH TX 75218 111.11
SMITH TX 75218 777.77
STATE total TX 777.77
FINAL total 2222.09

Produce:

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>ZIP total</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>ZIP total</td>
<td>JONES</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>ZIP total</td>
<td>SMITH</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>
The same report without the detail lines is a SUMMARY report. For example:

Statements:

```
FILE FILE1 CARD
LAST-NAME  1 20 A
STATE     21 2 A
ZIP       23 5 N
PAY-NET   28 5 N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 SUMMARY +
  DTLCTL  NONE
  SEQUENCE STATE ZIP
  CONTROL STATE ZIP
  LINE 01 STATE ZIP PAY-NET
END
BROWN     IL6007612345
BROWN     IL6007667890
JONES     IL6007709876
JONES     IL6007754321
SMITH     TX7521811111
SMITH     TX7521866666
```

Produce:

```
<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATE ZIP PAY-NET</td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL 60076</td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL 60077</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>ZIP total</td>
<td>TX 75218</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>
```
DTLCTL

The DTLCTL parameter of REPORT establishes the method for printing control field values on detail lines of a control report by using the subparameters EVERY, FIRST and NONE. The following exhibit illustrates an example program using DTLCTL options:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE     6  2 A
ZIP       8  5 N
PAY-NET   13 5 N 2
JOB INPUT FILE1 NAME MYPROG
    PRINT REPORT1
    *
REPORT REPORT1 LINESIZE 65 +
    DTLCTL option (* with option being EVERY, FIRST, and NONE *)
    SEQUENCE STATE ZIP LAST-NAME
    CONTROL STATE ZIP
    LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWN IL 60076 123.45
BROWN IL 60076  678.90
JONES IL 60077  98.76
JONES IL 60077  543.21
SMITH TX 75218 111.11
SMITH TX 75218  666.66

The following exhibit shows the results of using all three DTLCTL options:

EVERY -- prints all control fields on every detail line.

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>ZIP total</td>
<td>TX</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>
FIRST -- prints all control fields on the first detail line at top-of-page and after each break.

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>ZIP total</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>ZIP total</td>
<td>JONES</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>ZIP total</td>
<td>SMITH</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>

NONE -- prints no control fields on detail lines.

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>BROWN</td>
</tr>
<tr>
<td>ZIP total</td>
<td>BROWN</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>detail</td>
<td>JONES</td>
</tr>
<tr>
<td>ZIP total</td>
<td>JONES</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>detail</td>
<td>SMITH</td>
</tr>
<tr>
<td>ZIP total</td>
<td>SMITH</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>
SUMCTL

The SUMCTL parameter of REPORT establishes the method for printing control field values on total lines of a control report by using the subparameters ALL, HIAR, NONE and TAG. (The DTLCOPY subparameter controls all non-control non-total values on total lines.) The following exhibit shows an example program using these options:

FILE FILE1 CARD
LAST-NAME  1  5 A
STATE      6  2 A
ZIP        8  5 N
PAY-NET    13 5 N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 +
SUMCTL option
  (* with option being ALL, HIAR, NONE, and TAG *)
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666

The following exhibit illustrates the results of using three of the SUMCTL options:

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>TX</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td>TX</td>
</tr>
</tbody>
</table>
HIAR -- prints control fields in hierarchical order on total lines.

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>TX</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>

NONE -- prints no control fields on total lines.

<table>
<thead>
<tr>
<th>Control Fields</th>
<th>Accumulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAST-NAME</td>
<td>STATE</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>IL</td>
</tr>
<tr>
<td>STATE total</td>
<td>IL</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP total</td>
<td>TX</td>
</tr>
<tr>
<td>STATE total</td>
<td>TX</td>
</tr>
<tr>
<td>FINAL total</td>
<td></td>
</tr>
</tbody>
</table>

TAG

The TAG subparameter of SUMCTL creates a line area on the left side of the total line. This LINE 01 item is governed by the following rules:

- The length of the area is the length of the longest control-field-name plus seven.
- FINAL TOTAL is the annotation for the final totals line.
- The line item area is positioned at the left margin of the report.
The following exhibit illustrates how tags appear on a report.

Statements:

```
FILE FILE1 CARD
LAST-NAME 1 5 A
STATE   6 2 A
ZIP     8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 +
  SUMCTL TAG
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666
```

Produce:

```
LAST-NAME   STATE    ZIP     PAY-NET
BROWN      IL     60076      678.90
BROWN                        123.45
ZIP TOTAL                      802.35
JONES      IL     60077      543.21
JONES                         98.76
ZIP TOTAL                      641.97
STATE TOTAL                   1444.32
SMITH      TX     75218      666.66
SMITH                        111.11
ZIP TOTAL                      777.77
STATE TOTAL                    777.77
FINAL TOTAL                   2222.09
```

**DTLCOPY**

When printing control reports (particularly a summary report) you can include detail information in total lines. Normally, CA-Easytrieve/Plus prints only control field values and associated totals on total lines. The DTLCOPY subparameter of SUMCTL causes detail field contents (values just prior to the break) to be printed on total lines.
The following exhibit illustrates the use of DTLCOPY.

Statements:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 +
  SUMMARY SUMCTL DTLCOPY
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWN IL6007612345
BROWN IL6007667890
JONES IL6007709876
JONES IL6007754321
SMITH TX7521811111
SMITH TX7521866666

Produce:

LAST-NAME STATE ZIP PAY-NET
BROWN IL60076 882.35
JONES IL60077 641.97
   IL 1444.32
SMITH TX75218 777.77
   TX 777.77

   2222.09
DTLCOPYALL

DTLCOPYALL has the same effect as DTLCOPY except that the detail fields are printed for all control break totals. The following exhibit illustrates the use of DTLCOPYALL.

Statements:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
   PRINT REPORT1
   *
REPORT REPORT1 LINESIZE 65 +
   SUMMARY SUMCTL DTLCOPYALL
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE ZIP
   LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWN IL6007612345
BROWN IL6007667890
JONES IL6007709876
JONES IL6007754321
SMITH TX7521811111
SMITH TX7521866666

Produce:

LAST-NAME  STATE  ZIP  PAY-NET
BROWN    IL60076  802.35
JONES    IL60077  641.35
JONES    IL60077  1444.32
SMITH    TX75218  777.77
SMITH    TX75218  777.77
SMITH    TX75218  2222.09

Control Field Values in Titles

Occasionally, you may want to print control field values in report titles. For example, you can use control field annotation within the title of a report to emphasize the structure of an organization, particularly at its higher levels. This technique uses only basic report facilities, and does not require special parameters.
Consider the following:

Statements:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
PRINT REPORT1

REPORT REPORT1 LINESIZE 65 +
SUMMARY SUMCTL DLTCOPY
SEQUENCE STATE ZIP LAST-NAME
CONTROL STATE NEWPAGE ZIP
TITLE 'REPORT FOR THE STATE OF' STATE
LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWN IL 60076 12345
BROWN IL 60076 7890
JONES IL 60077 9876
JONES IL 60077 4321
SMITH TX 75218 1111
SMITH TX 75218 6666

Produce:

11/23/86 REPORT FOR THE STATE OF IL PAGE 1
LAST-NAME STATE ZIP PAY-NET
BROWN IL 60076 802.35
JONES IL 60077 641.97
IL 1444.32

11/23/86 REPORT FOR THE STATE OF TX PAGE 2
LAST-NAME STATE ZIP PAY-NET
SMITH TX 75218 777.77
TX 777.77
2222.09
Overflow of Total Values

In control reports, line items for totaled fields define an area not only for detail lines, but also for corresponding total lines. Since totals are normally larger than the detail, you need a means of expanding the item area. Without this expansion, the item area might be too small to contain the totals. If your report contains this overflow condition, CA-Easytrieve/Plus automatically depicts it by setting the right-most character of the item area byte to an * (asterisk character), as the exhibit below illustrates.

Statements:

FILE FILE1 CARD
LAST-NAME  1  5 A
STATE      6  2 A
ZIP        8  5 N
PAY-NET    13 5 N 2
*
JOB INPUT FILE1 NAME MYPROG
*
PRINT REPORT1
*
REPORT REPORT1 SUMSPACE 0 +
SUMCTL HIAR LINESIZE 65
SEQUENCE STATE ZIP LAST-NAME
CONTROL STATE NEWPAGE ZIP
TITLE 'REPORT FOR THE STATE OF' STATE
LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007789076
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666

Produce:

2/11/87 REPORT FOR THE STATE OF IL PAGE 1
LAST-NAME STATE ZIP PAY-NET
BROWN IL 60076 678.90
BROWN IL 60076 123.45
IL 60076 802.35
JONES IL 60077 543.21
JONES IL 60077 98.76
IL 60077 641.97
IL 444.32* <-overflow

2/11/87 REPORT FOR THE STATE OF TX PAGE 2
LAST-NAME STATE ZIP PAY-NET
SMITH TX 75218 666.66
SMITH TX 75218 111.11
TX 75218 777.77
TX 777.77
222.09* <-overflow
### Controlling Overflow

You can control this overflow through two methods:

1. Ensure that the detail field being totaled is large enough to absorb the totals. The exhibit below illustrates how overflow can be prevented by effectively expanding the line item to six digit positions.

   Statements:

   ```
   FILE FILE1 CARD
   LAST-NAME  1  5 A
   STATE   6  2 A
   ZIP    8  5 N
   PAY-NET 13  5 N 2
   T-PAY-NET W  6 N 2 HEADING ('PAY-NET')*
   JOB INPUT FILE1 NAME MYPROG
   T-PAY-NET = PAY-NET
   PRINT REPORT1
   *
   REPORT REPORT1 SUMSPACE 0 +
   SUMCTL HIAR LINESIZE 65
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE NEWPAGE ZIP
   "REPORT FOR THE STATE OF" STATE
   LINE 01 LAST-NAME STATE ZIP T-PAY-NET END
   BROWNIL6007612345
   BROWNIL6007667890
   JONESIL6007709876
   JONESIL6007754321
   SMITHTX7521811111
   SMITHTX7521866666
   ```

   Produce:

   **2/11/87 REPORT FOR THE STATE OF IL PAGE 1**

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>678.90</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>123.45</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td>60076</td>
<td>802.35</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>543.21</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>98.76</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td>60077</td>
<td>641.97</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td></td>
<td>1,444.32</td>
</tr>
</tbody>
</table>

   **2/11/87 REPORT FOR THE STATE OF TX PAGE 2**

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>666.66</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>111.11</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td></td>
<td>777.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2,222.09</td>
</tr>
</tbody>
</table>

---
2. Expand the item area by using the SUMSPACE parameter of the REPORT statement. The value of SUMSPACE is added to the basic length of total fields to determine an adjusted line item length. The resulting line item expansion is illustrated in the exhibit below as a print edit mask.

Statements:

```plaintext
FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2.* (999.99- mask without SUMSPACE specified)
*
JOB INPUT FILE1 NAME MYPROG

PRINT REPORT1
*
REPORT REPORT1 SUMSPACE 1 +
  SUMCTL HIAR LINESIZE 65
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE NEWPAGE ZIP
  TITLE 'REPORT FOR THE STATE OF' STATE
  LINE 01 LAST-NAME STATE ZIP PAY-NET
END

BROWN IL 60076 123.45
BROWN IL 60076 802.35
JONES IL 60077 543.21
JONES IL 60077 98.76
SMITH TX 75218 111.11
SMITH TX 75218 777.77

Produce:

2/11/87 REPORT FOR THE STATE OF IL PAGE 1

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>678.90</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>123.45</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>543.21</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>98.76</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>666.66</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>111.11</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>75218</td>
<td>777.77</td>
</tr>
</tbody>
</table>

2/11/87 REPORT FOR THE STATE OF TX PAGE 2

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>666.66</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>111.11</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>75218</td>
<td>777.77</td>
</tr>
</tbody>
</table>

2,222.09 (9999.99- mask with SUMSPACE 1)
Summary File

A summary file, which contains all the control and summed field values at each minor break, can be optionally generated during processing of a control report. JOB activities in your program can subsequently process the summary file to provide reports not otherwise available using the standard report facilities of CA-Easytrieve/Plus.

You can request the summary file by defining the file in the library and then referencing it through the REPORT SUMFILE parameter.

The FILE statement must contain the file-name, record format, logical record length, and blocksize. For most purposes, you should specify the file as an unblocked VIRTUAL file. The record format can be any standard format. The record length must be large enough to contain the data which is output. Blocksize should be appropriate for the specified format and record length.

The summary file record contains three parts (see the exhibit under the topic Control Reports, earlier in this chapter):

1. Control field area
2. TALLY
3. Summed field area

The control field area is a concatenation of the control fields specified on the CONTROL statement. The sum of the lengths of the control fields defines the length of the control field area.

TALLY is a 10-byte field.

The summed fields are concatenated to form the remaining segment of the summary file record. Each summed field is a 10-byte packed field with the same decimal specification as the source field.

Therefore, the summary file record length is the sum of the control field area length, plus 10 bytes for TALLY, plus 10 times the number of summed fields.
The exhibit below illustrates the use of SUMFILE data. The values of SFILE are listed in order of ascending magnitude within SFILE-STATE, without reprocessing the original data:

Statements:

FILE FILE1 CARD  
LAST-NAME  1 5 A  
STATE      6 2 A  
ZIP        8 5 N  
PAY-NET    13 5 N 2  
FILE SFILE F(30)  
SFILE-STATE 1 2 A  
SFILE-ZIP   3 5 N  
SFILE-TALLY 8 10 P 0  
SFILE-PAY-NET 18 10 P 2  
JOB INPUT FILE1 NAME MYPROG  
PRINT REPORT1  
*  
REPORT REPORT1 LINESIZE 65 +  SUMMARY SUMFILE SFILE SUMCTL DTLCOPY  
SEQUENCE STATE ZIP LAST-NAME  
CONTROL STATE NEWPAGE ZIP  
TITLE REPORT FOR THE STATE OF STATE  
LINE 01 LAST-NAME STATE ZIP PAY-NET  
*  
JOB INPUT SFFILE NAME MYPROG2  
PRINT REPORT2  
*  
REPORT REPORT2 NOADJUST  
SEQUENCE SFFILE-STATE SFFILE-PAY-NET  
LINE 01 SFFILE-STATE SFFILE-ZIP +  
SFFILE-TALLY SFFILE-PAY-NET  
END  
BROWNIL6007612345  
BROWNIL6007667890  
JONESIL6007709876  
JONESIL6007754321  
SMITHTX7521811111  
SMITHTX7521866666  
Produce:  

<table>
<thead>
<tr>
<th>SFFILE-STATE</th>
<th>SFFILE-ZIP</th>
<th>SFFILE-TALLY</th>
<th>SFFILE-PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL</td>
<td>60077</td>
<td>2</td>
<td>641.97</td>
</tr>
<tr>
<td>IL</td>
<td>60076</td>
<td>2</td>
<td>892.35</td>
</tr>
<tr>
<td>TX</td>
<td>75218</td>
<td>2</td>
<td>777.77</td>
</tr>
</tbody>
</table>

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 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:

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

- You cannot use the STOP statement.

- 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.

Field Reference

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

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.
Static Working Storage

Fields contained in S storage exhibit unique properties during report processing. S fields are stored in a static working storage area and 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; either immediately upon execution of the PRINT statement or after the processing of work files. 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.

The following exhibit illustrates the use of S fields versus W fields:

```
FILE FILEA CARD
KEY 1 3 A
SFLD S 1 A
WFLD W 1 A
JOB INPUT FILEA NAME MYPROG
IF RECORD-COUNT = 1
  SFLD = 'A'
  WFLD = 'A'
END-IF
IF RECORD-COUNT = 2
  SFLD = 'B'
  WFLD = 'B'
END-IF
PRINT RPT
* REPORT RPT LINESIZE 65
SEQUENCE KEY
LINE KEY SFLD WFLD
END
ZZZ
AAA

Produce:

<table>
<thead>
<tr>
<th>KEY</th>
<th>SFLD</th>
<th>WFLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>ZZZ</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>
```
In the preceding exhibit, note that the value assigned to SFLD for record number one (KEY = ZZZ) was 'A' but the report shows its value as 'B'. This occurs because S fields, as previously mentioned, are static working storage fields which are not spooled out to work files. The value of an S field when printed, is the last value assigned (calculated) to it. The following exhibit helps to illustrate where SFLD and WFLD got their values in the previous exhibit:

<table>
<thead>
<tr>
<th>Working Storage</th>
<th>Spool File</th>
</tr>
</thead>
<tbody>
<tr>
<td>SFLD</td>
<td>WFLD</td>
</tr>
<tr>
<td>KEY</td>
<td>WFLD</td>
</tr>
<tr>
<td>Record 1</td>
<td></td>
</tr>
<tr>
<td>Key = ZZZ</td>
<td>'A'</td>
</tr>
</tbody>
</table>

In the previous exhibit, two records were read and printed in a sequenced report which resulted in the data being written to an intermediary spool file prior to sequencing. As shown in the above diagram, the value for SFLD is not written to the spool file (unlike the value for WFLD). When the report is printed, the LINE statement retrieves its value for SFLD from working storage, but receives its value for WFLD from the spool file.

**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 below 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 when display-page-size is zero. To modify detail line data, change display-page-size to a value larger than zero.

**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 body. 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.

(REPORT-INPUT)<---(caused by the first PRINT statement)

5/18/84        PROCEDURE  USAGE         PAGE 1
STATE   ZIP   PAY-NET

  detail  (BEFORE-LINE)  IL  60076    678.90
         (AFTER-LINE)
  detail  (BEFORE-LINE)  IL  60076    123.45
         (AFTER-LINE)
  total   (BEFORE-BREAK)  IL  60076    802.35
         (AFTER-BREAK)
  detail  (BEFORE-LINE)  IL  60077    543.21
         (AFTER-LINE)
  detail  (BEFORE-LINE)  IL  60077     98.76
         (AFTER-LINE)
  total   (BEFORE-BREAK)  IL  60077    641.97
         (AFTER-BREAK)
  total   (BEFORE-BREAK)  IL  1444.32
         (AFTER-BREAK)
...            (ENDPAGE)

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 exhibit below illustrates use of the REPORT-INPUT procedure in final report input selection. Only the first record within each ZIP code is selected.

Statements:

```
FILE FILE1 CARD FB(80 8000)
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
HOLD-ZIP 5 5 N VALUE 00000
JOB INPUT FILE1 NAME MYPROG
    PRINT REPORT1 *
REPORT REPORT1 LINESIZE 65 +
    SUMMARY SUMCTL DTLCOPY
    SEQUENCE STATE ZIP LAST-NAME
    CONTROL STATE NEWPAGE ZIP
    TITLE ‘REPORT FOR THE STATE OF’ STATE
    LINE 01 LAST-NAME STATE ZIP PAY-NET *
REPORT-INPUT. PROC
    IF ZIP NE HOLD-ZIP
    HOLD-ZIP = ZIP
    SELECT
    END-IF
END-PROC *
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666

Produce:

```
11/23/86 REPORT FOR THE STATE OF IL PAGE 1

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>678.90</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>543.21</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td>1222.11</td>
<td></td>
</tr>
</tbody>
</table>

11/23/86 REPORT FOR THE STATE OF TX PAGE 2

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>666.66</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td>666.66</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1888.77</td>
</tr>
</tbody>
</table>
```
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.

When the display-page-size is greater than zero, the BEFORE-LINE procedure is executed, then the detail line is built. When the display-page-size is zero, the detail line for the report is already built, then the BEFORE-LINE procedure is executed. The second circumstance does not allow you to modify the contents of the detail line with the BEFORE-LINE procedure.

A BEFORE-LINE/AFTER-LINE procedure is commonly used to print an annotation before/after a detail line on the report.

The exhibit below illustrates how an AFTER-LINE procedure can cause information to be printed following a detail line of a report:

Statements:

FILE FILE1 CARD
LAST-NAME  1  5 A
STATE      6  2 A
ZIP        8  5 N
PAY-NET    13 5  N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 +
  DTLCTL EVERY
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET
*
AFTER-LINE. PROC
  IF PAY-NET GE 500
    DISPLAY '* EMPLOYEE ' LAST-NAME ' +
    EXCEEDED WEEKLY SALARY GOAL '*
  END-IF
END-PROC
*
END

BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666
Produce:

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>678.90</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>123.45</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>802.35</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>543.21</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>98.76</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>641.97</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>1444.32</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>666.66</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>111.11</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>777.77</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>2222.09</td>
</tr>
</tbody>
</table>
BEFORE-BREAK

A BEFORE-BREAK procedure can be used to calculate percentages and average totals. These values must be calculated immediately before printing.

The grand-total for percentage and average calculations is often maintained in S storage. TALLY is typically used as the number of items when calculating averages. The value of LEVEL (a system-defined field) can be used to determine which control break is being processed. Consider the following percentage calculation, paying special attention to when and how PERCENT is calculated:

```
FILE FILE1 CARD FB(80 8000)
LAST-NAME 1 5 A
STATE       6 2 A
ZIP         8 5 N
PAY-NET     13 5 N 2
  * 
PERCENT   W 2 N 2
TOTAL-NET  S 8 N 2
  * 
JOB INPUT FILE1 NAME MYPROG
  * 
  TOTAL-NET = TOTAL-NET + PAY-NET
  PRINT REPORT1
  *
REPORT REPORT1 LINESIZE 80 +
  SUMMARY SUMCTL DTLCOPY
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET PERCENT
  *
BEFORE-BREAK. PROC
  PERCENT = PAY-NET * 100 / TOTAL-NET
END-PROC
  *
END
BROWN  IL 60076  802.35  36.10
BROWN  IL 60077  641.97  28.89
  IL 1444.32  64.99
JONES  IL 60077  777.77  35.00
JONES  IL 60077  777.77  35.00
SMITH  TX 75218  222.09  100.00
SMITH  TX 75218  777.77  35.00
TX 777.77  35.00
2222.09  100.00

Produce:

<table>
<thead>
<tr>
<th>LAST-NAME</th>
<th>STATE</th>
<th>ZIP</th>
<th>PAY-NET</th>
<th>PERCENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60076</td>
<td>802.35</td>
<td>36.10</td>
</tr>
<tr>
<td>BROWN</td>
<td>IL</td>
<td>60077</td>
<td>641.97</td>
<td>28.89</td>
</tr>
<tr>
<td></td>
<td>IL</td>
<td></td>
<td>1444.32</td>
<td>64.99</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>777.77</td>
<td>35.00</td>
</tr>
<tr>
<td>JONES</td>
<td>IL</td>
<td>60077</td>
<td>777.77</td>
<td>35.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>222.09</td>
<td>100.00</td>
</tr>
<tr>
<td>SMITH</td>
<td>TX</td>
<td>75218</td>
<td>777.77</td>
<td>35.00</td>
</tr>
<tr>
<td></td>
<td>TX</td>
<td></td>
<td>777.77</td>
<td>35.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2222.09</td>
<td>100.00</td>
</tr>
</tbody>
</table>
**AFTER-BREAK**

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

Statements:

```
FILE FILE1 CARD
LAST-NAME  1  5 A
STATE      6  2 A
ZIP        8  5 N
PAY-NET    13 5 N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
  REPORT REPORT1 LINESIZE 65 +
     SUMMARY SUMCTL DTLCOPY
     SEQUENCE STATE ZIP LAST-NAME
     CONTROL STATE ZIP
  LINE 01 LAST-NAME STATE ZIP PAY-NET
  *
AFTER-BREAK. PROC
  IF LEVEL EQ 2
    DISPLAY 'TOTALS FOR THE STATE OF ' STATE
  END-IF
END-PROC
*
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666
```

Produce:

```
LAST-NAME STATE ZIP PAY-NET
BROWN IL 60076  802.35
JONES IL 60077  641.97
   IL            1444.32
TOTALS FOR THE STATE OF IL
SMITH TX 75218  777.77
   TX            777.77
TOTALS FOR THE STATE OF TX
                    2222.09
```
ENDPAGE

An ENDPAGE procedure can be used 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:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
JOB INPUT FILE1 NAME MYPROG
  PRINT REPORT1
  *
REPORT REPORT1 LINESIZE 65 +
  SUMMARY SUMCTL DLCOPY
  SEQUENCE STATE ZIP LAST-NAME
  CONTROL STATE NEWPAGE ZIP
  TITLE 'REPORT FOR THE STATE OF' STATE
  LINE 01 LAST-NAME STATE ZIP PAY-NET
  *
  ENDPAGE. PROC
  DISPLAY SKIP 2 '* CONFIDENTIAL - FOR INTERNAL USE ONLY *
  END-PROC
  *
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666
TERMINATION

A TERMINATION procedure is invoked at the end of the report. This procedure can be used to print report footing information, including control totals and distribution information. The exhibit below is an example of report footing:

FILE FILE1 CARD
LAST-NAME 1 5 A
STATE 6 2 A
ZIP 8 5 N
PAY-NET 13 5 N 2
TOTAL-NET 5 8 N 2
JOB INPUT FILE1 NAME MYPROG
   TOTAL-NET = TOTAL-NET + PAY-NET
PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65 +
   SUMMARY SUMCTL DTLCOPY
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE NEWPAGE ZIP
   TITLE 'REPORT FOR THE STATE OF' STATE
   LINE 01 LAST-NAME STATE ZIP PAY-NET
*
TERMINATION. PROC
   DISPLAY NEWPAGE
   DISPLAY SKIP 5 TOTAL-NET 'IS THE Y-T-D COMPANY NET PAY'
   DISPLAY SKIP 5 'PLEASE ROUTE THIS REPORT TO CORPORATE OFFICERS'
END-PROC
*
END
BROWNL6007612345
BROWNL6007667890
JONESL6007709876
JONESL6007754321
SMITHTX7521811111
SMITHTX7521866666
Report Work Files

CA-Easytrieve/Plus dynamically creates work files to temporarily store report information whenever:

- The report is sequenced using the SEQUENCE statement
- The report's printer file is already allocated to a previous report within the same job activity

Each work file record contains all of the data required to produce the report. The PRINT statements generate the work file. The order of occurrence of work file fields is the same as the field's reference occurrence in the REPORT statements. In the exhibit below, the underlined fields determine work file record contents:

```
FILE FILE1 CARD
   LAST-NAME  1  5 A
   STATE      6  2 A
   ZIP        8  5 N
   PAY-NET    13 5 N 2
JOB INPUT FILE1 NAME MYPROG
   PRINT REPORT1
*  REPORT REPORT1 LINESIZE 65 +
   DTLCTL EVERY
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE ZIP
   LINE 01 LAST-NAME STATE ZIP PAY-NET
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007709876
JONESIL6007754321
SMITHTX7521811111
SMITHTX7521866666
```

The DEBUG (DMAP) option automatically documents the contents of work files. This option causes a data map of the work file to be displayed. (Refer to Chapter 15, “System Facilities: DMAP.”)

By default, work files are written to the CA-Easytrieve/Plus system work file, xxxVFM (where xxx is the value of WKDSNPF in the options table). Refer to Appendix C, “Options Table,” for more information on this option. The algorithm for naming work files is xxxRnnn, where xxx is the value of the WKDSNPF option and nnn is the sequential number of the report within the job activity. A typical work file name for the first work file in a JOB is EZTR001.
You can save the work file contents for future processing by coding the FILE parameter on the REPORT statement. The corresponding FILE statement, coded in the library, must identify a SAM file. Also, you must specify that the work file's record length is at least as long as the dynamically created work file record. Records should be blocked to a reasonable value to ensure efficient processing.

Routing Printed Output

You can route reports to any printer. By default, CA-Easytrieve/Plus uses its system output printer (SYSPRINT for OS/390, or SYSLST for VSE). You can also route reports to any personal computer connected to CA-Corporate Tie. However, since most operating systems support multiple logical printers (spool files), you can realize significant performance improvements, if there is no SEQUENCE specified, by routing each output to a different logical printer.
Use the PRINTER parameter of the REPORT statement to route the printed report. The file named by this parameter corresponds to a library defined file. The FILE statement used to define the file must have the PRINTER option. To route the printed report to a PC using a CA-Corporate Tie Host Disk, add the Host Disk options to the FILE statement. The FORMAT option must be PRN. Additional Host Disk File information is in Chapter 12, “File Processing.” Unless otherwise designated, the record length of these files defaults based on the LINESIZ option. The exhibit below illustrates a program which takes advantage of print routing:

```
FILE PRINTR1 PRINTER F(121)
FILE PRINTR2 PRINTER F(251)
FILE FILE1 CARD
LAST-NAME  1  5 A
STATE      6  2 A
ZIP        8  5 N
PAY-NET    13 5 N 2
JOB INPUT FILE1 NAME MYPROG
   IF ZIP EQ 60076, 60077
       PRINT REPORT1
   ELSE
       PRINT REPORT2
   END-IF

* REPORT REPORT1 PRINTER PRINTR1 LINESIZE 120 +
   SUMMARY SUMCTL DLTCOPY
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE NEWPAGE ZIP
   TITLE 'REPORT FOR THE STATE OF' STATE
   LINE 01  LAST-NAME STATE ZIP PAY-NET
   *
* REPORT REPORT2 PRINTER PRINTR2 LINESIZE 250 +
   SUMMARY SUMCTL DLTCOPY
   SEQUENCE STATE ZIP LAST-NAME
   CONTROL STATE NEWPAGE ZIP
   TITLE 'REPORT FOR THE STATE OF' STATE
   LINE 01  LAST-NAME STATE ZIP PAY-NET
   *
END
BROWNIL6007612345
BROWNIL6007667890
JONESIL6007799876
JONESIL60077754321
SMITHTX7521811111
SMITHTX752186666
```

This CA-Easytrieve/Plus facility is an efficient way to separate output to different printer form types such as standard paper, labels, or pre-printed forms. Also, it can be used to create a report with a linesize larger than is permitted in the Options Table.
Introduction

CA-Easytrieve/Plus contains all the facilities necessary to process any file or database. Capabilities range from simple automatic input processing to complex controlled database maintenance.

CA-Easytrieve/Plus processes SAM, ISAM, VSAM, IMS/DLI, and CA-IDMS database files, CA-Corporate Tie Host Disk, and CA-Easytrieve/Plus virtual (VFM) files. User-written exit programs support all other file types.

This chapter describes file processing of sequential, random, and Host Disk files. Other chapters or guides discuss SQL, and special-case processes, such as PRINTER, TABLE, IMS/DLI databases, CA-IDMS databases, and file exits. Refer to the index for page references on these processes or to the database interface option guide for the database interface you are using.

Control of Input/Output

File input can be either automatic or controlled.

JOB and SORT statements designate automatic input. SORT and the SUMFILE parameter of REPORT (discussed in Chapter 11, “Report Processing”) are the only automatic output functions. All other output is under programmer control.

Input and output statements (GET, POINT, PUT, READ, and WRITE) enable controlled processing. You can code these statements in a JOB activity, with or without automatic input.
Automatic and controlled file processing interact according to the following rules:

1. Controlled statements are not permitted in SORT or REPORT procedures.
2. The GET statement cannot reference an automatic input file in the same JOB.
3. Controlled statements are not normally valid for automatic input files in the JOB activity, except as permitted by the following exceptions:
   - The POINT statement can be used with automatic input for VSAM and ISAM files. This interaction permits skip-sequential input processing while under system control.
   - The PUT and WRITE statements can be used with automatic input of a VSAM file, except when using synchronized file processing.

**Data Access Modes**

CA-Easytrieve/Plus uses one of two data access modes, depending on the type of file activity being performed:

1. In LOCATE mode, CA-Easytrieve/Plus accesses the logical record in the buffer area. CA-Easytrieve/Plus uses this access mode whenever possible, because it is the most efficient.
2. In MOVE mode, CA-Easytrieve/Plus moves the logical record from the buffer into a work area. CA-Easytrieve/Plus uses this mode whenever LOCATE mode is impractical (such as in the case of VSAM file update), or when WORKAREA is coded on the FILE statement.

**Record Format**

File records must be in one of three formats:

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

Fixed-length and variable-length records can be blocked.

Regardless of the record format chosen, all formats must adhere to the standards established for processing by IBM input/output control system routines. Those records that deviate from the standards for fixed-length or variable-length records can be processed only as undefined-length records.
CARD, PUNCH and VSAM

CA-Easytrieve/Plus makes the following assumptions about the format of CARD, PUNCH, and VSAM files:

1. CARD and PUNCH file records are always a fixed length; 80 characters long in OS/390, and 81 characters long in VSE.
2. VSAM records have undefined length.

When producing variable-length or undefined-length records, the length of the output record is controlled by the current contents of the output file's RECORD-LENGTH field. Unless otherwise specified, all records created by CA-Easytrieve/Plus have a maximum data length based on the file's record-size attributes.

Record Addressability

The address of a record points to the first data byte of the record. The record-control-word of variable-length records is accessible only through the system-defined RECORD-LENGTH field.

System-Defined File Fields

CA-Easytrieve/Plus automatically provides the special data fields listed below for each of your files.

These fields are stored as part of working storage but can be qualified by file-name. As working storage fields, they are not subject to invalid file reference errors.

1. RECORD-LENGTH – a two-byte binary field used for all file types to determine or establish the length of the current data record. For variable-length records, this field contains only the length of the record's data. That is, CA-Easytrieve/Plus automatically adjusts the field to account for the four-byte record-control-word and four-byte block-control-word. For variable-length files, assign the length of the record to the RECORD-LENGTH field before the PUT or WRITE statement is executed.
2. RECORD-COUNT – a read-only four-byte binary field which contains the number of logical I/O operations performed to the file.
3. FILE-STATUS – a read-only field which contains the results of the most recent I/O operation on a file:

- For DOS/ISAM files, the one-byte binary field is set to the contents of the DTF field file-nameC, described in the *IBM DOS Supervisor and I/O Macro Manual*.

- For OS/ISAM files, the two-byte binary field is set to the contents of the one-byte binary DCB fields, DCBEXCD1 and DCBEXCD2, described in the *IBM OS Data Management Macro Instructions Manual*. DCBEXCD1 is the left-most field of FILE-STATUS.

- For VSAM files, the four-byte binary field is set to the VSAM reason code value, RPLERRCD, when the return code, RPLRTNCD, is non-zero, as described in the *IBM VSAM Programmer's Guide*.

- For IMS/DLI files, the two-byte field is set to the value contained in bytes 11 and 12 of the PCB, following each call issued to IMS/DLI.

- For SQL files, the four-byte binary field is set to the value contained in SQLCODE.

4. CHKP-STATUS – a read-only two-byte field which is set to the value contained in bytes 11 and 12 of the I/O PCB following each DLI CHKP and DLI XRST call.

**Error Conditions**

Error conditions that arise during file processing activities generally fall into one of three categories:

1. File OPEN errors – commonly caused by incorrect or missing JCL information. The operating system detects these errors and terminates CA-Easytrieve/Plus processing.

2. Invalid file reference errors – caused by statements that refer to data from a file which does not have a current record (for example, after end-of-file or record not found). CA-Easytrieve/Plus issues a diagnostic message for these errors when the FLDCHK option is in effect.

3. Improper handling of nonzero STATUS conditions returned from statements such as READ. You are responsible for correctly handling these conditions. (Refer to the previous topic, “System Defined File Fields,” for information on FILE-STATUS.)
Data Availability Tests

CA-Easytrieve/Plus provides conditional expressions to assist you in resolving questions of data availability or currency. You can test these conditions after GET, POINT, READ, and WRITE statements and in association with synchronized file processing. The use of these conditionals is described later in this chapter. A full description of conditional statements appears in Chapter 8, “Decision and Branching Logic.”

Opening and Closing Files

During the initiation of a JOB or SORT activity, CA-Easytrieve/Plus opens all files used in the activity (except for those specified with the DEFER parameter). DEFERed files are opened when the first input/output statement is issued for them.

As part of its file-opening process, CA-Easytrieve/Plus performs the following tasks as needed:

- Validates block length for disk devices
- Sets the FULLTRK value
- Allocates buffer areas
- Allocates work areas
- Loads user exit programs.

All opened files are closed at the end of each activity. This establishes output files, for which no PUT was issued, as null files unless DEFER is coded.

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 activity.
2. You can create SAM files in one activity and process them by subsequent activities.
3. CA-Easytrieve/Plus permits only one CARD file in a program.
SAM Input

CA-Easytrieve/Plus provides both automatic and controlled processing of SAM files. The following examples illustrate how to process a SAM file using each facility.

Automatic Processing

```plaintext
FILE SEQFILE FB(150 1800)
%PERSNL
JOB INPUT SEQFILE NAME MYPROG
   PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65
   LINE EMP# NAME
```

Controlled Processing

```plaintext
FILE SEQFILE FB(150 1800)
%PERSNL
JOB INPUT NULL NAME MYPROG
   GET SEQFILE
   IF EOF SEQFILE
      STOP
   END-IF
   PRINT REPORT1
*
REPORT REPORT1 LINESIZE 65
   LINE EMP# NAME
```

Card Input

You can process one of the input files through the system input stream (SYSIN for OS/390, SYSIPT for VSE). “Execution only” card input is simply placed in the system input stream. When the mode of operation is syntax check, compile, and execute (the default), an END record must be placed between the CA-Easytrieve/Plus source statements and the card data.

```plaintext
FILE CARDFILE CARD
FIELD 1 5 A
JOB INPUT CARDFILE NAME MYPROG
   DISPLAY FIELD
END
CARD1
CARD2
CARD3
CARD4
CARD5
```
SAM Output

You can create output files under programmer control with the PUT statement.

Fixed-Length File Creation

FILE INFILE
FIELD 1 5 A
FILE OUTFILE FB(100 500)
FIELD 1 5 A
JOB INPUT INFILE NAME MYPROG
   PUT OUTFILE FROM INFILE

Variable-Length File Creation

Variable-Length SAM File Creation with Value

FILE CARDFILE CARD
FIELD 1 5 A
FILE OUTFILE VB(100 504)
FIELD 1 5 A
JOB INPUT CARDFILE NAME MYPROG
   OUTFILE:RECORD-LENGTH = 5
   PUT OUTFILE FROM CARDFILE
END
CARD1
CARD2
CARD3
CARD4
CARD5

Variable-Length SAM File Creation with Record-Length

FILE INFILE VB(100 504)
FIELD 1 5 A
FILE OUTFILE VB(100 504)
FIELD 1 5 A
JOB INPUT INFILE NAME MYPROG
   OUTFILE:RECORD-LENGTH = INFILE:RECORD-LENGTH
   PUT OUTFILE FROM INFILE

Note: In the above examples the actual record-length is 96 bytes plus the 4-byte RDW. The records are blocked at 5 for a value of 500 plus the extra 4 bytes for the BDW.
PUNCH Files

In a VSE environment, a PUNCH file is a special case of a SAM output file. Because the operating system has device-dependent constraints, PUNCH files require special consideration under VSE. You must specify the PUNCH parameter on the FILE statement to output records to the card punch.

Except for the PUNCH parameter, CA-Easytrieve/Plus treats PUNCH files the same as any other 80-byte SAM file. The following exhibit illustrates PUNCH file output:

```plaintext
FILE CARDFILE CARD
FIELD 1 5 A
FILE OUTFILE PUNCH
FIELD 1 5 A
JOB INPUT CARDFILE NAME MYPROG
   PUT OUTFILE FROM CARDFILE
END
CARD1
CARD2
CARD3
CARD4
CARD5
```

No consideration for files assigned to a card punch is required in OS/390 due to its device independence. Simply define an output SAM file as fixed-length 80 characters and assign it to the proper SYSOUT class using the DD card.

ISAM Files

CA-Easytrieve/Plus processes Indexed Sequential (ISAM) files as input only files. You can use sequential, skip-sequential, or random file processing. Sequential processing can be under the control of either the system or the programmer. Random processing is always under programmer control.

Automatic Processing

ISAM Sequential Input under Automatic Control

```plaintext
FILE ISAM IS
$PERSNL
JOB INPUT ISAM NAME MYPROG
   DISPLAY NAME
```
ISAM Sequential Input with a Starting Record

FILE ISAM IS
%PERSNL
JOB INPUT ISAM NAME MYPROG START POINT-PROC
   DISPLAY EMP# ' ' NAME
*
POINT-PROC. PROC
   POINT ISAM EQ '02200' STATUS
   IF FILE-STATUS NE 0
      DISPLAY 'FILE-STATUS= ', ISAM:FILE-STATUS
      STOP
   END-IF
END-PROC
*

Controlled Processing

FILE ISAM IS
%PERSNL
JOB INPUT NULL NAME MYPROG
GET ISAM
   IF EOF ISAM
      STOP
   END-IF
   IF ISAM:FILE-STATUS NE 0
      DISPLAY 'FILE-STATUS= ' FILE-STATUS
      GOTO JOB
   END-IF
   PRINT REPORT1
*
 REPORT REPORT1 LINESIZE 65
   LINE EMP# NAME
*

Skip-Sequential Processing

FILE ISAM IS
%PERSNL
JOB INPUT NULL NAME MYPROG
GET ISAM STATUS
   IF EOF ISAM
      STOP
   END-IF
   IF ISAM:FILE-STATUS NE 0
      DISPLAY 'FILE-STATUS= ' FILE-STATUS
      GOTO JOB
   END-IF
   IF EMP# EQ 1000 THRU 1999
      PERFORM POINT-ISAM
      GOTO JOB
   END-IF
   PRINT REPORT1
*
 POINT-ISAM. PROC
   POINT ISAM GE '02000' STATUS
   IF ISAM:FILE-STATUS NE 0
      DISPLAY 'FILE-STATUS= ' FILE-STATUS
   END-IF
END-PROC
*
 REPORT REPORT1 LINESIZE 65
   LINE EMP# NAME
Random Processing

```
* MVS VERSION *
FILE ISAMRAN IS
SPERSNL
FILE INKEYS CARD
WHO * 5 N
JOB INPUT INKEYS NAME MYPROG
READ ISAMRAN KEY WHO
IF FILE-STATUS EQ X'8000'
   DISPLAY 'BAD KEY = ' +1 WHO
   GOTO JOB
END-IF
DISPLAY SKIP 2 HEX ISAMRAN
END
01963
01730
03571
90909

* VSE VERSION *
FILE ISAMRAN IS
SPERSNL
FILE INKEYS CARD
WHO * 5 N
JOB INPUT INKEYS NAME MYPROG
READ ISAMRAN KEY WHO
IF FILE-STATUS EQ X'10'
   DISPLAY 'BAD KEY = ' +1 WHO
   GOTO JOB
END-IF
DISPLAY SKIP 2 HEX ISAMRAN
END
01963
01730
03571
90909
```

VSAM Files

CA-Easytrieve/Plus supports both sequential and random (direct) processing of VSAM files. The VSAM file organizations are:

- ESDS (entry-sequenced data sets)
- KSDS (key-sequenced data sets)
- RRDS (relative-record data sets).
File Creation

The FILE statement and the PUT statement are used to create (load) files. You can reference a newly created file in subsequent activities by coding another FILE statement with a different file-name, but whose JCL points to the same physical file. The exhibit below illustrates reloading a fixed-length ESDS. You can create KSDS and RRDS files using a similar technique, although the FILE statement is somewhat altered. The data set must be defined as reusable to use the RESET option on the FILE statement.

```plaintext
FILE ESDS VS (ES F CREATE RESET)
%PERSNL
FILE PERSNL FB(150 1800)
COPY ESDS
JOB INPUT PERSNL NAME MYPROG
  PUT ESDS FROM PERSNL STATUS
  IF ESDS:FILE-STATUS NE 0
    DISPLAY 'LOAD ERROR STATUS= ' ESDS:FILE-STATUS
    STOP
  END-IF
END-JOB
```

**Note:** When using multiple files, you should qualify FILE-STATUS. The above example shows ESDS:FILE-STATUS.

VSAM Input

CA-Easytrieve/Plus processes VSAM sequential input files automatically or under programmer-control. The following examples illustrate how to process a VSAM file using each facility.

**Automatic Processing**

VSAM Sequential Input under Automatic Control

```plaintext
FILE ESDS VS (ES F)
%PERSNL
JOB INPUT ESDS NAME MYPROG
  DISPLAY EMP# ' ' NAME
```

VSAM Sequential Input with a Starting Record

```plaintext
FILE KSDS VS
%PERSNL
JOB INPUT KSDS NAME MYPROG START POINT-PROC
  DISPLAY EMP# ' ' NAME
  POINT-PROC PROC
  POINT KSDS EQ '12318' STATUS
  IF FILE-STATUS NE 0
    DISPLAY 'FILE-STATUS= ', KSDS:FILE-STATUS
    STOP
  END-IF
END-JOB
```
VSAM Files

Controlled Processing

FILE ESDS VS (ES F)
%PERSNL
JOB INPUT NULL NAME MYPROG
GET ESDS STATUS
IF EOF ESDS
STOP
END-IF
IF ESDS:FILE-STATUS NE 0
   DISPLAY 'FILE-STATUS= ' ESDS:FILE-STATUS
   GOTO JOB
END-IF
DISPLAY EMP# ' ' NAME

Skip-Sequential Processing

Skip-sequential processing is accomplished by using the POINT statement with normal sequential processing.

FILE VSAM VS
%PERSNL
JOB INPUT VSAM NAME MYPROG
   IF EMP# EQ 1000 THRU 1999
      PERFORM POINT-VSAM
   END-IF
   PRINT REPORT1
*  
   POINT-VSAM PROC
      POINT VSAM GE '02000' STATUS
      IF VSAM:FILE-STATUS NE 0
         DISPLAY 'FILE-STATUS= ' FILE-STATUS
      END-IF
   END-PROC
*  
   REPORT REPORT1 LINESIZE 65
   LINE EMP# NAME

Random Input

KSDS and RRDS files can be input randomly (directly) by the READ statement.

FILE BADKEYS FB(150 1800) PRINTER
FILE VSAMRAN VS
%PERSNL
FILE INKEYS CARD
WHO * 5 N
JOB INPUT INKEYS NAME MYPROG
   READ VSAMRAN KEY WHO STATUS
   IF FILE-STATUS EQ 16
      DISPLAY BADKEYS 'BAD KEY = ' +1 WHO
      GOTO JOB
   END-IF
   DISPLAY SKIP 2 HEX VSAMRAN
END
01963
01730
03571
90909
Note: A key field for an RRDS file is defined as a four-byte binary field.

**VSAM Record Addition**

You can use the WRITE statement to add records to any established VSAM file. The WRITE statement adds a single record to the file, but to take advantage of VSAM's mass-sequential-insertion capabilities, you should use the PUT statement to add many records to the same place in the file.

If you use the WRITE or PUT statements, you must include the UPDATE parameter on the FILE statement. UPDATE informs CA-Easytrieve/Plus that all input records can potentially be updated or deleted. The following examples illustrate single and mass-insertion record addition.

**Single Record Addition**

```
FILE DUPKEYS FB(25 25) PRINTER
FILE FAILKEYS FB(25 25) PRINTER
FILE VSAMRAN VS (UPDATE)
&PERSNL
FILE INKEYS CARD
WHO * 5 N
PHONE * 10 N
JOB INPUT INKEYS NAME MYPROG
   MOVE WHO TO EMP#
   MOVE PHONE TO TELEPHONE
   WRITE VSAMRAN ADD STATUS
   IF VSAMRAN:FILE-STATUS EQ 8
      DISPLAY DUPKEYS 'DUP KEY =' +1 WHO
      GOTO JOB
   END-IF
   IF VSAMRAN:FILE-STATUS NE 0
      DISPLAY FAILKEYS 'FAIL KEY =' +1 WHO
      GOTO JOB
   END-IF
   DISPLAY SKIP 2 HEX VSAMRAN
END
666663123346591
```

**Mass-Sequential Insertion**

```
FILE ESDS VS (ES F)
&PERSNL
FILE PERSNL
COPY ESDS
JOB INPUT PERSNL NAME MYPROG
   PUT ESDS FROM PERSNL STATUS
   IF FILE-STATUS NE 0
      DISPLAY 'ADD FAILED'
      DISPLAY HEX PERSNL
      STOP
   END-IF
```
VSAM Files

VSAM Record Deletion

You can use the WRITE statement to delete individual records from either a KSDS or an RRDS VSAM file. The deleted record is the file’s current input record.

```plaintext
FILE KSDS VS (UPDATE)
FILE KEYS CARD
WHO 1 5 N
JOB INPUT KEYS NAME MYPROG
READ KSDS KEY WHO STATUS
IF FILE-STATUS NE 0
    DISPLAY 'READ FAILED...KEY=' WHO
    STOP
END-IF
WRITE KSDS DELETE STATUS
IF FILE-STATUS NE 0
    DISPLAY 'DELETE FAILED'
    STOP
END-IF
END
```

VSAM Record Update

You can modify and rewrite the current VSAM input record by using the WRITE statement.

```plaintext
FILE KSDS VS (UPDATE)
FILE KEYS CARD
WHO 1 5 N
PHONE 6 10 N
JOB INPUT KEYS NAME MYPROG
READ KSDS KEY WHO STATUS
IF FILE-STATUS NE 0
    DISPLAY 'READ #1 FAILED...KEY=' WHO
    STOP
END-IF
DISPLAY HEX KSDS
MOVE PHONE TO TELEPHONE
WRITE KSDS UPDATE STATUS
IF FILE-STATUS NE 0
    DISPLAY 'UPDATE FAILED...KEY=' WHO
    STOP
END-IF
READ KSDS KEY WHO STATUS
IF FILE-STATUS NE 0
    DISPLAY 'READ #2 FAILED...KEY=' WHO
    STOP
END-IF
DISPLAY HEX KSDS
END
```

66663125695999
Virtual File Manager

Virtual File Manager (VFM) is a sequential access method designed to serve the needs of program work files. Typically, when work files are needed within a program, separate disk areas must be reserved for each work file. VFM, however, maintains as much area in memory as possible. If the area in memory is exhausted, VFM writes the excess data to a single spill area. By using VFM, you only need to define one physical file.

As a virtual file is read back into the program, the space it occupied is released and the area can be immediately reused. You can, however, retain VFM files for subsequent CA-Easytrieve/Plus activities.

The use of VFM is identical to SAM processing with the following extra considerations:

- VFM files without the RETAIN option are deleted once CA-Easytrieve/Plus has processed them as input files.
- VFM files with the RETAIN option are deleted at the end of the associated CA-Easytrieve/Plus execution.
- VFM files are automatically blocked.

The following exhibit illustrates a typical use of the CA-Easytrieve/Plus VFM access method:

```plaintext
FILE PERSNL FB(150 1800)
PERSNL
FILE SORTFILE VIRTUAL F(150)
COPY PERSNL
SORT PERSNL TO SORTFILE USING PAY-NET NAME MYSORT
JOB INPUT SORTFILE NAME MYPROG
  PRINT REPORT1
* REPORT REPORT1
  LINE NAME PAY-NET
```

Since the file SORTFILE is a virtual file, you do not have to define it in the JCL.

Synchronized File Processing

The Synchronized File Processing (SFP) facility can be used with one file or multiple files.

- Synchronized File Input performs match/merge operations on multiple files.
- Single File Keyed Processing compares the contents of a key field or fields from one record to the next in a single file.
Synchronized File Input

CA-Easytrieve/Plus has a twofold solution to help you avoid coding complex logic for match/merge operations:

- Automatic input that includes a universally-adaptable match/merge algorithm
- Special conditional expressions that help to determine simple, yet precise file relationships.

The synchronized file match/merge algorithm is based on the following assumptions and rules:

- Two or more files capable of being processed sequentially can be accessed.
- All files involved in the operation must be in ascending order by their key values.
- The number of keys for each file must be identical.
- Corresponding keys of all files must be either alphanumeric or numeric. An alphanumeric key must be alphanumeric in all files, but can have different lengths. A numeric key must be numeric in all files, but can have different data types (N, P, U, B) and lengths.
- A maximum of 23 files can be used.
- Because the algorithm must “read ahead” to perform a match/merge, INDEXED, RELATIVE, and SQL files cannot be updated during synchronized file processing.
- You can use the POINT statement to position an ISAM or VSAM file at a record other than the first record before processing. Use a START procedure to perform the positioning.
- To override the default SELECT statement for an SQL file, code DEFER on the FILE statement to defer opening the file and place the SELECT statement in a START procedure.
Example

The INPUT parameter of the JOB statement designates files and their keys for synchronized file input. The exhibit below illustrates a variety of synchronized file and key combinations:

```
FILE FILE1   ...
  KEY1A  1  5  A
  KEY1B  6  4  P
  ...
  KEY1X   ...
...
FILE FILE2   ...
  KEY2A  24  5  A
  KEY2B  1  2  B
  ...
  KEY2X   ...
...
FILE FILEN   ...
  KEYNA  17  5  A
  KEYNB  10  7  N
  ...
  KEYNX   ...
...
JOB INPUT(FILE1 KEY KEY1A +
         FILE2 KEY KEY2A)
...
JOB INPUT (FILEN KEY(KEYNB, KEYNA) +
         FILE1 KEY(KEY1B, KEY1A) +
         FILE2 KEY(KEY2B, KEY2A))
...
JOB INPUT (FILE1 KEY(KEY1A ...) +
         ... +
         FILEN KEY(KEYNA ...))
...
```

Record Availability

Records from files in CA-Easytrieve/Plus synchronized file input are made available for processing based on the relationship of the files' keys. Records with the lowest keys are made available first and the match is hierarchical based upon the order of the files specified on the JOB statement.

Refer to the following three input files for an example of synchronized file input:

```
FILE1    FILE2    FILE3
  1       2       1
  2       3 A      3
  3 A      3 B      4
  3 B      4 A      5
  8 A      4 B      7
  8 B      6       8 A
  9       7       8 B
```

The key is the single numeric digit and the letter indicates duplicates for illustrative purposes.
The JOB statement to process the three files is:

```
JOB INPUT (FILE1 KEY(KEY1) +
          FILE2 KEY(KEY2) +
          FILE3 KEY(KEY3)) NAME MYPROG
```

Duplicate key values affect record availability differently based on which file contains the duplicates. Remember, the matching algorithm is hierarchical so the key is exhausted on the lowest level before another record is processed from the next higher level file.

The following exhibit illustrates the output from the synchronized file input process. The output shows the results of each iteration (loop) through the JOB activity. 'N/A' under a file indicates that a record from the file is not available and no fields from this file can be referenced during the associated iteration.

**Note:** CA-Easytrieve/Plus provides special IF statements to help you determine record availability. See “Special IF Statements” next.

### Match/Merge Operation Output

<table>
<thead>
<tr>
<th>JOB ITERATION</th>
<th>FILE1 RECORD</th>
<th>FILE2 RECORD</th>
<th>FILE3 RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>3A</td>
<td>3A</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3A</td>
<td>3B</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>3B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>4A</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>N/A</td>
<td>4B</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>N/A</td>
<td>N/A</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>N/A</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>N/A</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>8A</td>
<td>N/A</td>
<td>8A</td>
</tr>
<tr>
<td>12</td>
<td>8A</td>
<td>N/A</td>
<td>8B</td>
</tr>
<tr>
<td>13</td>
<td>8B</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Refer to iterations 3 through 5 in the above output. FILE1 and FILE2 both contain two records with a key value of 3. FILE3 contains only one record of key 3. These records are processed by CA-Easytrieve/Plus as follows:

- **Iteration 3:** The first record 3 from FILE1 and FILE2 and the only record with key 3 from FILE3 are available.

- **Iteration 4:** Since the next record on FILE3 is a key 4 record and there are still key 3 records to process in the other files, FILE3’s record is not available. CA-Easytrieve/Plus goes back to FILE2 and gets the next key 3 record. The original key 3 record from FILE1 and the second key 3 record from FILE2 are available.

- **Iteration 5:** Since the next record on FILE2 is a key 4 record and there is still a key 3 record on FILE1 to process, FILE2 is now unavailable. CA-Easytrieve/Plus returns to FILE1 and retrieves the next record. This time the only record available is the second key 3 record from FILE1.
**Special IF Statements**

CA-Easytrieve/Plus provides a simple way of determining the contents of current synchronized file input with special conditional expressions.

**MATCHED**

Use the MATCHED test to determine the relationship between the current record of one file and the current record of one or more other files.

`IF [NOT] MATCHED [file-name-1 ... file-name-2 ...]`

Refer to the Match/Merge Operation Output exhibit, shown above, which depicts automatic synchronized file input.

- IF MATCHED is true for JOB iteration 3.
- IF MATCHED FILE1, FILE3 is true for JOB iterations 1, 3, 11, and 12.
- IF MATCHED FILE2, FILE3 is true for JOB iterations 3, 6, and 10.

**File Existence**

To determine the presence of data from a particular file, use the following special conditional expressions:

`IF [NOT] file-name`

`IF [NOT] EOF file-name`

When the IF file-name condition is true, a record from that file is present and available for processing. The IF NOT file-name condition is true when the file does not contain a record with a current key. When this condition is true, no fields from the file can be referenced in the activity. If you reference a field in an unavailable file, CA-Easytrieve/Plus issues a runtime error.

Refer again to the automatic input example (Match/Merge Operation Output exhibit, shown above).

- IF FILE1 is true for JOB iterations 1 through 5 and 11 through 14.
- IF NOT FILE2 is true for JOB iterations 1, 5, 8, and 11 through 14.
- IF EOF FILE2 is true for JOB iterations 11 through 14.
DUPLICATE, FIRST-DUP, and LAST-DUP

The DUPLICATE, FIRST-DUP, and LAST-DUP tests determine the relationship of the current record of a file to the preceding and following records in the same file:

{DUPLICATE}
IF [NOT] {FIRST-DUP} file-name
{LAST-DUP }

The following record relationship tests are based on the previous example of automatic synchronized file input (Match/Merge Operation Output exhibit, shown above).

- IF DUPLICATE FILE1 is true for JOB iterations 3 through 5 and 11 through 13.
- IF FIRST-DUP FILE2 is true for JOB iterations 3 and 6.
- IF LAST-DUP FILE3 is true for JOB iteration 12.

Note: The FIRST-DUP and LAST-DUP conditions are also DUPLICATE conditions. A record that satisfies the IF LAST-DUP or IF FIRST-DUP condition also satisfies the IF DUPLICATE condition.

Refer to Chapter 8, “Decision and Branching Logic” for more detailed examples of conditional expressions.

Updating a Master File

The exhibit shown next illustrates updating a master file based upon matching transaction file records. The program makes the following assumptions:

- A new master record is written when a match exists between the master file and the transaction file.
- There should be no duplicate transactions for a given master record. If this occurs, the first duplicate is processed but subsequent duplicates are bypassed.
- No transaction records should exist without a matching master record. If this occurs, the record is displayed on an error report and processing is bypassed.
FILE OLDSTR SEQUENTIAL
  O-KEY 1 2 N
  O-AMT 3 3 N
FILE TRANS SEQUENTIAL
  T-KEY 1 2 N
  T-AMT 3 3 N
FILE NEWSTR SEQUENTIAL
  N-KEY 1 2 N
  N-AMT 3 3 N

JOB INPUT (OLDSTR KEY(O-KEY) +
           TRANS KEY(T-KEY)) NAME MYPROG
  * FOR Matched: Update with Tran AMT and put NewSTR.
  * If Tran IS a Duplicate But NOT the First. BYPASS the Record.
    IF Matched
      IF Duplicate TRANS and Not First-Dup TRANS
        GOTO JOB
      END-IF
      MOVE O-KEY TO N-KEY
      N-AMT = O-AMT + T-AMT
      PUT NEWSTR
      GOTO JOB
      END-IF
  * On OldSTR Only: Put the NewSTR Without ANY Update.
    IF OldSTR
      PUT NewSTR FROM OldSTR
      GOTO JOB
    END-IF
  * On Trans Only: Print Error REPORT.
    IF TRANS
      PRINT Error-RPT
      GOTO JOB
    END-IF
  *
    REPORT Error-RPT
    TITLE 'REPORT OF TRANSACTION WITH INVALID KEYS'
    LINE T-KEY T-AMT

**Single File Keyed Processing**

Using Synchronized File Processing on a single file enables you to compare the
contents of a key field or fields from one record to the next and use IF tests to
group records according to the key fields. The file name is coded on the JOB
INPUT statement as follows:

JOB INPUT (filename  KEY (keyfield...))

Using single file input enables you to determine the start of a new key value and
the end of the current key value by use of IF tests.

The following IF statement determines the start of a new key.

IF FIRST-DUP filename OR NOT DUPLICATE filename.

The following IF statement determines the end of the current key.

IF LAST-DUP filename OR NOT DUPLICATE filename.

**Note:** The file must be in ascending order by its key value(s).
Host Disk Files

CA-Corporate Tie is a Computer Associates product that provides a transparent link between the mainframe and Personal Computers (PCs). It enables you to:

- Access data from the mainframe
- Send data to the mainframe
- Share data with other microcomputer users in your network.

CA-Easytrieve/Plus provides a method to access data on the mainframe and converts it from mainframe format (EBCDIC) to popular PC formats. The converted data is then stored in the CA-Corporate Tie Host Disk. The data can be routed to one or more PC users in your network.

The CA-Corporate Tie External Security Interface requires that the CA-Easytrieve/Plus LOADLIB be APF authorized and EZTPA00 be linked with an authorization code of 1.

Refer to the CA-Corporate Tie documentation for specific information about operating CA-Corporate Tie.

Host Disk Definition

To transfer files to CA-Corporate Tie Host Disk, define a file with a file type of HOSTDISK. Refer to Chapter 5, “Data Definition,” for complete details on the Host Disk file statement.

You define a field to describe the data you want CA-Corporate Tie to convert. CA-Corporate Tie uses a field’s description to correctly convert its data into a PC format. Only fields you defined in the Library Section for a Host Disk file are converted. Those fields defined after the first JOB or SORT statement for a Host Disk file are invalid. Do not define a field that creates an overlay redefinition, either implicitly or explicitly. Such a definition is invalid. An explicit overlay redefinition is:

```
DEFINE  NAME         17            20  A
DEFINE  LAST-NAME  NAME +10        10  A
```

An implicit overlay redefinition is:

```
DEFINE  NAME         17            20  A
DEFINE  LAST-NAME    27            10  A
```

An overlay redefinition can also be defined by a COPY filename statement. To permit such definitions would duplicate and incorrectly convert data.
To send the output of a DISPLAY statement or a REPORT statement to a Host Disk file, code the PRINTER keyword and the HOSTDISK keyword on the FILE statement. When using a HOSTDISK PRINTER file, do not define any fields for it. CA-Easytrieve/Plus automatically defines the fields needed to convert the output into PRN format. PRN is the only permissible format when PRINTER is specified.

### Host Disk Formats

For each format, the order of translated fields in the PC file matches their order of appearance in the CA-Easytrieve/Plus source code. A VARYING field is sent as two fields, a two-byte binary field with zero decimals followed by an alphanumeric field of the remaining length.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOTUS</td>
<td>A PC file matching 123 format WKS is produced.</td>
</tr>
<tr>
<td>DBASEIII</td>
<td>A PC file matching the format of DBASEIII files is produced.</td>
</tr>
<tr>
<td>BASIC</td>
<td>A comma delimited PC file is produced. Alphanumeric fields are enclosed in quotes.</td>
</tr>
<tr>
<td>PRN</td>
<td>A fixed length PRN is produced. The starting position of each field in the PRN record is determined by the sum of the maximum lengths of the PC format fields placed before it.</td>
</tr>
<tr>
<td>EZTPC</td>
<td>A CA-Easytrieve/Plus PC data file in EBCDIC format.</td>
</tr>
</tbody>
</table>

### Writing to the Host Disk File

There are five methods you can use to write data to a Host Disk file:

- You can use the PUT statement to extract data from a mainframe file and download it to a PC. You can move data into the Host Disk file using the Assignment statement, MOVE statement, MOVE LIKE statement, or PUT FROM statement. You have the full power of CA-Easytrieve/Plus to select and manipulate the data that you want to write to the Host Disk file.

- You can specify a Host Disk file as the output file of the SORT statement. This enables you to control the sequence of the records written to the Host Disk file. You can use a SORT BEFORE procedure to select records to be sorted and written to the Host Disk file.

- You can specify the Host Disk file as a PRINTER file (using the PRINTER keyword on the FILE statement) and have CA-Easytrieve/Plus print REPORT output to the file. Specify the Host Disk file as the PRINTER file on the REPORT statement to cause all REPORT output to be written to the Host Disk file.
Similarly, if the Host Disk file is also specified as a PRINTER file on the FILE statement, you can use the DISPLAY statement to print data to the file.

You can specify a Host Disk file as the SUMFILE on a REPORT statement. This causes CA-Easytrieve/Plus to write the resulting summary file to the Host Disk. You must code DEFINE statements for the Host Disk file that match the data created for the summary file. See Chapter 11, “Report Processing,” for details on summary file creation.

**Note:** You must define fields for a report summary file. The CA-Easytrieve/Plus system-defined fields are not known to CA-Corporate Tie. Only fields that you define are converted to PC format.

### Host Disk File Processing

In the Library Section, you define a Host Disk file, its parameters, and its fields. Those field definitions and the FORMAT parameter tell CA-Corporate Tie how to correctly convert the mainframe data to PC data. CA-Corporate Tie uses the other parameters, TO, FROM and HOSTFILE to manage Host Disk access.

In the Activity Section, you must assign values to all field parameters of a Host Disk file before CA-Easytrieve/Plus executes the first PUT for that file. The TO parameters are mandatory and are required to be fields. The START procedure is a convenient place to set values for parameter fields.

Also, in the Activity Section, you can code MOVEs and assignments to fill the Host Disk fields. When each field has a value, you issue a PUT. CA-Corporate Tie takes the record, converts each field to its PC format in a new record, and writes the new record to the Host Disk under the HOSTFILE name.

CA-Easytrieve/Plus copies Host Disk file parameters for CA-Corporate Tie. CA-Easytrieve/Plus copies the parameters once in each JOB or SORT activity that uses the file. CA-Easytrieve/Plus copies the parameters when it opens the file, unless DEFER was coded on the file. If DEFER was coded, then CA-Easytrieve/Plus copies the parameters when the first PUT is issued.

When the first PUT is issued, CA-Corporate Tie examines the Host Disk parameters. CA-Corporate Tie checks the FROM userid and password. It verifies that each userid or groupid in the TO recipient list exists. If a userid or groupid exists, then CA-Corporate Tie marks the corresponding entry in the TO recipient list with a Y. If a recipient does not exist, then CA-Corporate Tie marks the corresponding entry with an N.

Following the PUT, you test if this is the first time the PUT is executed (for example, IF file-name:RECORD-COUNT = 1).

If this is the first time, you examine each entry in the checklist for an N. If you find an N, you can display an error message and, possibly, stop the program. It is recommended you examine the checklist, but it is not required.
If all of the destinations are invalid, an open error A038-code 164 is issued.

When you use a Host Disk file as a REPORT PRINTER file or a REPORT SUMFILE you should define any Host Disk parameter field as an S working storage field. Within a REPORT procedure, any field you use to test or process, the Host Disk parameter fields should also be defined as an S working storage field. The STOP statement is not permitted in a REPORT procedure.

If the HOSTDISK parameter of a Host Disk file identifies an existing file, CA-Corporate Tie does not overwrite the file on the Host Disk, unless RESET is coded on the FILE statement. Without RESET, CA-Corporate Tie creates an additional file with the same name on the Host Disk.

Examples

Static Parameter Specification

This example demonstrates the use of static Host Disk parameters. The parameters on the Host Disk file are declared as literals. The TO parameters are required to be fields.

FILE PERSNL
DEFINE REGION 1 1 N
DEFINE NAME 17 20 A
DEFINE NET 90 4 P 2
DEFINE GROSS 94 4 P 2
DEFINE PAM S 8 A VALUE 'PAM'
DEFINE CHECKLIST S 1 A

FILE PC123 HOSTDISK ( TO ( PAM CHECKLIST ) +
FROM ( 'JON' 'GUAC' ) +
HOSTFILE ( 'PC123.DAT' ) +
FORMAT ( 'LOTUS' ) )

DEFINE REGION 1 1 N
DEFINE NAME 5 20 A
DEFINE NET 25 4 P 2
DEFINE GROSS 29 4 P 2

JOB INPUT PERSNL NAME SPREADSHEET
MOVE LIKE PERSNL TO PC123
PUT PC123

In this example, CA-Corporate Tie user JON sends a LOTUS format copy of the PERSNL file to the Host Disk. There, it waits for CA-Corporate Tie user PAM to receive it to her PC.
Dynamic Parameter Specification

The example below demonstrates the use of dynamic Host Disk parameters. The parameters on the Host Disk file are declared as fields. The fields used on the Host Disk file are supplied their values at execution time from a card file.

```
FILE PARMS CARD
  DEFINE PARM-FROM  1  8  A
  DEFINE PARM-PW   9  8  A
  DEFINE PARM-HDFILE  17  12  A
  DEFINE PARM-FORMAT  29  8  A
  DEFINE PARM-TO   40  8  A

  DEFINE WORK-FROM S  8  A
  DEFINE WORK-PW S  8  A
  DEFINE WORK-HDFILE S  12  A
  DEFINE WORK-FORMAT S  8  A
  DEFINE WORK-TO  S  8  A  OCCURS 50
  DEFINE WORK-CHECKLIST S  1 A  OCCURS 50
  DEFINE WORK-COUNT S  2 P 0 VALUE 1
  DEFINE ERR-COUNT S  2 P 0

FILE PERSNL
  DEFINE REGION 1 1 N
  DEFINE NAME 17 20 A
  DEFINE NET 90 4 P 2
  DEFINE GROSS 94 4 P 2

FILE PC123 HOSTDISK ( TO ( WORK-TO WORK-CHECKLIST ) +
                        FROM ( WORK-FROM WORK-PW ) +
                        HOSTFILE ( WORK-HDFILE ) +
                        FORMAT ( WORK-FORMAT ) )

  DEFINE REGION 1 1 N
  DEFINE NAME 5 20 A
  DEFINE NET 25 4 P 2
  DEFINE GROSS 29 4 P 2

JOB INPUT PERSNL NAME SPREADSHEET START INIT
  MOVE LIKE PERSNL TO PC123
  PUT PC123
  IF PC123:RECORD-COUNT = 1
    PERFORM CHECKLIST
  END-IF

CHECKLIST. PROC
  DO WHILE WORK-COUNT GT 0
    IF WORK-CHECKLIST(WORK-COUNT) EQ 'N'
      DISPLAY 'CORPORATE TIE does not know userid' +
             WORK-TO(WORK-COUNT) '!
      ERR-COUNT = ERR-COUNT + 1
    END-IF
    WORK-COUNT = WORK-COUNT - 1
  END-DO
  IF ERR-COUNT GT 0 STOP
  END-IF
END-PROC

INIT. PROC
  GET PARMS
  IF EOF PARMS
    DISPLAY 'No records in PARMS file!' STOP
  END-IF
```
WORK-FROM = PARM-FROM
WORK-PW = PARM-PW
WORK-HDFILE = PARM-HDFILE
WORK-FORMAT = PARM-FORMAT

DO WHILE WORK-COUNT LE 50 AND NOT EOF PARMS
   IF PARM-TO NOT SPACES
      WORK-TO(WORK-COUNT) = PARM-TO
      WORK-COUNT = WORK-COUNT + 1
   END-IF
   GET PARMS
END-DO
WORK-COUNT = WORK-COUNT + 1
IF WORK-COUNT GT 50
   DISPLAY 'More than 50 Host Disk file recipients!'
   STOP
END-IF
END-PROC
END

JON GUAC PC123.DAT LOTUS PAM
RON
ED
MARTHA

This example provides identical processing to the previous example except all parameters for the Host Disk file are received from a card file. This lets you change the parameters simply by changing the input cards, instead of modifying and recompiling the CA-Easytrieve/Plus source statements. The parameters are set in the INIT proc. CA-Easytrieve/Plus does not access them until the PUT statement is executed.
Sending Report Output to the Host Disk

The example below illustrates how the output from a CA-Easytrieve/Plus report can be sent to a Host Disk file.

FILE PERSNL
DEFINE REGION 1 1 N
DEFINE NAME 17 20 A
DEFINE NET 90 4 P 2
DEFINE GROSS 94 4 P 2
DEFINE PAM S 8 A VALUE ‘PAM’
DEFINE CHECKLIST S 1 A

FILE PCRPT PRINTER HOSTDISK ( TO ( PAM CHECKLIST ) +
FROM ( ‘JON’ ‘GUAC’ ) +
HOSTFILE ( ‘REGION.RPT’ ) +
FORMAT ( ‘PRN’ ) )

JOB INPUT PERSNL NAME PCREPORT
PRINT RPT
REPORT RPT PRINTER PCRPT
SEQUENCE REGION NAME
CONTROL REGION
TITLE 01 ‘Personnel Listing by Region’
LINE 01 REGION NAME GROSS

ENDPAGE. PROC
    DISPLAY ‘*** Confidential - Internal Use Only ***’
END-PROC

This example shows how printed output can be sent to a Host Disk file.

Note: The Host Disk FILE statement uses the PRINTER keyword to indicate a print output file. The REPORT PRINTER subparameter references the Host Disk file as the target file for the report. The ENDPAGE procedure demonstrates how you can annotate the report. The Host Disk file receives the DISPLAY statement output as it is part of the report.
Introduction

Chapter 13 discusses the CA-Easytrieve/Plus facilities for processing the two basic categories of tabular information:

- File oriented table information processed by the SEARCH statement.
- Arrays, segmented data, and data strings processed through subscripting or index manipulation.

First, this chapter explains how a table is defined and how it is identified, stressing the role of the SEARCH statement in processing file oriented table information.

Second, this chapter discusses how, through subscripting or index manipulation, one dimension and multiple dimension arrays can be defined and processed. Also, it describes segmented data and data string processing.

Table Definition

A table is a collection of uniform data records that presents unique processing opportunities. All tables have two parts:

<table>
<thead>
<tr>
<th>Parts</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argument</td>
<td>Uniquely identifies a table entry.</td>
</tr>
<tr>
<td>Description</td>
<td>Remainder of the table entry.</td>
</tr>
</tbody>
</table>

Some typical examples of table usage include organization structures, parts list for assembly processes, and accounting chart-of-accounts.

The dichotomized (binary) search of CA-Easytrieve/Plus is extremely efficient. Therefore, table use is recommended for applications which need to validate encoded data and/or retrieve code description.
Defining Tables

There are two types of tables which can be specified on the FILE statement:

1. Instream - (specified by the INSTREAM parameter on the TABLE option) directs CA-Easytrieve/Plus to look for table data within your program immediately following the FILE statement. This table is established at the time your program is compiled. Its size is limited only by the amount of available memory.

2. External - (specified by the literal-10 parameter on the TABLE option) indicates that your table is located in a file external to your program. This file must be sequentially accessible. An external table is established just before use. Literal-10 specifies the maximum number of entries.

An external table can be:

a. An existing file that is in EBCDIC ascending order by its search argument.

b. Created by specifying the name of the table as the TO file-name parameter in a SORT activity.

All data needed to create small tables (to be processed by the SEARCH statement) can be entered instream along with CA-Easytrieve/Plus statements; that is, the table data can immediately follow the library definition statements for the table. The data is delimited by the word ENDTABLE in the first eight positions of a record.

Instream data is 80 characters per record and is unaffected by the SCANCOL option. An instream table can be retrieved from a macro file. This provides easy access to common information. The macro must contain the entire table definition (FILE statement through ENDTABLE). The exhibit below illustrates the table-of-days definition:

```
FILE  DAYTABL  TABLE   INSTREAM
ARG 1    1  A.   DESC   3  9  A
1 SUNDAY   }
2 MONDAY   }
...       }      (instream data)
7 SATURDAY }
ENDTABLE   }
```

The only way to modify an instream table is to recompile the program while supplying new table data. However, you can modify external tables without program change because CA-Easytrieve/Plus builds these tables dynamically just prior to each use.

All tables must be sorted in EBCDIC ascending order by their search argument. No duplicate search arguments are allowed. Table sequence is validated as the table is established.
The only fields defined for table files are ARG (argument) and DESC (description). ARG defines the field used when searching the table. DESC defines the field that contains the desired information. The maximum length for an alphanumeric ARG or DESC field is 254 bytes.

**Note:** All records between the ARG and DESC definitions and the ENDTABLE statement are used as instream table data. For example, a record starting with an asterisk is handled as TABLE data, not a comment. ENDTABLE is only used to delimit instream table data. Do not include ENDTABLE in your external table data.

The exhibit below illustrates a typical table file description. The resulting table provides descriptions of a hypothetical high school curriculum:

```
1011  ENGLISH 1   }
1012  ENGLISH II  }
...                              }
9712  HOME ECONOMICS }
---------------------------------------
FILE   CLASSES TABLE (150)...
      ARG 1  4  A.    DESC 10  40  A
```

**SEARCH Statement**

The SEARCH statement provides access to table information. Special conditions of the IF statement can be used to validate the results of SEARCH operations. The syntax of the SEARCH statement is:

**Syntax**

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

**Operation**

*file-name*

File-name is the name of the file which describes the table and its source. The file must have the TABLE attribute and must be a fixed length.
**SEARCH Statement**

**WITH field-name-1**

The WITH field-name-1 parameter identifies the field containing the search argument for the binary search. This parameter is defined in any file, except for files with the TABLE attribute or it can be defined in working storage.

The length and field type of field-name-1 must match the length and field type of the ARG field defined for file-name. If field-name-1 contains DBCS data, then before making the search, CA-Easytrieve/Plus converts field-name-1 into the DBCS code system of file-name. When performing the binary search of the table with a DBCS key, CA-Easytrieve/Plus performs a bit by bit comparison.

**GIVING field-name-2**

The GIVING field-name-2 parameter identifies the receiving field for the results of the table search. This parameter is defined in any file, except for files with the TABLE attribute or it can be defined in working storage.

The length and field type of field-name-2 must match the length and field type of the DESC field defined for file-name. If field-name-2 contains DBCS data then the result of the search is converted from the DBCS code system of file-name into the DBCS code system of field-name-2.

After each SEARCH statement, you can code an IF file-name test to determine the success of the table search. When the search is successful (IF file-name is true), field-name-2 contains table descriptive data corresponding to the search argument of field-name-1. When the search is unsuccessful (IF file-name is false), the contents of field-name-2 are unchanged.
Searching Tables

You can code SEARCH statements any place within a JOB activity, and issue any number of SEARCHes against any number of tables.

The exhibit below illustrates the retrieval of high school class descriptions, based on class identification codes.

Statements:

```
DEFINE CODE          W   4 A
DEFINE DESCRIPTION   W  40 A
FILE CLASSES TABLE INSTREAM
ARG   1  4 A
DESC 10 40 A
1011     ENGLISH I
1012     ENGLISH II
1013     ENGLISH III
1014     ENGLISH IV
ENDTABLE
JOB INPUT NULL NAME MYPROG
  MOVE '1012' TO CODE
  SEARCH CLASSES WITH CODE, GIVING DESCRIPTION
  IF CLASSES
    DISPLAY DESCRIPTION
  ELSE
    DISPLAY 'CLASS NOT FOUND'
  END-IF
STOP
```

Produce:

ENGLISH II

Single Dimension Arrays

An array is a series of consecutive memory locations in one or more dimensions. You can process identical elements in arrays by using either subscripting or index manipulation.

Index Attribute

Any data field definition can contain the INDEX attribute. An index can be used to reference data fields which occur multiple times. If you do not use an index, you must either use subscripts or assign individual field names to multiple field occurrences.

The data field starting location is adjusted by the contents of its indexes to determine the desired field occurrence. The INDEX indexname value is set to:

\[(\text{desired occurrence number} - 1) \times (\text{length of element})\]
Example

The following one-dimension array is typical of those found in most programs. Data definition is straightforward. The value of MONTH-INDEX controls access to the desired data occurrence, MONTH.

Statements:

```
DEFINE ARRAY-ELEMENT W 2 N
DEFINE MONTHS       W 120 A VALUE +
  'JANUARY  +
  FEBRUARY +
  MARCH   +
  APRIL   +
  MAY     +
  JUNE    +
  JULY    +
  AUGUST  +
  SEPTEMBER +
  OCTOBER +
  NOVEMBER +
  DECEMBER '
DEFINE MONTH       MONTHS 10 A +
  OCCURS (12) INDEX (MONTH-INDEX)
JOB INPUT NULL NAME MYPROG
ARRAY-ELEMENT = 11
  MONTH-INDEX = (ARRAY-ELEMENT - 1) * 10
  DISPLAY MONTH
STOP
```

Produce:

```
NOVEMBER
```

Since MONTHS is 10 bytes long, the following relationships are true:

<table>
<thead>
<tr>
<th>ARRAY-ELEMENT</th>
<th>MONTH-INDEX</th>
<th>DATA OCCURRENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>JANUARY</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>FEBRUARY</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>MARCH</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>110</td>
<td>DECEMBER</td>
</tr>
</tbody>
</table>
Multiple Dimension Arrays

Multiple dimension arrays can be defined in two different ways. You can define a single field with multiple indexes, or you can index a redefining field, as well as the parent field.

The exhibit below illustrates two arrays which are identical in size and usage, but are defined very differently.

<table>
<thead>
<tr>
<th>MONTH-INDEX-1</th>
<th>MONTH-INDEX-2</th>
<th>MONTH</th>
<th>ROW-INDEX</th>
<th>COL-INDEX</th>
<th>MONTH-CELL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>JANUARY</td>
<td>0</td>
<td>0</td>
<td>JANUARY</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>FEBRUARY</td>
<td>0</td>
<td>10</td>
<td>FEBRUARY</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>MARCH</td>
<td>0</td>
<td>20</td>
<td>MARCH</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>APRIL</td>
<td>30</td>
<td>0</td>
<td>APRIL</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>MAY</td>
<td>30</td>
<td>10</td>
<td>MAY</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>JUNE</td>
<td>30</td>
<td>20</td>
<td>JUNE</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>JULY</td>
<td>60</td>
<td>0</td>
<td>JULY</td>
</tr>
<tr>
<td>60</td>
<td>10</td>
<td>AUGUST</td>
<td>60</td>
<td>10</td>
<td>AUGUST</td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>SEPTEMBER</td>
<td>60</td>
<td>20</td>
<td>SEPTEMBER</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
<td>OCTOBER</td>
<td>90</td>
<td>0</td>
<td>OCTOBER</td>
</tr>
<tr>
<td>90</td>
<td>10</td>
<td>NOVEMBER</td>
<td>90</td>
<td>10</td>
<td>NOVEMBER</td>
</tr>
<tr>
<td>90</td>
<td>20</td>
<td>DECEMBER</td>
<td>90</td>
<td>20</td>
<td>DECEMBER</td>
</tr>
</tbody>
</table>

In both cases, the sum of the indexes determines which data occurrence is referenced. Both MONTH and MONTH-CELL are 10-character fields with two indexes.

Both fields also occur 12 times. MONTH-INDEX-1 and ROW-INDEX, and MONTH-INDEX-2 and COL-INDEX are considered similar indexes.
You can define and access arrays of more than two dimensions by a simple extension of the following examples:

Statements:

```
DEFINE QUARTER-ROW W 2 N
DEFINE MONTH-COL   W 2 N
DEFINE MONTHS      W 120 A VALUE +
  'JANUARY  +
  FEBRUARY +
  MARCH   +
  APRIL   +
  MAY     +
  JUNE    +
  JULY    +
  AUGUST  +
  SEPTEMBER +
  OCTOBER +
  NOVEMBER +
  DECEMBER '
DEFINE MONTH  MONTHS 10 A OCCURS (12) +
  INDEX (MONTH-INDEX-1, MONTH-INDEX-2)
JOB INPUT NULL NAME MYPROG
  QUARTER-ROW = 4
  MONTH-COL   = 2
  MONTH-INDEX-1 = (QUARTER-ROW - 1) * 30
  MONTH-INDEX-2 = (MONTH-COL - 1) * 10
  DISPLAY MONTH
  STOP
```

<table>
<thead>
<tr>
<th>JANUARY</th>
<th>FEBRUARY</th>
<th>MARCH</th>
<th>← Quarter-Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>APRIL</td>
<td>MAY</td>
<td>JUNE</td>
<td>← Quarter-Row</td>
</tr>
<tr>
<td>JULY</td>
<td>AUGUST</td>
<td>SEPTEMBER</td>
<td>← Quarter-Row</td>
</tr>
<tr>
<td>OCTOBER</td>
<td>NOVEMBER</td>
<td>DECEMBER</td>
<td>← Quarter-Row</td>
</tr>
</tbody>
</table>

produce:

NOVEMBER
Statements:

DEFINE QUARTER-ROW W 2 N
DEFINE MONTH-COL W 2 N
DEFINE MONTHS W 120 A VALUE +
    'JANUARY +
    FEBRUARY +
    MARCH +
    APRIL +
    MAY +
    JUNE +
    JULY +
    AUGUST +
    SEPTEMBER +
    OCTOBER +
    NOVEMBER +
    DECEMBER'
DEFINE MONTH MONTHS 10 A +
    OCCURS (12)
DEFINE MONTH-ROW MONTH 30 A, +
    OCCURS 4, INDEX (ROW-INDEX)
DEFINE MONTH-COLS MONTH-ROW 10 A, +
    OCCURS 3, INDEX (COL-INDEX)
DEFINE MONTH-CELL MONTH-COLS 10 A
JOB INPUT NULL NAME MYPROG
    QUARTER-ROW = 4
    MONTH-COL = 2
    ROW-INDEX = (QUARTER-ROW - 1) * 30
    COL-INDEX = (MONTH-COL - 1) * 10
    DISPLAY MONTH-CELL
    STOP
Subscripts

Subscripts are an alternate method available to select an individual element from an array. The use of subscripts removes from you the requirement of computing the index value, and places it on CA-Easytrieve/Plus. In order for CA-Easytrieve/Plus to be able to perform the index computation, certain restrictions must be placed on the way arrays are defined.
Defining a One Dimension Array

A one-dimension array is defined just as it would be if indexing were to be used. Referring back to the Single Dimension Arrays Example, the exhibit below illustrates the relationship between the array element and the corresponding array element value:

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>Is</td>
</tr>
<tr>
<td>MONTH(1)</td>
<td>JANUARY</td>
</tr>
<tr>
<td>MONTH(2)</td>
<td>FEBRUARY</td>
</tr>
<tr>
<td>MONTH(3)</td>
<td>MARCH</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>MONTH(12)</td>
<td>DECEMBER</td>
</tr>
</tbody>
</table>

For this array the maximum value to be specified for the occurrence number is 12.

Defining a Two-Dimensional Array

A two-dimensional array is somewhat more complicated. To define a two-dimensional array, you must define the length and number of occurrences of each dimension. The next two exhibits illustrate this as follows:

```
DATA W               30 A  VALUE 'AA+ BB+ CC+
                     ...   OO'
RANK DATA            10 A  OCCURS 3
COLUMN RANK          2 A, OCCURS 5
ELEMENT COLUMN       2 A
```

This illustration defines a two-dimensional array, ELEMENT, with three rows and five columns, each occurrence of which is an alphabetic field of two characters. The first dimension, RANK, is defined as having three occurrences. The second dimension, COLUMN, is defined as having five occurrences. The length of the first dimension, RANK, must be the length of the second dimension, COLUMN, times the number of occurrences of the second dimension, COLUMN.

The exhibit below illustrates the relationship between the array element and the corresponding array element value:

<table>
<thead>
<tr>
<th>ELEMENT</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>Is</td>
</tr>
<tr>
<td>ELEMENT(1,1)</td>
<td>AA</td>
</tr>
<tr>
<td>ELEMENT(1,2)</td>
<td>BB</td>
</tr>
<tr>
<td>ELEMENT(1,3)</td>
<td>CC</td>
</tr>
<tr>
<td>ELEMENT(1,4)</td>
<td>DD</td>
</tr>
<tr>
<td>ELEMENT(1,5)</td>
<td>EE</td>
</tr>
<tr>
<td>ELEMENT(2,1)</td>
<td>FF</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ELEMENT(3,5)</td>
<td>00</td>
</tr>
</tbody>
</table>
Defining a Three-Dimensional Array

A three-dimensional array is a simple extension of a two-dimensional array. To define a three-dimensional array, you define the length and number of occurrences of each dimension (as you did for a two-dimensional array). The only difference is that you add the definition of a third dimension (MONTH-LET). This third dimension enables you to easily select individual positions within a cell in the array. The exhibit below illustrates the definition and use of a three-dimensional array:

Statements:

```almond
DEFINE QUARTER-ROW W 2 N
DEFINE MONTH-COL W 2 N
DEFINE MONTHS W 120 A VALUE +
  'JANUARY +
  FEBRUARY +
  MARCH +
  APRIL +
  MAY +
  JUNE +
  JULY +
  AUGUST +
  SEPTEMBER +
  OCTOBER +
  NOVEMBER +
  DECEMBER +
DEFINE MONTH-ROW MONTHS 30 A, +
  OCCURS 4
DEFINE MONTH-COLS MONTH-ROW 10 A, +
  OCCURS 3
DEFINE MONTH-LET MONTH-COLS 1 A, +
  OCCURS 10
DEFINE MONTH-CELL MONTH-LET 1 A
JOB INPUT NULL NAME MYPROG
*  THIS PROGRAM DISPLAYS THE 3RD
*  LETTER OF THE MONTH IN THE 4TH
*  ROW, 2ND COLUMN (THE V IN NOVEMBER)
  DISPLAY MONTH-CELL (4, 2, 3)
STOP
```

Produce:

V
Using Subscripts

You can use subscripts with a field name in the following manner:

[file-name:] [record-name:] field-name ( subscript ... )

The following restrictions apply to the use of subscripts:

1. You can specify no more than 30 subscripts.
2. A subscript must be a field name or a literal. An arithmetic expression cannot be coded for a subscript.
3. Subscript value must be a positive integer, no greater than the value specified for the OCCURS parameter of the DEFINE statement for field-name.
4. You cannot subscript a field name used as a subscript.
5. An indexed field cannot be used as a subscript.

Segmented Data

One of the most common data structures is segmented data. Each record contains a fixed portion of data and multiple occurrences of data segments. The actual number of occurrences is not known until execution time. In COBOL, these structures are known as variable-length table definitions and are defined with an “occurs depending on” clause.

The exhibit below illustrates the field definitions necessary to describe a personnel record with a fixed area and variable occurrences of dependent and salary history segments.

FILE MASTER
*  
* FIXED PORTION
  *
  EMP-ID 1 5 N
  EMPNAME 6 20 A
  NO-OF-DEPENDS 26 2 N
  NO-OF-JOBS 28 2 N
  *
  DEPENDENT SEGMENTS
  *
  DEPEND-INFO 30 26 A OCCURS 20
  DEPEND-NAME 30 20 A INDEX DEPINDEX
  DEPEND-BIRTH 50 6 N INDEX DEPINDEX
  *
  SALARY HISTORY SEGMENTS
  *
  SALARY-HISTORY 30 16 A OCCURS 10
  SALARY-AMOUNT 30 8 N 2 INDEX SLINDEX
  SALARY-GRADE 38 2 N INDEX SLINDEX
  SALARY-EFF-DATE 40 6 N INDEX SLINDEX
Because the starting location for each variable occurring segment is not known, the first position after the fixed portion is used. Later, to access the data, the length of the preceding segment(s) is added to the index to determine the starting location of the next variable segment. The OCCURS parameter specifies the maximum number of occurrences for each variable portion.

The exhibit below illustrates the index manipulation statements necessary to access the data contained in the file:

```plaintext
FILE MASTER
  * FIXED PORTION
  *
EMP-ID            1   5   N
EMPNAME           6   20  A
NO-OF-DEPENDS     26  2   N
NO-OF-JOBS        28  2   N
  *
  * DEPENDENT SEGMENTS
  *
DEPEND-INFO       30  26  A        OCCURS 20
DEPEND-NAME     30  20  A        INDEX DEPINDEX
DEPEND-BIRTH    50  6   N        INDEX DEPINDEX
  *
  * SALARY HISTORY SEGMENTS
  *
SALARY-HISTORY    30  16  A        OCCURS 10
SALARY-AMOUNT   30  8   N   2    INDEX SALINDEX
SALARY-GRADE    38  2   N        INDEX SALINDEX
SALARY-EFF-DATE 40  6   N        INDEX SALINDEX
WORK-CTR          W   2   N
  *
JOB INPUT MASTER NAME PERSONNEL-REPORTS
  MOVE ZEROS TO DEPINDEX, WORK-CTR . * INITIALIZE DEPENDENT INDEX, CTR
  DO WHILE WORK-CTR < NO-OF-DEPENDS . * PROCESS ALL DEPENDENT PORTIONS
    PRINT DEPEND-REPORT
    WORK-CTR = WORK-CTR + 1
    DEPINDEX = DEPINDEX + 26
  END-DO
  *
  MOVE ZERO TO WORK-CTR . * REINITIALIZE CTR
  SALINDEX = (NO-OF-DEPENDS * 26) . * START OF SALARY HISTORY IS THE
  *
  DO WHILE WORK-CTR < NO-OF-JOBS . * PROCESS ALL SALARY PORTIONS
    PRINT SALARY-REPORT
    WORK-CTR = WORK-CTR + 1
    SALINDEX = SALINDEX + 16
  END-DO
  *
REPORT DEPEND-REPORT LINESIZE 72 SPACE 1
  TITLE 'DEPENDENT REPORT'
  LINE EMP-ID EMPNAME DEPEND-NAME DEPEND-BIRTHDATE
  *
REPORT SALARY-REPORT LINESIZE 72 SPACE 1
  TITLE 'SALARY REPORT'
  LINE EMP-ID EMPNAME SALARY-AMOUNT SALARY-GRADE SALARY-EFF-DATE
```
Data Strings

Evaluating strings of data is another common index process. An example of this process is:

FILE NAMES CARD

| DATA-NAME | 1   | 20 | A  |
| SCAN-NAME | DATA-NAME | 1   | A  | INDEX SUB1 |
| REVERSED-NAME | W   | 20 | A  |
| SCAN-REVERSED | REVERSED-NAME | 1   | A  | INDEX SUB2 |
| COUNTER | W   | 2   | P  |
| SAVE-COUNT | W   | 2   | P  |

* INITIALIZE REVERSED NAME, SUB1, SUB2, AND COUNTER FIELDS

MOVE SPACES TO REVERSED-NAME
MOVE ZEROS TO SUB1, SUB2, COUNTER

* FIND LENGTH OF LAST NAME

DO WHILE SCAN-NAME NQ ','
    COUNTER = COUNTER + 1
    SUB1 = SUB1 + 1
END-DO

SAVE-COUNT = COUNTER  . *SAVE LENGTH OF LAST NAME
COUNTER = 0             . *RESET COUNTER
SUB1 = SUB1 + 1        . *BUMP SUB1 PAST THE COMMA

* FIND FIRST NAME AND MOVE TO REVERSED NAME

DO WHILE SCAN-NAME NQ ' ' +
    AND COUNTER LE 20
    SCAN-REVERSED = SCAN-NAME
    COUNTER = COUNTER + 1
    SUB2 = SUB2 + 1
    SUB1 = SUB1 + 1
END-DO
COUNTER = 0             . *RESET COUNTER
SUB1 = 0                . *RESET TO BEGINNING OF LAST NAME
SUB2 = SUB2 + 1         . *BUMP SO SPACE IS BETWEEN FIRST AND LAST NAMES

* MOVE LAST NAME TO REVERSED NAME FIELD

DO WHILE COUNTER LQ SAVE-COUNT - 1
    SCAN-REVERSED = SCAN-NAME
    COUNTER = COUNTER + 1
    SUB1 = SUB1 + 1
    SUB2 = SUB2 + 1
END-DO
PRINT NAMES-REPORT
REPORT NAMES-REPORT LINESIZE 78
LINE REVERSED-NAME DATA-NAME
END
The exhibit below illustrates a technique for taking names from the input record, reversing them, and then printing them. The results of this program are:

<table>
<thead>
<tr>
<th>REVERSED-NAME</th>
<th>DATA-NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIM WIMN</td>
<td>WIMN, JIM</td>
</tr>
<tr>
<td>BOB SNIGGOC</td>
<td>SNIGGOC, BOB</td>
</tr>
<tr>
<td>CARL PETERSON</td>
<td>PETERSON, CARL</td>
</tr>
<tr>
<td>PAULVIN DEMBOMAN</td>
<td>DEMBOMAN, PAULVIN</td>
</tr>
</tbody>
</table>
Programming Languages

The facilities of CA-Easytrieve/Plus provide all of the functions necessary to perform standard input/output, data examination, and data manipulation. You can also invoke subprograms written in other programming languages through the EXIT parameter of the FILE statement for input/output related events (see Chapter 5, “Data Definition”) and through the CALL statement for all other requirements.

- Input/output exits can be used to support file types that CA-Easytrieve/Plus does not process directly.
- CALL exits typically supply unsupported functions or interface with existing programs in other languages.

This chapter describes the techniques used to invoke subprograms of other languages. The following points are discussed:

- Program loading
- Storage management
- Linkage (register usage) conventions
- Parameter list
- LE-Enabled CA-Easytrieve/Plus
- Error condition handling.

Subprograms written in Assembler, COBOL, FORTRAN, and PL/I are supported by CA-Easytrieve/Plus with certain limitations. These limitations are discussed in this chapter. The Programmer Guide for the programming language you use describes the idiosyncrasies of that language when being used as a subprogram.
Program Loading

Programs written in other languages are loaded into storage as a part of activity (JOB or SORT) initiation. No matter how many times you reference a program in the CA-Easytrieve/Plus program, only one copy is loaded. If the same program is to perform multiple functions, parameters must be passed to the program to identify the desired function.

In OS/390, all programs are loaded by invoking the LOAD function of the operating system. The LOAD function dynamically places the program into storage and returns the program's entry point to CA-Easytrieve/Plus.

In VSE, programs are loaded in three different ways:

- VSAM file exits are loaded by the CDLOAD function of the operating system. These relocatable or self-relocating programs are placed in the partition's GETVIS area.

- Non-relocatable programs are loaded by the LOAD function of the operating system. These programs are loaded at the location where they were link edited. To protect against overlaying of CA-Easytrieve/Plus controlled storage, you must load non-relocatable programs at a location that is at least 4K-bytes higher than the STORMAX option value (see Appendix C, “Options Table”).

- All other programs that must be relocatable or self-relocating are loaded by the LOAD function into space controlled by the EXITSTR option value (see Appendix C, “Options Table”).

![VSE Program Partition Diagram]

controlled by "SIZE-nn" on the "// EXEC" JCL statement

CA-Easytrieve Plus area

EXITSTR (via LOAD) relocatable area

STORMAX (via LOAD) non-relocatable area

Partition size
Storage Management

In VSE, the author of programs in other languages is responsible for managing required storage. If additional storage is needed, (for example, to LOAD another program), you cannot use DOS COMREG facilities. All storage must be:

- Within the originally loaded program
- Obtained using GETVIS
- Uniquely controlled within the STORMAX area.

Linkage (Register Usage) Conventions

When CA-Easytrieve/Plus invokes a subprogram written in another programming language, it adheres to standard IBM register management conventions. The called subprogram must honor these conventions.

Linkage Register Usage

- REGISTER 1 — Address of the parameter list
- REGISTER 13 — Address of an 18-fullword register Save Area
- REGISTER 14 — Address of where to return to within CA-Easytrieve/Plus
- REGISTER 15 — Address of the entry point in the subprogram.

The subprogram must save the CA-Easytrieve/Plus registers in the save area addressed by REGISTER 13 and must restore them prior to returning using REGISTER 14. This 18-fullword register Save Area provided by CA-Easytrieve/Plus must be maintained as illustrated below:

Register Save Area Usage

- WORD 1 — Reserved
- WORD 2 — Set by CA-Easytrieve/Plus to the address of the Save Area for the internal routine prior to the one issuing the subprogram call.
- WORD 3 — Set by the subprogram to the address of the Save Area within the subprogram.
- WORD 4 through WORD 18 — Set by the subprogram to values contained in CA-Easytrieve/Plus REGISTERS 14 through 12 upon entry to the subprogram.
As previously stated, CA-Easytrieve/Plus can invoke subprograms written in any other programming language, as long as the subprogram follows the conventions described here. Typical linkage is to assembler language, COBOL, or PL/I subprograms.

**Assembler Subprogram Linkage**

Assembler language subprograms present no linkage problems. The exhibit below depicts the instructions necessary to successfully control assembler language subprogram linkage.

```
ASMPGM CSECT
  STM  14,12,12(13) save registers 14 through 12
  LR   11,15       set base register
  USING ASMPGM,11  assign base register
  LA   14.0(0.13)  address of CA-Easytrieve/Plus' save area
  LA   13,MYSAVE   address of subprogram's save area
  ST   13.8(0,14)  chain forward
  ST   14,MYSAVE+4 chain backward
  LR   10,1        save parameter list address
  ...
  ...
  ...
RETURN  L   13.4(0,13) address of CA-Easytrieve/Plus' save area
  LM   14,12,12(13) restore CA-Easytrieve/Plus' registers
  MVI  8(13),X'FF' indicate unused save area
  SR   15,15       set zero return code
  BR   14          return to CA-Easytrieve/Plus
  ...
MYSAVE DC   18A(0) 18 fullword save area
  ...
  ...
```

OS/390 Assembler subprograms, which are called, can have an AMODE of 24, 31, or ANY and an RMODE of 24 or ANY. EXIT programs must have an AMODE of 24 and an RMODE of 24.
COBOL Subprogram Linkage

COBOL subprogram linkage is dependent upon the operating system (OS/390 or VSE) and the COBOL parameters that were in effect when the COBOL subprogram was compiled. Refer to the COBOL Programmer's Guide for specific details on these parameters and linkage conventions. The exhibit below depicts typical COBOL instructions necessary to control subprogram linkage:

```
... LINKAGE SECTION.
  01  PARAMETER-1.
  ... 
  01  PARAMETER-2.
  ... 
  01  PARAMETER-N.
  ...
  PROCEDURE DIVISION USING PARAMETER-1,
      PARAMETER-2,
      ...
      PARAMETER-N.
  ...
  GOBACK
  ...
```

In VSE, for OS/VS COBOL subroutines to behave correctly as called programs, the linkage control module ILBDMNS0 must be assembled and linked instream with the COBOL phase as follows:

```
// OPTION CATAL
  PHASE CBLNAME.*
// EXEC ASSEMBLY
  ILBDMNS0 CSECT
     DC    X'FF000000000000000000000000000000'
  * Older versions of COBOL used an eight-byte ILBDMNS0
  * Check existing link maps to determine the length.
  END
/*
  // EXEC FCOBOL
    ...COBOL SOURCE
/*
  // EXEC LNKEDT
```

**Note 1:** Different releases of the COBOL compiler create ILBDMNS0 modules of differing lengths. Please check one of your ILBDMNS0 link maps for the correct length.

**Note 2:** Do not link ILBDMNS0 for COBOL II and COBOL for VSE (LE). Instead, use the ENVIRONMENT COBOL parameter, as described in the ENVIRON Option and ENVIRONMENT Parameter section in this chapter.
There are two ways to establish a proper execution environment for COBOL subprograms in OS/390:

1. For OS/VS COBOL, you can compile your COBOL subprogram with the NOENDJOB and NORESIDENT options. This establishes the subprogram properly as a called program. The NOENDJOB parameter is not available in COBOL II.

2. For COBOL II or LE COBOL, use the CA-Easytrieve/Plus ENVIRON system option or the ENVIRONMENT parameter of the JOB and PARM statements. Using any of these tells CA-Easytrieve/Plus to establish the environment for a called COBOL subprogram.

**ENVIRON Option and ENVIRONMENT Parameter**

A COBOL environment is set for JOB activities containing CALL statements in one of three ways:

1. The ENVIRONMENT COBOL parameter of the JOB statement is coded. This establishes the proper LE and/or COBOL environment only for subprograms called in that JOB activity. (See JOB Statement in Chapter 6, “Processing Activities.”)

2. The ENVIRONMENT COBOL subparameter of the PARM statement is coded. This establishes the proper LE and/or COBOL environment for all subprograms called in all JOB activities of the CA-Easytrieve/Plus program. (See PARM Statement in Chapter 4, “Environment Definition.”)

3. The ENVIRON=COBOL option of the system options table is set when CA-Easytrieve/Plus is installed. This establishes the proper LE and/or COBOL environment for all subprograms called in all JOB activities for all CA-Easytrieve/Plus programs. (See Appendix C, “Options Table.”)

**COBOL ENVIRONMENT Operation**

When the ENVIRON=COBOL option or ENVIRONMENT COBOL parameter is in effect, the correct environment is established for JOB activities containing CALL statements, and JOB activities referencing files that use FILE EXIT, according to the hierarchy described earlier.

CA-Easytrieve/Plus loads and calls a “stub” program to establish the proper environment. This stub program is supplied in object form and is link edited with CA-Easytrieve/Plus. For COBOL II, a COBOL II stub program is called. The stub acts as the main COBOL program in the run unit. It initializes the LE and/or COBOL environment at the start of the JOB activity and terminates the environment at the end of the JOB activity. This enables subprograms to act properly as called subroutines without using the ENDJOB compiler option.
**COBOL Environment Rules**

The rules for using ENVIRONMENT COBOL are as follows:

- The parameter functions for CALLED subprograms, FILE EXITs; it functions in OS/390 and VSE.
- All COBOL programs in the run unit must be compiled with the RESIDENT and REENTRANT compiler options. This ensures that all COBOL modules access the same copy of the global data area and optimum performance is obtained.
- FILE EXITs and subprograms can have an AMODE of 24, 31, or ANY and an RMODE of 24 or ANY.
- When the COBOL subprogram issues a GOBACK statement, control returns to the CA-Easytrieve/Plus statement following the CALL statement.
- When the COBOL subprogram executes a STOP RUN statement, the statement causes the current CA-Easytrieve/Plus activity to terminate. When an activity terminates due to a STOP RUN, any spooled reports currently being printed are terminated. Any unprinted spooled reports are purged, therefore, programs that use FORTRAN service routines cannot be called.
- Support for Language Environment (LE) takes place through the ENVIRONMENT COBOL parameter. See the LE-Enabled CA-Easytrieve/Plus section in this chapter for guidelines concerning subprograms in LE.

**VS FORTRAN Subprogram Linkage**

Linkage with VS FORTRAN is supported in a none I/O mode only. VS FORTRAN requires a static link of module #VFEIN with the calling program that does the I/O. CA-Easytrieve/Plus does not support static linkage, therefore, programs that use FORTRAN service routines cannot be called.

**PL/I Subprogram Linkage**

Linkage with PL/I is unique due to its non-standard conventions. This limits its use to 24-bit processing only. It also requires use of the PROC OPTIONS(COBOL) parameter. Refer to the PL/I Optimizing Compiler Programmer’s Guide for details on the linkage of PL/I subprograms with other programming languages.
Parameter Lists

The parameter list for both input/output and CALL exits (pointed to by REGISTER 1) passes information to the subprogram. Each entry in this contiguous group of fullwords identifies one parameter. The end of the list is indicated by the high-order bit of the last entry being set to a one.

Parameter List Format

```
<table>
<thead>
<tr>
<th>Register 1</th>
<th>Parameter List</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>00 address 1</td>
</tr>
<tr>
<td></td>
<td>00 address 2</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>80 address n</td>
</tr>
</tbody>
</table>
```

The parameter lists passed to subprograms for EXIT (FILE) and CALL are quite similar. In fact, the list for CALL is identical to that associated with the USING subparameter of EXIT. The only difference is that EXIT always passes at least two parameters.

Exit Parameter List

You can use the EXIT parameter of the FILE statement to invoke subprograms written in other programming languages for input/output related events. Code the name of these subprograms on the EXIT parameter of the FILE statement in the library of your program.

```
<table>
<thead>
<tr>
<th>Parameter List</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 address 1</td>
</tr>
<tr>
<td>00 address 2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>80 address n</td>
</tr>
</tbody>
</table>
```

- address of work area
- address of control code
- USING parameters

Optional
For input/output exits, work area address and the control code address are required parameters. The control code is a fullword used to indicate the function to be performed by the exit. For instance:

<table>
<thead>
<tr>
<th>Control Code</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>input request</td>
</tr>
<tr>
<td>00000004</td>
<td>output request</td>
</tr>
<tr>
<td>00000008</td>
<td>file close request, or end-of-input (set by input exit subprogram)</td>
</tr>
</tbody>
</table>

For MODIFY exits (subparameter of the FILE statement), the required two parameters are record area address and work area address because the exit receives all records after input and before output.

Parameter List

<table>
<thead>
<tr>
<th>00</th>
<th>address 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>address 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>80</td>
<td>address n</td>
</tr>
</tbody>
</table>

- address of record
- address of work area
- optional
- USING parameters
Parameters coded on the optional USING subparameter of EXIT are appended to the standard two parameters. The exhibit below shows input/output and MODIFY subprogram parameter list examples:

\((\text{INPUT/OUTPUT})\)  
\[
\ldots 
\text{FILE USERFIL EXIT \ (ASMPGM USING (RECORD-LENGTH))} 
\ldots 
\]

\begin{table}
<table>
<thead>
<tr>
<th>Parameter List</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>00</td>
</tr>
</tbody>
</table>
\end{table}

\((\text{MODIFY})\)  
\[
\ldots 
\text{FILE USERFIL EXIT \ (ASMPGM MODIFY) WORKAREA(500)} 
\ldots 
\]

\begin{table}
<table>
<thead>
<tr>
<th>Parameter List</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
</tr>
<tr>
<td>80</td>
</tr>
</tbody>
</table>
\end{table}
CALL Statement

The CALL statement provides a means to dynamically invoke subprograms written in other programming languages. The syntax of the CALL statement is:

Syntax

CALL program-name [NR] USING({field-name-1} ...)

[RETURNS field-name-2]

Operation

program-name

Program-name is the name of the subprogram that you want invoked. It is loaded into storage as part of the JOB or SORT activity initiation.

[NR]

NR is a VSE-only option which specifies the subprogram as non-relocatable.

 USING({field-name-1} ...)

USING specifies the parameter list passed to the subprogram. Field-name-1 must identify a system-defined field, a working storage field, or a field defined in an accessible file. If field-name contains DBCS data, then CA-Easytrieve/Plus passes it unaltered to the subprogram. Literal can be any literal you want passed to the program. The DBCS code system to be used for this literal is the DBCS code system defined as the Processing code. The data format for the literal does not change. You can pass a maximum of 64 fields to the called program.

[RETURNS field-name-2]

RETURNS identifies a numeric field to contain the return code passed back by a called subprogram. If calling a COBOL subprogram, the return code is the value in the COBOL RETURN-CODE field. If calling an Assembler subprogram, the return code is the value contained in REGISTER 15. Field-name-2 is a numeric CA-Easytrieve/Plus field which contains the RETURNed value. The field can be a user-defined field or the system-defined field RETURN-CODE, passing the return code to the operating system.
CALL Parameter Lists

The exhibit below illustrates a CALL statement and its associated parameter list. When the CALL statement does not contain a USING parameter, REGISTER 1 is set to 0 to indicate that no parameter list is being passed.

```
CALL ASMPGM USING ('USERFIL', ...)
```

```
Parameter List

<table>
<thead>
<tr>
<th>00</th>
<th>address of literal USERFIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>address of second parameter</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>80</td>
<td>address of last parameter</td>
</tr>
</tbody>
</table>
```

```
CALL ASMPGM
...
Register 1 is zero indicating no parameter list
```

LE-Enabled CA-Easytrieve/Plus

CA-Easytrieve/Plus supports LE, COBOL II, and OS/VS/COBOL via the ENVIRONMENT COBOL parameter.

When PARM ENVIRONMENT(COBOL) is specified in a CA-Easytrieve/Plus program (or in the CA-Easytrieve/Plus Options Table), the CA-Easytrieve/Plus initialization establishes either the LE/COBOL or the COBOL II environment. You control which environment is established by your placement of the LE runtime library (SCEERUN in OS/390 or SCEEBASE in VSE). In order to run in LE mode, the LE runtime loadlib must be in the linklist, joblib, steplib, or LIBDEF preceding any other COBOL runtime libraries. If you do not do this, CA-Easytrieve/Plus will not do the LE setup. If CA-Easytrieve/Plus does not find the LE runtime lib members, the COBOL II environment is established.

**Note:** Leave LE runtime library out in order to run older COBOLs.

When running with LE runtime libraries and PARM ENVIRONMENT(COBOL), the CA-Easytrieve/Plus program becomes the "main" LE program in the job execution.
CA-Easytrieve/Plus Calling COBOL

To call COBOL programs, specify `PARM ENVIRONMENT (COBOL)` in the CA-Easytrieve/Plus program or in the Options Table. We highly recommend that you recompile and link the COBOL programs using the LE compiler and link libraries.

Results are unpredictable if you call old compiled and linked COBOL programs while running with LE libraries.

CA-Easytrieve/Plus Calling Assembler

The instructions in this section are needed only if the CA-Easytrieve/Plus program is calling LE-enabled Assembler programs. Your existing Assembler programs do not need to be LE-enabled because of CA-Easytrieve/Plus LE support.

To call Assembler, specify `PARM ENVIRONMENT (COBOL)` in the CA-Easytrieve/Plus program or in the Options Table.

On the CEEENTRY macro in the Assembler program, specify the following:

```
MAIN=NO
PARMREG=1
```

CA-Easytrieve/Plus Calling LE CEEExxxx Routines

To call LE CEEExxxx routines, specify `PARM ENVIRONMENT (COBOL)` in the CA-Easytrieve/Plus program or in the Options Table.

CA-Easytrieve/Plus Calling PL/I

In order to call PL/I, use the following guidelines:

- Specify `PARM ENVIRONMENT (COBOL)` in the CA-Easytrieve/Plus program (or in the Options Table).
- In the PL/I program, specify `PROC OPTIONS (FETCHABLE COBOL)`. Do not specify `MAIN` in the `OPTIONS` because the CA-Easytrieve/Plus program executes as the LE "main" program.
- Recompile and link the PL/I programs using the LE PL/I compiler and LE link libraries. The programs must be linked with `AMODE (31)` and `RMODE (ANY)`. 
Error Condition Handling

File Exit Programs

**PARM ENVIRONMENT (COBOL)** establishes an environment for FILE EXITS. (You no longer have to use the EZT CALL statement for FILE EXIT functionality.)

Error Condition Handling

Program errors that occur in subprogram exits cause the abnormal termination of CA-Easytrieve/Plus programs. Since these errors are occasionally difficult to analyze within the complex environment of CA-Easytrieve/Plus, exits should be tested first using simulation.
Chapter 15

System Facilities

Introduction

The CA-Easytrieve/Plus system facilities assist you in writing and debugging programs. These facilities include:

- The format of the compile listings
- Information printed at execution time
- Abnormal termination facilities.

This chapter discusses the format of compile listing options in detail, including:

- Page header
- Statement listing
- Parameter listings
- CLIST statement offset map
- DMAP data map
- PMAP program map
- Cross-reference report.

This chapter also discusses execution listings which help you understand your output.

Finally, Chapter 15 describes the abnormal termination facilities of CA-Easytrieve/Plus. These facilities pinpoint the problems in your program.
Compiler Directives

End

END is used to delimit CA-Easytrieve/Plus control statements from a CARD input file. The word END must fall within columns 1 and 72 (scan column default) to function properly as the END record.

The END record:
- Must be on a source record by itself.
- Is recognized as a delimiter and does not appear on the CA-Easytrieve/Plus compiler listing.
- Is not recognized in a macro.

The syntax is:

Syntax

```
END
```

Utility Programs

EZTPX01

EZTPX01 is a called subprogram used to interrogate a PARM coded on a JCL EXEC statement. It is distributed with CA-Easytrieve/Plus.

EZTPX01 requires two parameters, the system defined PARM-REGISTER, and a user defined input/output field. The input/output field must consist of a 2-byte binary field immediately followed by a character portion to contain the actual PARM information.

You are responsible for placing the maximum length you expect for the PARM information (the character portion) into the 2-byte binary field before calling EZTPX01. EZTPX01 moves your PARM data from your JCL EXEC statement into the character portion of your input/output field and updates the length in the 2-byte binary portion.
If the PARM data is larger than the maximum length you specified, it is truncated on the right and the maximum length you defined is retained. If the PARM data is shorter than the length you specified, the length is updated to reflect the actual length of the PARM data. See the *CA-Easytrieve/Plus Application Guide*, Chapter 16, “Processing JCL Parameters,” for a sample program using EZTPX01.

**EZTPX03**

EZTPX03 builds the Double Byte Character Set options module. See the *CA-Easytrieve/Plus Installation Guide*, Chapter 11, “DBCS Option Installation,” for more information.

**EZTPX04**

EZTPX04 builds the extended reporting options module. See the *CA-Easytrieve/Plus Extended Reporting Facility Manual*, Chapter 4, “Options Module,” for more information.

**EZTPX05**

EZTPX05 is a called subprogram and is distributed with CA-Easytrieve/Plus to recreate the original control cards used to generate your current CA-Easytrieve/Plus options table. EZTPX05 requires one parameter, an 80-byte alphanumeric array occurring once for each possible CA-Easytrieve/Plus option. Currently, there are approximately 80 options available. See the *CA-Easytrieve/Plus Installation Guide* for a sample program using EZTPX05.

**Compile Listing**

CA-Easytrieve/Plus compiles your source program into executable machine language. The compiler also produces several listings that inform you about the results of the compilation. Optional PARM statement parameters select the types of printed output generated by the compilation.

**Note:** A compiled program uses the options that were in effect at the time of compilation.

The compile listing is always directed to the CA-Easytrieve/Plus system output file (SYSPRINT/SYSLST). If the Extended Printer Options module (EZTPXRPT) has associated this file with an extended reporting printer, then the logical print records satisfy the requirements of that printer.
Header

The header consists of two lines. The first line contains the following:

1. The date and the time of the compile

   If an extended reporting printer is defined for the system output file (SYSPRINT/SYSLST) and that printer supports DBCS format data, then the date specified in the DBCS Option module (DATE parameter) is printed. If that date is not specified or the system output file does not support DBCS data, the date is in the format, defined in the CA-Easytrieve/Plus system option module.

   The date is in the format specified by the DATE option during installation. The time is in hh.mm.ss format.

2. Compiler identification.

3. The page number.

   As with the date, if the system output file is associated with an extended reporting printer that supports DBCS type data, then CA-Easytrieve/Plus edits the page count and converts it into the corresponding DBCS value. This number is then combined with the Kanji page character. If the printer does not support DBCS format data, then, CA-Easytrieve/Plus formats the page number using the value coded in the CA-Easytrieve/Plus system option module (the PAGEWRD keyword).

   The second line contains the installation name as defined by the option COMPNAME. The header is repeated at the top of each page of compiler-generated output.

   If the printer associated with the system output files (SYSPRINT/SYSLST) supports DBCS data and the COMPNAME option in the DBCS Option module is specified, then the DBCS installation name is output on this line. If the printer does not support DBCS data, then CA-Easytrieve/Plus uses the company name defined in the CA-Easytrieve/Plus Options module. See the COMPNAME option in the CA-Easytrieve/Plus Installation Guide.
Statement Listing

Input to the CA-Easytrieve/Plus compiler is listed one record per line. The line consists of a CA-Easytrieve/Plus generated statement number (A), followed by the input record (B). If the input is from a macro, ‘-macroname’ (C) is appended to the line. The exhibit below illustrates this.

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FILE TESTIN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>%DATADEF</td>
<td>-DATADEF</td>
</tr>
<tr>
<td>3</td>
<td>REGION 1 1 N</td>
<td>-DATADEF</td>
</tr>
<tr>
<td>4</td>
<td>BRANCH 1 1 N</td>
<td>-DATADEF</td>
</tr>
<tr>
<td>5</td>
<td>EMP# 9 5 N. CODE 16 1 N</td>
<td>-DATADEF</td>
</tr>
<tr>
<td>7</td>
<td>SEX 127 1 A</td>
<td>-DATADEF</td>
</tr>
<tr>
<td>8</td>
<td>JOB INPUT(TESTIN)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>DISPLAY NEWPAGE REGION BRANCH EMP#</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>STOP</td>
<td></td>
</tr>
</tbody>
</table>

If the compilation processes a DBCS literal (as identified by the presence of shift codes), and the CA-Easytrieve/Plus system output file is directed to a printer that supports DBCS data, then CA-Easytrieve/Plus prints the DBCS data as part of the normal compile listing.

Listing Control Statements

Listing control statements allow you to control (format) the physical layout of the statement listing:

- You can place listing control statements anywhere in the CA-Easytrieve/Plus source.
- Listing control statements must be on a record by themselves.
- Listing control statements do not appear in the printed output.

The five listing control statements of CA-Easytrieve/Plus are:

- LIST
- NEWPAGE
- SKIP
- PUSH
- POP.
LIST

LIST regulates the printing or suppression of all statements. Its syntax is:

**Syntax**

```plaintext
LIST [ON | OFF] [MACROS | NOMACROS]
```

ON specifies that all subsequent statements are to be printed. OFF suppresses the printing of all subsequent statements.

MACROS specifies that macro statements are to be printed if a LIST ON is in effect. NOMACROS suppresses the printing of macro statements.

The default is LIST ON MACROS.

To suppress all CA-Easytrieve/Plus listing information, use the following:

```plaintext
LIST OFF
PARM LIST(NOPARM NOFILE)
```

For more information on the PARM statement, refer to Chapter 4, “Environment Definition.”

NEWPAGE

NEWPAGE ejects the printer to the top of the next page before printing the next line of the source program. Its syntax is:

**Syntax**

```plaintext
NEWPAGE
```

SKIP

SKIP spaces the printer the designated integer number of lines before printing the next line. Integer must be positive. Its syntax is:

**Syntax**

```plaintext
SKIP integer
```
PUSH

PUSH saves the current status of the listing control indicators. Its syntax is:

Syntax

PUSH

POP

POP restores the previous status of the listing control indicators. Its syntax is:

Syntax

POP

PUSH and POP are especially useful in macros to control the listing of the macro expansion without affecting listing control outside the macro.

Example

The exhibit below illustrates some of the listing control statements:

Input program:

```plaintext
FILE TESTIN
SKIP 1
LIST NOMACROS
%DATADEF
SKIP 1
JOB INPUT(TESTIN)
   DISPLAY NEWPAGE NAME
   STOP
```

Produces this compile listing:

```plaintext
1 FILE TESTIN
2 %DATADEF
8 JOB INPUT(TESTIN)
9   DISPLAY NEWPAGE NAME
10   STOP
```
Diagnostic Format

When CA-Easytrieve/Plus detects syntax errors in the program source, an error message with the following format is printed:

1. Number of the statement in which the error occurred (A).
2. Seven asterisks to bring attention to the error (B).
3. Diagnostic message (C). This message is either a standard EBCDIC message or the EBCDIC message translated into Japanese and printed using the DBCS code system assigned to the CA-Easytrieve/Plus system printer. CA-Easytrieve/Plus makes the selection of the message type when the DBCS options module is generated. See the CA-Easytrieve/Plus Installation Guide for more details.

```
2 ERRFIELD W 2 A 1
(A) (B)                         (C)
2*******B055 INVALID LENGTH, TYPE OR DECIMAL PLACES - 1
```

The format of diagnostic messages is also described in Appendix A, “Diagnostics.”

Parameter (PARM) Listing

By default, CA-Easytrieve/Plus produces a complete listing of PARM statement options following the statement listing. The heading ‘OPTIONS FOR THIS RUN - ’ precedes this list. Override is through the LIST NOPARM parameter of the PARM statement.
The CLIST option produces a cross-reference between statement numbers and the relative storage locations at which the machine language code for the statements begin. The code for a job can span one or more 4K blocks of storage. A set of entries is produced for each block. A header identifying the program storage block number on line (A) always prefaces the CLIST. The next line (B) consists of the compiler name, maintenance level, date and time of the compilation, and the activity identification. The actual condensed list of offsets is a line pair: the first line (C) shows the hexadecimal offsets corresponding to the CA-Easytrieve/Plus statement numbers depicted on line two (D) of the pair. These line pairs are repeated for the length of each activity. END OF PROGRAM is printed at the end of all activities.

CLIST

(A) PROGRAM STORAGE BLOCK NUMBER  1

(B) CA-EASYTRIEVE/PLUS v.r yy-mm-dd/yy-mm-hh-mm-pgname

(C) OFFSET (HEX) 00F2  011E  0122  0126  0130

(D) STATEMENT NO.  4   5   6   7   8

END OF PROGRAM
DMAP

Code the DMAP option of the PARM statement to format a printed data map following the parameter listing. The data map is a table of all files and fields in the program together with their declared and default attributes. Imported field definitions also import appropriate information to the DMAP. The following exhibit shows the overall structure of the DMAP.

DMAP
work fields
file fields
activity 1 report 1 work fields
   summary file fields
      ...
   report n fields
      sum fields

FILE - (0008) FILENAME file-attributes
LOGICAL-RECORD - (008) EMP-JOB-LR

BASE ST DSPL LENGTH FMT DEC OCCURS ED M R KE LVL NAME
0000 46 A 01 EMPLOYEE
0028 6 AV 32,767 .. D .. 02 START-DATE-0415
0028 2 N 03 START-YEAR-0415
002A 2 N 03 START-MONTH-0415
002C 2 N 03 START-DAY-0415

FILE - (0008) FILENAME DLI (DBDNAME)
RECORD - (0009) SKILL ROOT

BASE ST DSPL LENGTH FMT DEC OCCURS ED M R KE LVL NAME
0000 4 N .K .D. 01 SKILL_ID-0455

RECORD - (0009) EXPERTISE PARENT : SKILL
BASE ST DSPL LENGTH FMT DEC OCCURS ED M R KE LVL NAME
0000 4 N .K .K.

activity n ...
...
**Field Group Header**

The first line of the DMAP is an identifying header preceding each field group. The field group header has the format:

```
field-type  [- (integer)]   name
```

**field-type**

Field-type is either WORK, FILE, or REPORT. When the field type is FILE, the attributes from your FILE statement definition also display.

**[- (integer)]**

Integer is the relative CA-Easytrieve/Plus file number or the report group relative reference number.

**name**

Name is the file-name or report-name being mapped. When a report is being mapped, a second field group header specifies the relative report reference number.

**Logical Record Display (CA-IDMS)**

This CA-IDMS-only portion of the DMAP displays logical record data. VERB occurs once for each valid verb. Its values are OBTAIN, STORE, MODIFY, and ERASE. KEYWORDS are 32-character IDMS-defined names printed three across.

**Record Display (DL/I)**

For DL/I files ROOT identifies the DL/I root record and PARENT identifies the DL/I record's parent record.

**Field Header**

Within each field group a field header identifies attributes associated with each field. The field header has the format:

```
BASE  ST  DSPL  LENGTH  FMT  DEC  OCCURS  ED  M  R  KE  LVL  NAME
```

**BASE**

is the identifier for the storage block in which the field is stored. Only working storage fields and internally generated work fields have BASE specified.

**ST**

is a one-character indicator of storage type (W, S, or I).
DSPL

is the hexadecimal representation of the relative displacement of the field from the beginning of the record or storage block.

LENGTH

indicates the decimal length of the field in bytes.

FMT

identifies the data type, or format, of the fields:

- A (ALPHA) - alphanumeric
- M (MIXED) - mixed
- K (KANJI) - double byte
- V (VARYING) - varying length field
- B (BINARY) - binary
- I (INDEX) - index for the most recent field that is not an index.
- P (PACKED) - packed decimal
- U (UNSIGN) - unsigned packed decimal
- Z (ZONED) - zoned decimal.

DEC

is the number of decimal positions for a quantitative field.

OCCURS

is the value in the OCCURS clause.

ED

indicates whether (H) HEX, (B) BWZ, and/or (K) KANJI is specified for the field.

M

identifies the mask associated with the field. Mask indicators A through Y specify either installation default or programmer specified masks.

R

designates that the R (RESET) option is specified for a W field.

KE

is the indicator for key fields. Possible entries are D (DL/I key) or C (IDMS CALC key).

LVL

is the level of the field. Levels 2 through 5 are indented and 6 through 50 are at the same level as 5.
NAME

is the name of the field.

PMA

Code the PMA option to request a formatted map of the compiled code. CA-Easytrieve/Plus prints this listing after the statement and DMAP listings. The PMA is prefaced by a header that identifies the program storage block number. The exhibit below illustrates the formatted listing of compiled code.

```
PMA
PROGRAM STORAGE BLOCK NUMBER   1
0000   BC   F0  F032
0180   20   PACK 71  3118  5001
0186   CP   71  3118  6F58
018C   BC   70  B1A6
0190   21   LA   00  0015
    ├─── relative displacement
    │     └─── statement number
    │         └─── generated code
```

The first column of the map in the above exhibit is the relative displacement (in hexadecimal) from the origin of the storage block. The statement number corresponds to the statement listing numbers and marks the beginning of generated code for that statement. The actual code is next in opcode/operand format. Interspersed throughout the PMA are titles and literals as they appear in your source program. The code to support the STATE and FLOW options for each statement is not printed.

XREF

The XREF option of the PARM statement causes the production of a cross-reference listing for all file names, field names, procedure names, report names, segment names, and statement labels. This listing follows the statement, DMAP, and PMA listings. The exhibit below illustrates the format of the cross-reference listing.

```
XREF LONG
SYMBOL            DEFINED AT     REFERENCES
COMPUTE-OVERTIME     15              27
FLOW                  3              37
FLDS                  4
NAME                  7
NET-PAY               8              23  34
OVERTIME              9              44
```
Execution Listing

File Statistics

When each activity ends, CA-Easytrieve/Plus optionally produces file statistics that provide information on the files used during the activity. The file statistics are prefaced by a header, the compiler name, the maintenance level, the date and time of compilation, and the activity identification.

Note: Database file statistics are only maintained during automatic retrieval.

<table>
<thead>
<tr>
<th>Column (A)</th>
<th>Column (B)</th>
<th>Column (C)</th>
<th>Column (D)</th>
<th>Column (E)</th>
<th>Column (F)</th>
<th>Column (G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESTIN</td>
<td>48</td>
<td>INPUT</td>
<td>SAM</td>
<td>FIX</td>
<td>BLK</td>
<td>150</td>
</tr>
<tr>
<td>EZTR001</td>
<td>3</td>
<td>OUTPUT</td>
<td>VFM</td>
<td>FIX</td>
<td>UNBLK</td>
<td>29</td>
</tr>
</tbody>
</table>

Column (A) is the file-name.

Column (B) is the RECORD-COUNT.

Column (C) is the file type.

Column (D) is the file access method.

Column (E) is the file format.

Column (F) is the logical record length.

Column (G) is the blocksize (or VSAM control interval size).

Virtual files (VFM) are automatically blocked.

Abnormal Termination

Most programming errors fall into two categories:

- Syntax errors
- Execution errors.

When CA-Easytrieve/Plus encounters a syntax error, it prints diagnostic messages which pinpoint the error and terminates after completing the compilation of the entire program.

When CA-Easytrieve/Plus encounters an execution error, it prints a diagnostic message for the error and terminates immediately.
If the error was generated by an interrupt code of 1 through 11,

CA-Easytrieve/Plus optionally produces an error analysis report through the ABEXIT parameter of the PARM statement.

**Diagnostic Messages**

**Syntax Errors**

Most of the errors made in programming are syntax errors relating to clerical mistakes or a misunderstanding of the language being used. With CA-Easytrieve/Plus, simple syntax rules and logical program structure nearly eliminate these errors. To pinpoint violations, CA-Easytrieve/Plus provides an extensive set of diagnostic messages (see Appendix A, “Diagnostics”).

**Execution Errors**

Also, you can encounter execution errors, most of which are easily remedied. The execution errors that CA-Easytrieve/Plus intercepts include:

- Insufficient storage
- File OPEN errors
- Table file out-of-sequence
- Database errors
- Program interrupts 1 through 11.

You can code DEBUG(FLDCHK) on the PARM statement to request that CA-Easytrieve/Plus validate all references to data fields at execution time. This validation detects invalid file field references such as referring to a field in a file after end-of-file. When FLDCHK is active and CA-Easytrieve/Plus detects an invalid field reference, it produces the message:

```
******A010 INVALID FILE REFERENCE - MASTER
```

By examining the statement involved, you can resolve the great majority of errors detected at execution time. The error message indicates the number of the incorrect statement if the STATE or FLOW options of the PARM statement are in effect. For program interrupts, you can analyze the problem in more depth. For interrupt codes 1 through 11, CA-Easytrieve/Plus provides the error analysis report and supporting DEBUG options.

The operating system detects execution errors and gives a cross-referenced diagnostic for such things as input data validity, data set format, illogical access method requests, security violations, and program interrupts.
Error Analysis Report

There are eight possible sections to the error analysis report as illustrated below:

Section 1
27 *******A066 PROGRAM INTERRUPT - CODE 7 (DATA EXCP)

Section 2
INTERRUPT OCCURRED AT 0130 BLOCK 1 FROM EP CA-EASYTRIEVE/PLUS v.r yy-mm-dd/yy-
hh.mm-pgmname

Section 3
INSTRUCTION AT 09D140 IS FA75AA85AB8
FIRST OPERAND ADDRESS 0A7AA8 CONTENTS 000000000000001C
SECOND OPERAND ADDRESS 0A7AB8 CONTENTS 0000000000000BB5

Section 4
FLOWTABLE - MAXIMUM ENTRIES 100
  16 18 31 19 43 20 21 51 27

Section 5
PSW AT INTERRUPT 078D0007 EC9D146

Section 6
REGISTERS AT INTERRUPT
  R0 0000001B R1 00097784 R2 000988DC R3 000A7BA0 R4 00097000
  R5 000975B8 R9 000A7FA0 R10 8009865A R11 0009D010 R12 00000017

Section 7
FILE ID/NAME   RECORD ADDRESS  RECORD LENGTH     STATUS
  0001 SYSPRINT   09EF78              56          ACTIVE
  0002 SYsin    000000              80          CLOSED
  0003 WORK     000000              8          CLOSED
  0004 EZIPRE   09C8F8              8          CLOSED
  0005 TESTIN   09C8F8             150          ACTIVE

Section 8
WORKING STORAGE

   BLOCK ADDRESS
   0001 097000
   0002 0A7000

Section 1

Section 1 identifies the statement number where the interrupt occurred (if the
STATE or FLOW option of the PARM statement were coded) and the type of
interrupt.

Section 2

Section 2 gives the relative displacement of the failing machine instruction from
the entry point of the indicated activity.

Section 3

Section 3 gives the storage location of the failing machine instruction and its
hexadecimal image. It also lists the operands of the instruction and their storage
addresses.
Section 4

Section 4 is the optional FLOW table.

If the FLOW option is in effect and an abnormal termination occurs, CA-Easytrieve/Plus prints a formatted list consisting of statement numbers. The list is prefaced by the header 'FLOW TABLE - MAXIMUM ENTRIES 100', where the '100' is set at installation time or by FLOWSIZ on the PARM statement (see Chapter 4, “Environment Definition”). The list of statement numbers follows the header and is read left-to-right, top-to-bottom, corresponding to the most recently executed statements. The flow table is created in a wraparound manner.

Section 5

Section 5 shows the Program Status Word (PSW) at the time of the interrupt.

Section 6

Section 6 lists the general purpose registers at the time of the interrupt.

Section 7

Section 7 lists the files used, the address of the current record of each file, the record length of each file, and the status of the file.

Section 8

Section 8 lists the starting location for each working storage block.

For OS/390, the SNAP dump option prints the standard (formatted) portion first, followed by the save area trace and the storage areas.

Cause A Data Exception

The CA-Easytrieve/Plus enhanced debugging aids only process program interrupt codes 1 through 11. In the exhibit below, we contrive an interrupt code 7 (data exception) at statement number 27.

```
24 ******** CAUSE A DATA EXCEPTION *********
25      DEFINE  BADDATA  W  2  A  VALUE '$$'
26      DEFINE  WORSTDATA BADDATA  2  N  0
27      WORSTDATA = 1 + WORSTDATA
28 ******** CAUSE A DATA EXCEPTION *********
```

To generate the data exception, define an alpha field and give it an initial value (statement 25), then redefine the alpha field as numeric (statement 26). When the field WORSTDATA is used in a numeric computation, the operating system detects the invalid numeric value '$$' and generates a data exception. The interrupt is passed to CA-Easytrieve/Plus and it prints an error analysis report.
Analyzing the Report

When you analyze the error analysis report, first determine the statement number where the interrupt occurred. There are numerous methods for finding the statement:

Section 1

If the STATE or FLOW options were active, the error analysis report contains the statement number (27) in the message.

27 *******A006 PROGRAM INTERRUPT - CODE 7 (DATA EXCP)

If you did not specify STATE or FLOW, you can locate the failing statement in two other ways:

INTERRUPT OCCURRED AT 01BC BLOCK 1 FROM EP CA-EASYTRIEVE/PLUS v.r yymm-mm/dd/yy-hh..mm-pgmname

Section 2

If the CLIST option is in effect, locate program storage block 1 from entry point 'CA-EASYTRIEVE/PLUS v.r yymm-mm/dd/yy-hh..mm-pgmname' in the CLIST. When that is found, scan the offsets until the displacement (01BC in this case) is equal to or between two offsets.

If the displacement is equal to one of the offsets, the corresponding statement number is the failing statement. If the displacement is between two offsets, the lower offset corresponds to the failing statement, as illustrated in the exhibit below:

PROGRAM STORAGE BLOCK NUMBER 1 CA-EASYTRIEVE/PLUS v.r yymm-mm/dd/yy-hh..mm-pgmname
OFFSET (HEX)          0190 019A 01AA 01B0...
STATEMENT NO.          21 27 29...

If the PMAP option is in effect, locate program storage block 1 from entry point 'CA-EASYTRIEVE/PLUS v.r yymm-mm/dd/yy-hh..mm-pgmname' in the PMAP listing. Scan down the offsets until the displacement (01BC) is found. When that is found, scan up to the nearest CA-Easytrieve/Plus statement number (27). That is the failing statement.

0190 21 LA 00 0015
... BALR EF
01A6 27 LA 00 001B
01AA L F0 315B
01AE BALR EF
01B0 ZAP 70 65C 6F01
01B6 PACK 71 65D 6F50
01BC AP 77 65C 6D0
01C2 TM 03 65C7
...
Section 3

Section 3 of the error analysis report identifies the actual machine instruction image and the operands involved in the failure.

INSTRUCTION AT 0ED1CC IS FA7765C065D0
FIRST OPERAND ADDRESS 0F75C0 CONTENTS 000000000000001C
SECOND OPERAND ADDRESS 0F75D0 CONTENTS 0000000000000BB5

As you can see, the instruction image is for an AP (add packed) instruction and the second operand is not a valid packed number; thus the data exception. ('0BB5' is the result of packing '5B5B' ($$), before the add.)

Section 4

Section 4 of the error analysis report is the FLOW trace. It shows the immediately preceding statement numbers. The last number listed is the failing statement.

FLOW TABLE - MAXIMUM ENTRIES 100
 16 17 18 31 19 43 20 21 51 27

Section 5

Section 5 displays the operating system's Program Status Word (PSW) which you can use to derive the address of the failing instruction in the dump.

Section 6

Section 6 follows the PSW with a formatted listing of the general purpose registers at the time of interrupt. You can use the contents of the register with the PMAP listing (or the dump) to do more detailed, classical debugging.
Section 7

Section 7 locates file fields for error determination. To locate the desired field, first determine the file that contains the field by referencing the DMAP listing. When you locate the desired field in the DMAP, add the displacement (DSPL - from the DMAP) and the corresponding RECORD ADDRESS from the error analysis report. This gives the storage location for the desired field. For example, in the exhibit below, if the desired field was SOCSECNUM, you would reference the DMAP listing and find:

DMAP

WORK FIELDS

...  
FILE - (0005) TESTIN
BASE  DSPL  LGH  DEC  TYPE  OCCR  MASK  EDIT  NAME  
...     0003  S   PACKED   A   SOCSECNUM
...  

then referencing the error analysis report, you would find:

FILE ID/NAME  RECORD ADDRESS  RECORD LENGTH  STATUS  
...  
0005  TESTIN  0EF8F8  150  ACTIVE  
...  
adding: RECORD ADDRESS 0EF8F8 to: DSPL 0003

you get storage location: 0EF8FB

which is the storage address of the field SOCSECNUM.

in the SNAP dump, that location would appear as:

0EF8E0.... .... .... .... .... F4F0F102 5305228C *......*
Section 8

Section 8 locates working storage fields in a dump. To locate the desired field, reference the DMAP listing and determine the working storage block number and the displacement of the field. Now, add the displacement (DSPL) to the corresponding BLOCK ADDRESS from the error analysis report. For example, if you want to locate the field WORSTDATA in the dump, reference the DMAP listing and find:

DMAP

WORK FIELDS

<table>
<thead>
<tr>
<th>BASE</th>
<th>DSPL</th>
<th>LGH</th>
<th>DEC</th>
<th>TYPE</th>
<th>OCCR MASK</th>
<th>EDIT</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0002</td>
<td>W</td>
<td>0F50</td>
<td>2</td>
<td>ZONED</td>
<td></td>
<td></td>
<td>WORSTDATA</td>
</tr>
</tbody>
</table>

...  

then referencing the error analysis report, you would find:

WORKING STORAGE

BLOCK ADDRESS

| 0001 | 0E7000 |
| 0002 | 0F7000 |

...  

adding: BLOCK ADDRESS 0F7000 to: DSPL 0F50

you get storage location: 0F7F50

which is the storage address of the field WORSTDATA.

In the dump the field appears as:

0F7F40 .... .... .... .... 5B500000 .... .... *.......*
Chapter 16 discusses the CA-Easytrieve/Plus macro facility. This feature permits often-repeated source statements to be duplicated easily for any program. This facility enables the language to be tailored to the programming standards of your installation.

Even the most casual programmer can use CA-Easytrieve/Plus macros. The macro library is a very convenient place to store the data definition statements of frequently used files. This use of the macro facility provides standardized data-naming conventions.

Chapter 16 first discusses how macros are invoked using the macro invocation statement. Second, it discusses the two parts of a macro:

- The macro prototype statement
- The macro body.

Chapter 16 concludes with a description of macro processing and parameter substitution.

**Macro Invocation Statement**

The macro invocation statement consists of a macro name preceded by a percent (%) sign and followed by an optional subtype for use with VSE Source Statement Libraries. Its syntax is:

```
%macro-name [ .subtype ]
```

%macro-name

Macro-name is the name of a previously stored macro that you want to invoke.
Macro Invocation Statement

[.subtype]

For VSE users of Source Statement Libraries, subtype is an optional one-character subtype for an SSL member. It is delimited from the macro-name (book name) with a period (.). If subtype is not coded, the default subtype is taken from the CA-Easytrieve/Plus options table. The default subtype for CA-Easytrieve/Plus macros stored in Source Statement Libraries is Z.

**Invoking Macros**

To invoke a macro, code a macro invocation statement any place within the CA-Easytrieve/Plus source program. Macro invocation statements cause a series of statements to be retrieved from the macro library and included as source statements in the CA-Easytrieve/Plus program. The series of statements can be modified by parameters supplied on the macro invocation statement.

```
Source Input                      Macro Library

...                               macro-1
macro-1 invocation
...                               series of macro-1
macro-2 invocation
...                               statements
...                               ...
macro-2
...                               series of macro-2
                                          statements
...

Produces
CA-Easytrieve Plus Input

...                               expanded statements
macro-1 expanded statements
...                               expanded statements
```
Macro Library

Macro statements are stored and maintained in a macro library. The MACRO options table entry specifies the macro library storage access method. The types of access methods are:

- PAN — macros stored in a CA-Panvalet library and maintained through CA-Panvalet utilities.
- LIBR — macros stored in a CA-Librarian library and maintained through CA-Librarian utilities.
- PDS — (OS/390 only) macros stored in a partitioned data set.
- SSL — (VSE only) macros stored in a VSE Source Statement Library.
- USER — any user-supplied library facility that you can use for the macro library.
- VSAM — a specially formatted VSAM data set that you can use as the macro library.

Macro Library Security

CA-Easytrieve/Plus provides the following techniques to protect your macro statements.

CA-Panvalet

In addition to having the maintenance and backup capabilities provided by CA-Panvalet, CA-Easytrieve/Plus gives you the ability to secure the macro against unauthorized access. This is accomplished through a security access code, which can be applied to a CA-Panvalet member, and by the CA-Panvalet library control codes.

Security Access Code

A security access code applies to an individual CA-Panvalet library member. You must supply the security access code on an ACCESS record before CA-Easytrieve/Plus can retrieve a secured member.

```
ACCESS 'eight-byte code'
```

Library Control Codes

Library control codes apply to an entire CA-Panvalet library or libraries. A library control code is the sum of the CA-Panvalet installation code and a library’s security code. To gain access to a secured CA-Panvalet library, you must supply the correct library control code on a CONTROL record.

```
CONTROL 'literal-1'
```
The string 'literal-1' must be in quotes and can be 1-8 characters long. CA-Easytrieve/Plus does not print the CONTROL record. The CONTROL record must be on a record by itself. For more information on library control codes, see the CA-Panvalet System Management Guide.

**Note:** The CONTROL record used to pass CA-Panvalet control codes has no relation at all to the CA-Easytrieve/Plus CONTROL statement used for control breaks.

**CA-Librarian**

CA-Librarian manages and maintains the CA-Easytrieve/Plus macro statements that are stored in its libraries. CA-Easytrieve/Plus requires access to the CA-Librarian modules FAIROPN, FAIRMOD, FAIRREC, FAIRNTE, FAIRPNT, and FAIRCLS for the processing of this macro interface. The modules are loaded by CA-Easytrieve/Plus during program compilation and therefore, the CA-Librarian product library must be available.

**Maintenance**

The macros are maintained, added, updated, and deleted on the CA-Librarian library by the services of CA-Librarian. Refer to the CA-Librarian Command Reference (Batch) Manual for details on maintenance procedures.

**Security**

If your CA-Librarian data set is protected from access by the use of a “management code,” then this access is secured with CA-Easytrieve/Plus.

To gain access to a secured CA-Librarian data set, you must supply the correct “management code” on the CA-Easytrieve/Plus CONTROL record.

```
CONTROL 'literal-1'
```

The string 'literal-1' must be in quotes and conform to the CA-Librarian rules. CA-Easytrieve/Plus does not print the CONTROL record. The CONTROL record must be on a line by itself. For more information on library “management codes,” the -MCD statement, see the CA-Librarian Lock Facility Manual.

**PDS/SSL**

OS/390 and VSE provide the capability, through program products, such as RACF or ACF2, to secure an entire macro library.
USER

Security functions of a USER macro library must be provided by the USER macro interface.

VSAM

VSAM provides the capability of protecting the macro library through the use of VSAM password protection. Before CA-Easytrieve/Plus can retrieve a macro from a secured library, you must supply the library password on an ACCESS record prior to the first macro call.

ACCESS 'eight-byte password'

For both CA-Panvalet and VSAM macro storage access methods, the ACCESS record can appear anywhere in the CA-Easytrieve/Plus program prior to the retrieval of the macro, and remains in effect until the next ACCESS record is encountered. CA-Easytrieve/Plus does not print the ACCESS record. The ACCESS record must be on a record by itself. Refer to the CA-Easytrieve/Plus Installation Guide, Appendix D, “Macro Libraries,” for detailed information about creating and maintaining macro libraries.

Macro Definition

Macros are composed of three parts:

1. The macro prototype defines the parameters of the macro.
2. The macro body contains the CA-Easytrieve/Plus statements to be generated by a macro invocation statement.
3. The optional macro termination command.
Macro Definition

The name of a macro is the same as the member name in the macro storage library.

```
PROTOTYPE
STATEMENT MACRO 2 NUMBER RESULT

* *
* NAME: MACRO EXAMPLE *
* CALCULATE THE CUBE OF A NUMBER *
* *
BODY * FUNCTION: THIS MACRO CALCULATES THE CUBE OF A NUMBER.* *
* *
* ***********************************************
DEFINE CUBE_NUMBER_ S 6 N VALUE 000000
  CUBE_NUMBER_ = &NUMBER * &NUMBER * &NUMBER
RESULT = CUBE_NUMBER_
```

Prototype Statement

The prototype statement must be the first statement of a macro. Optionally, it
defines the parameters of the macro. Either positional and/or keyword
parameters can be used.

Syntax

The syntax of the prototype statement is:

```
MACRO [literal][positional-] ... [keyword-] ... 
   [parameters] [parameters] 
```

Operation

MACRO must be the first word on a prototype statement.
[literal]

Literal is an optional parameter that specifies the number of positional parameters on the prototype statement. It is required only when you use keyword parameters. You must code the value of literal as zero when you specify only keyword parameters on the prototype statement.

[positional-parameters]

You must code positional-parameters before any keyword-parameters. The positional values are substituted according to their position on the prototype statement.

[keyword-parameters]

Keyword-parameters have two parts: the keyword name and the default value.

Positional Parameters

Use positional parameters when a value is always required for the parameters each time the macro is invoked. Frequently-used parameters are often positional, since you need to code only the value of the parameter.

Keyword Parameters

Use keyword parameters:

■ To help keep track of a large number of parameters
■ For optionally used parameters
■ To specify a default value for parameters.

Prototype Examples

The following series of examples depict the coding of macro prototype statements:

Macro with No Substitution Parameters

MACRO
...

Macro with Only Positional Parameters

MACRO POS1 POS2
...

...
The number of positional parameters is not indicated. You could have coded the optional parameter as a '2'.

**Macro with Only Keyword Parameters**

```
MACRO 0 KEY1 VALUE1 KEY2 VALUE2
... ... ...
```

Code the number of positional parameters as zero. This is a required parameter when you use keyword parameters.

**Macro with Positional and Keyword Parameters**

```
MACRO 1 POS1 KEY1 VALUE1
... ... ...
```

Macros with both positional and keyword parameters require that you supply positional parameters first.

**Macro Body**

The macro body consists of a series of model and actual CA-Easytrieve/Plus statements. The model statements contain one or more parameters that are replaced by the values of corresponding parameters on the prototype statement.

**Macro Termination Command**

The optional macro termination command is used at the end of a macro.

```
MEND
```

This statement is required when updating Source Statement Libraries with LIBR.

**Instream Macros**

Macro statements can also be included in the source input to CA-Easytrieve/Plus. This capability is particularly useful for testing new macros prior to storing them in the macro library. When an instream macro has the same name as a macro in the library, the instream macro is used.
Instream Macros

Syntax

Instream macros are placed at the beginning of the source input prior to any other statements. Each instream macro is bounded by an MSTART and an MEND statement. The format of these statements is:

```
MSTART macro-name
MACRO 2 NUMBER RESULT
****************************************************************
* 
* NAME: MACRO EXAMPLE 
* 
* CALCULATE THE CUBE OF A NUMBER 
* 
* FUNCTION: THIS MACRO CALCULATES THE CUBE OF A NUMBER. 
* 
****************************************************************
DEFINE CUBE_NUMBER_ S 6 N VALUE 0000
   CUBE_NUMBER_ = &NUMBER * &NUMBER * &NUMBER
&RESULT = CUBE_NUMBER_
MEND
```

Operation

```
macro-name
```

Macro-name is the name of the macro. It can be from one to eight characters long. The first character must be alphabetic.
Example

The exhibit below illustrates the use of instream macros.

Statements:

MSTART EXMACRO
MACRO 2 NUMBER RESULT
PUSH
SKIP 1
SKIP 1
LIST OFF
*****************************************************************
*                                                               *
*  NAME:  MACRO EXAMPLE                                         *
*         CALCULATE THE CUBE OF A NUMBER                        *
*                                                               *
*  FUNCTION:  THIS MACRO CALCULATES THE CUBE OF A NUMBER.       *
*                                                               *
*****************************************************************
POP
SKIP 1
DEFINE CUBE_NUMBER_ S 6 N VALUE 000000
SKIP 1
   CUBE_NUMBER_ = &NUMBER * &NUMBER * &NUMBER
   &RESULT = CUBE_NUMBER_
SKIP 1
MEND
*
DEFINE CUBED_RESULT W 6 N VALUE 000000 MASK (J 'ZZZZZ9')
JOB INPUT NULL NAME MACROI
%EXMACRO 3 CUBED_RESULT
DISPLAY CUBED_RESULT
STOP

Produce:

27

Macro Processing

Macro processing occurs whenever a macro invocation statement appears in a CA-Easytrieve/Plus program. Designate a macro invocation by prefixing a '% (percent sign) to the macro name. Each macro invocation retrieves a fresh copy of the macro from the library and, if necessary, replaces parameters with their corresponding values from the macro invocation statement or the prototype statement.
Parameter Substitution

The rules for substituting macro parameters are the basic rules-of-syntax and the following:

1. You must specify positional parameter values on the macro invocation statement in the same order that they appear on the prototype statement.
2. CA-Easytrieve/Plus gives the value of a null string to unsupplied positional parameter values. That is, the parameter is treated as nonexistent.
3. You can specify keyword parameter values in any order on the macro invocation statement.
4. CA-Easytrieve/Plus gives unsupplied keyword parameter values the default value from the prototype statement.
5. Within the body of a macro, the & (ampersand) is the prefix concatenated to parameter substitution words. Spell parameter substitution words exactly like their counterparts on the macro prototype except for the leading &. Delimit parameter substitution words by a ' ' (space) or a '.' (period). Use the '.' delimiter when the substituted value is to be concatenated with another word. CA-Easytrieve/Plus deletes the '.' when the parameter is replaced by its value. When you desire to have an '&' character in the macro body remain as an '&' character, you must code two consecutive ampersands (&&), even if the '&' is in a comment.

CA-Easytrieve/Plus treats a macro invocation statement that is within the body of a macro (nested) as if it were outside of the macro. That is, no special consideration is necessary. There is no limit to the nesting level.

Examples

Positional Parameter Substitution

The second parameter value (') is supplied simply to maintain correct positioning for the third parameter ('FB (150 1800)').

```plaintext
Macro invocation | Macro member = FILE
----------------|-------------------------
...             | MACRO NAME  TYPE  FORMAT
%FILE TESTIN ' ' + | FILE &NAME &TYPE &FORMAT
                 | 'FB (150 1800)'         
...             |                         

Produces

...            
FILE TESTIN FB (150 1800)  
...
Keyword Parameter Substitution

The default value of ' ' (space) for the second keyword entry (TYPE) is a good technique to use for seldom needed parameters.

```
Macro invocation | Macro member = FILE
------------------|--------------------------------
| %FILE NAME TESTIN + | MACRO & NAME FILEA +
| FORMAT 'V (1000)' + | TYPE ' ' +
| TYPE VIRTUAL | FORMAT 'FB(150 1800)'
... | FILE &NAME &TYPE &FORMAT
```

Produces

```
FILE TESTIN VIRTUAL V (1000)
...```

'&' and '.' in a Macro

The exhibit below illustrates the use of the '&' character within a macro body statement and concatenated substitution words. The extra '&' and the concatenation '.' characters are not part of the resulting statements.

```
Macro invocation | Macro member = FILE
------------------|--------------------------------
| ... | MACRO NAME PREFIX
| %FILE TESTIN NEW | FILE &NAME
... | &PREFIX.-SSN 1 9 N
| &PREFIX.-MAIL 10 75 A, +
| HEADING 'NAME & ADDRESS'

```

Produces

```
... FILE TESTIN
NEW-SSN 1 9 N
NEW-MAIL 10 75 A, +
HEADING 'NAME & ADDRESS'
...```
Introduction

CA-Easytrieve/Plus provides a comprehensive set of diagnostic messages that describes the types of errors that can occur when a program is compiled and executed.

Diagnostic messages fall into two groups which describe:

- Operational messages
- Program syntax errors.

Diagnostic Message Format

All CA-Easytrieve/Plus diagnostic messages conform to the same format:

```
*******XNNN X------------X - S------------S
```

<table>
<thead>
<tr>
<th>Message ID</th>
<th>Diagnostic Message</th>
<th>Supplement</th>
</tr>
</thead>
</table>

Message ID

The message ID, a four-character code, identifies each error message. The first character of the message ID designates the type of error. Message IDs beginning with the character A identify operational errors. Errors beginning with a B identify program syntax errors.

Diagnostic Message

The diagnostic message is a description of the detected error.
Message Supplement

The message supplement is optional, depending on the diagnostic message and context of the message. If possible, CA-Easytrieve/Plus provides a message supplement to identify the particular object which is in error.

Operational Diagnostic Messages

The following is a list of operational diagnostic messages with brief explanations for each:

A001  FILE OPEN ERROR - file-name

The operating system detected an error while attempting to open the indicated file. The file remains unopened and the job is terminated. Validate the existence and characteristics of the actual file. Make sure that the characteristics of the file match those parameters specified on the FILE statement.

A002  INVALID BLOCK SIZE - file-name

The FILE statement for the indicated file specified an incorrect value for the blocksize. Any of the following can cause this error:

- The device assigned to the file cannot support a blocksize as large as that specified.
- FULLTRK was specified for a file contained on a device other than disk.
- The required blocksize value is not specified.

A003  INSUFFICIENT CORE STORAGE AVAILABLE

The partition or region in which CA-Easytrieve/Plus is running is too small. If possible, a supplemental message is provided which defines the necessary storage type, amount, and the identity of the CA-Easytrieve/Plus routine that requested the storage. This problem can usually be corrected by increasing the partition size or region size. If the supplemental message is 'EZTVFM', either permit VFM to go to a disk or increase the VFM core storage.

A004  CATASTROPHIC ERROR IN MODULE - modname

If another Annn message preceded this error message, you should correct the problem that the preceding message describes and rerun your program. If the error persists, call Computer Associates Technical Support for assistance.
A005  I/O ERROR - file-name

The operating system has detected an input/output error for the indicated file. For OS/390, the contents of the SYNADAF buffer are appended to the message to provide additional debugging information.

A006  PROGRAM INTERRUPT - CODE x

CA-Easytrieve/Plus has intercepted a program interrupt, codes 1 through 11. Refer to Chapter 15, “System Facilities: Abnormal Termination,” for further information and the associated debugging techniques on this message.

A007  TABLE INPUT IS NOT IN SEQUENCE - file-name

The indicated external table file is not in ascending sequence by the defined argument (ARG), or the table has a duplicate key. This message applies to external tables only.

A008  TOO MANY TABLE ENTRIES - file-name

There are more table entries in the indicated external table file than specified on the related FILE statement. Recompile the program after increasing the value in the file's TABLE parameter.

A009  REPORT PROCESSING ABORTED DUE TO ERROR

Report processing has aborted due to a program interrupt or sort error. CA-Easytrieve/Plus error analysis (see Chapter 15, “System Facilities: Abnormal Termination”) or the sort program provides an error message describing the cause of termination.

A010  INVALID FILE REFERENCE - file-name

A field-name was referenced in a file which had no active record. The file might have been closed, or might be at end-of-file, or in synchronized file mode with no active record. The name of the file is specified by file-name.
A011  VSAM - type ERROR - FILE file-name - CODE nnn(xx)

This message indicates that an uncorrectable VSAM error occurred. CA-Easytrieve/Plus displays the name of the file along with the error code, which appears in both decimal (nnn) and hexadecimal (xx) formats. One of four possible types of VSAM errors are possible:

1. GENCB indicates an error occurred while building or modifying a VSAM control block.
2. OPEN indicates an error occurred while attempting to open the specified file.
3. LOGICAL indicates that an illogical I/O request was made (such as a random keyed access to an ESDS).
4. PHYSICAL indicates that a physical I/O error occurred for the specified file.

These error codes 'nnn(xx)' are described in the IBM VSE/VSAM Messages and Codes Manual.

A012  INVALID LENGTH - file-name

The record length of the specified output file is incorrect. The record length must be in the range 1 to 32767.

A013  WRONG LENGTH RECORD - file-name

The current record length for the specified input file is incorrect. Verify that the FILE statement parameters correctly describe the actual file.

A014  PREMATURE TERMINATION DUE TO PREVIOUS ERROR(S)

This message indicates that some previously identified error has caused a termination of the CA-Easytrieve/Plus execution.

A015  UNEXPECTED DBMS ERROR - file-name - FEEDBACK CODE = cc - PARM LIST IS AT - xxxxx

An unexpected condition has occurred during the path processing of a database. This is probably a result of incorrect definition of the database to CA-Easytrieve/Plus, or a failure of the database system. CA-Easytrieve/Plus provides the feedback code and the address of the parameter list for the last reference to the database system. The parameter list is contained in the CA-Easytrieve/Plus snap dump or the operating system dump, whichever is available.

The feedback code 'cc' is described in the appropriate DBMS Programmer's Guide.
A016 LOAD ERROR - program-name

An error was detected while attempting to load program-name.

A017 DBD IS NOT FOUND WITHIN PSB

The DBD specified in the FILE statement cannot be located within the PSB which was passed to CA-Easytrieve/Plus through DLI-IMS/DB.

A018 DLI-IMS/DB UPDATE IS NOT ALLOWED - function

The system was installed with the option UPDTDLI=NO and a variable function request of DLET, ISRT, or REPL was made.

A019 POINT TO ACTIVE SFP FILE IS NOT ALLOWED

A POINT statement cannot be executed for a VSAM file that is being used in a synchronized file processing job.

A020 VSAM UPDATE IS NOT ALLOWED

The system was installed with the option UPDTV$=NO and an attempt was made to extend a file by using the CREATE or UPDATE option of the FILE statement.

A021 SELECTABLE UNIT IS NOT AVAILABLE

The selectable option that you have requested is not available at this time. Please check with your system programmer to be sure the CA-Easytrieve/Plus installation was complete and that you are pointing to all necessary execution libraries.

A022 INSUFFICIENT MEMORY FOR VFM SPACE ALLOCATION

There is insufficient memory in the region/partition to allocate the amount of memory specified by theVFMSPACE parameter.

A023 VFM SPACE ALLOCATION EXCEEDED

The amount of memory specified by the VFMSPACE parameter was not enough to contain all the virtual file data. Memory is assumed when no JCL for VFM is found.
A024  ERROR OPENING THE VFM DATASET - rc

An irrecoverable error occurred opening the VFM data set.

The supplemental text “rc” is printed only when executing the program under CMS. It is the return code from the FSOPEN macro. See the CMS Command and Macro Reference Manual for an explanation of this return code.

A025  UNABLE TO RESTART THE VFM DATASET

The CA-Easytrieve/Plus Restart Control Program received a nonzero return code from the “Restart Initialization” call to VFM. A more detailed message can come from VFM.

A026  RESTART ID IS INVALID - supplemental

The Checkpoint ID specified by the user in the JCL was not accepted by IMS. The supplemental message gives further details.

**Note**: If the Checkpoint ID specified was not found by IMS on the restart log tape, IMS ABENDS with a U0102 return code.

A027  ERROR RESTORING DATA FOR RESTART - supplemental

The CA-Easytrieve/Plus Restart Control Program found unreconcilable discrepancies between the checkpointed data or control blocks and the restarting data or control blocks. This occurs when the CA-Easytrieve/Plus program is illegally modified between checkpoint and restart. The supplemental message gives details.

A028  ERROR WRITING VFM HEADER BLOCK - rc

An irrecoverable error occurred writing the file descriptor block during the initialization of the EZTVFM data set.

When the program is executed under OS/390 or VSE, an A005 message should precede this message in the listing.

The supplemental text “rc” is printed only when the program is executed under CMS. It is the return code from the FWRITE macro. See the CMS Command and Macro Reference Manual for an explanation of this return code.
A029  ERROR WRITING A VFM DATA BLOCK - rc

An irrecoverable error occurred during an attempt to write to the EZTVFM data set.

When the program is executed under OS/390 or VSE, and A005 message should precede this message in the listing.

The supplemental text “rc” is printed only when the program is executed under CMS. It is the return code from the FSWRITE macro. See the CMS Command and Macro Reference for an explanation of this return code.

A030  ERROR READING A VFM DATA BLOCK - rc

An irrecoverable error occurred during an attempt to read from the EZTVFM data set.

When the program is executed under OS/390 or VSE, an A005 message should precede this message in the listing. The supplemental text “rc” is printed only when the program is executed under CMS. It is the return code from the FSWRITE macro. See the CMS Command and Macro Reference for an explanation of this return code.

A031  CHECKPOINT-ID EXCEEDS 99,999,999

A maximum of 99,999,999 checkpoints are permitted.

A032  DATA BASE COULD NOT BE REPOSITIONED

The database could not be repositioned. Possible reasons are paths changed or database segments were added or deleted.

A033  HOSTDISK: TOO MANY FIELDS DEFINED FOR DBASEIII FORMAT -

FILE filename

The number of fields defined for the indicated file exceeds the limit set by dBaseIII.

A034  HOSTDISK: INVALID QFAM OPTION TABLE - FILE filename

Verify that a valid CA-Corporate Tie Options Table has been specified. See your CA-Corporate Tie Installation Guide for more information.
A035  HOSTDISK:  TRANSLATE TABLE NOT FOUND IN QFAM OPTION TABLE - FILE filename

Verify that the translate table was included during the assembly of the CA-Corporate Tie Options Table. See your CA-Corporate Tie Installation Guide for more details.

A036  HOSTDISK:  INVALID PC FILE FORMAT FOR FILE filename

The format specified for the indicated file is incorrect. Valid options are: LOTUS, dBASEIII, PRN, EZTPC, and BASIC.

A037  HOSTDISK:  FIELD DEFINITIONS REQUIRED FOR FILE filename

There must be at least one field defined for the specified filename. Only PRINTER files do not contain field definitions.

A038  HOSTDISK:  filename, QFAM REQUEST function: RETURN CODE: return code

CA-Corporate Tie has detected an input/output error for the specified file. The I/O request is identified by function. The error code is specified in return code. Valid functions are: open for input, open for output, read record, write record and close file. This message is always accompanied by an A039 message which identifies the CA-Corporate Tie guide to use in problem resolution.

A039  HOSTDISK:  SEE CA-CORPORATE TIE PROGRAMMER'S GUIDE, QFAM REQUEST RETURN CODES

This message always accompanies the A038 message and identifies the CA-Corporate Tie guide to use in problem resolution.

A044  INVALID OPTION TABLE PARAMETER

You specified a CA-Easytrieve/Plus option that is not valid in your operating system environment. Verify your CA-Easytrieve/Plus options.

A045  SORT TERMINATED DUE TO NON-ZERO RETURN CODE - value

SORT did not run successfully. The value parameter indicates the SORT return code.
A046  SQL - supplemental

An SQL error has occurred. The supplemental message gives detailed information on the SQL error condition as returned from the SQL interface.

Syntax Diagnostic Messages

The following group of messages describe errors detected while syntax checking the CA-Easytrieve/Plus source program. The optional supplemental messages in this group specify:

- Additional diagnostic information
- Specific object in error
- Word most likely in error.

B001  LITERALS CANNOT EXCEED 254 CHARACTERS

The literal exceeds the maximum length of 254 characters.

B002  INVALID HEXADECIMAL CHARACTER STRING

A hexadecimal character string is constructed incorrectly. (See Chapter 3, “System Overview: Hexadecimal Literals.”)

B003  EXPECTED CONTINUATION NOT RECEIVED

A statement continuation was indicated, but end-of-file on the CA-Easytrieve/Plus source input file has been detected.

B004  REPORT EXCEEDS PAGESIZE

The number of lines required to print a detail record plus any title or heading exceeds the PAGESIZE value.

B005  INSTALLATION ERROR -- CALL CA-EASYTRIEVE/PLUS SUPPORT

An installation error has occurred. Call Computer Associates Technical Support for assistance.
B006 MACRO SYSTEM -

{ LIBR } -- additional diagnostic information
{ PAN  }
{ PDS  }
{      }
{ SSL  }
{ USER }
{ VSAM }

An error has occurred within the macro system library interface. The type of library interface routine is indicated. A supplemental message, supplied by the indicated macro library interface, describes the particular problem.

B007 VALID IF/END-IF PAIRING - count

An END-IF statement is either missing or incorrectly placed. Every IF statement must be delimited by an END-IF statement. The value, count, indicates how many IF statements were not delimited by an END-IF. Look for previous error messages B173 or B185 indicating that the nesting level was in error.

B008 INVALID LOGICAL UNIT

The SYSxxx number specified is not valid or is not within the range of SYS000 through SYS240.

B009 NO MATCHING 'PROC' STATEMENT

An END-PROC statement was encountered without a valid matching PROC statement.

B010 INVALID BLOCKSIZE - file-name

An inconsistent value for the blocksize is specified on the FILE statement for the indicated file.

For fixed-length records, the blocksize must be an integral multiple of the record length.

For variable-length records, the minimum blocksize is the record length plus four (4).

B011 TABLE INPUT IS NOT IN SEQUENCE

The current INSTREAM table file is not in ascending sequence by the argument (ARG), or the table has a duplicate key.
**B012 DUPLICATE NAME -**

- `{file-name }
- `{field-name }
- `{report-name}

The indicated name is a duplicate. The name can be a file-name, field-name, or report-name.

**B013 ASSIGNMENT OPERATOR MISSING - field**

An assignment statement was being processed but an equal sign (=) was not found in the second position. Field indicates the symbol found where the equal sign was expected.

**B014 UNABLE TO RECOGNIZE STATEMENT - word**

The indicated statement is not recognizable as a CA-Easytrieve/Plus source or control statement. The optional supplemental message indicates the invalid statement. If the supplemental message is not present, the entire jobstream is unrecognizable and the input is flushed.

**B015 SELECTABLE UNIT IS NOT AUTHORIZED**

An attempt to use a function of a CA-Easytrieve/Plus selectable unit was made; however, your installation does not have that selectable unit.

**B016 INVALID OR CONFLICTING KEYWORD - word**

The indicated word is not valid for the associated statement, or it is inconsistently used.

**B017 VSAM UPDATE IS NOT ALLOWED**

The system was installed with the option UPDTVS=NO and a WRITE or PUT statement was issued for a VSAM file, or a PUT statement was issued for a file other than FILE file-name, VS(CREATE).

**B018 DLI - IMS/DB UPDATE IS NOT ALLOWED - function**

The system was installed with the option UPDTDLI=NO and a literal function code of DLET, ISRT, or REPL was specified.
B019  ADJUSTMENT NOT ALLOWED OR INVALID - word

For a DEFINE statement, an overlay redefinition for a field is in error. The redefining field must be contained within the redefined field.

For a TITLE or LINE statement, the space adjustment is invalid. A negative adjustment cannot cause line item overlay. A positive adjustment cannot extend beyond the end of the logical line.

B020  PARAMETER MUST BE NUMERIC - word

The indicated word must be numeric.

B021  PARAMETER IS TOO LARGE - word

The value of the indicated word is too large. Refer to the statement description for the valid range.

B022  IDMS UPDATE IS NOT ALLOWED

The system was installed with the option UPDTIDM=NO and a CA-IDMS command of MODIFY, STORE, ERASE, CONNECT, or DISCONNECT was specified.

B023  DECIMAL SPECIFICATION NOT ALLOWED

Decimal places are not permitted in this field.

B024  SINGLE SUBSCRIPT IDENTIFIER IS BOTH A FILE AND A FIELD NAME

The name used as a subscript has been defined as both a fieldname and a filename. Only a fieldname is valid in this context.

B025  MASK DOES NOT MATCH FIELD -

{ letter }  
{ literal }

The number of digit selectors in the indicated mask does not match the number of digits in the associated field.
**B026 REQUIRED PARAMETER IS NOT CODED - word**

Additional parameters are required. That is, parameter(s), subparameter(s), or their associated value(s) are missing. Refer to the statement syntax description for correct information.

**B027 NOT A VALID NAME -**

{ file-name   }
{ field-name  }
{ report-name }

The indicated name is not valid or is used out of context.

**B028 VALUE DOES NOT MATCH FIELD TYPE OR SIZE - word**

A VALUE clause specified an initial value which does not match the type and/or size of the associated field.

**B029 PARAMETER IS INVALID - word**

The indicated word is invalid as used in the current statement.

**B030 UNBALANCED PARENTHESES**

Parentheses must be balanced across a statement. (See Chapter 3, “System Overview: Rules of Syntax.”)

**B031 UNBALANCED APOSTROPHES**

Apostrophes must be balanced across a statement. (See Chapter 3, “System Overview: Rules of Syntax.”)

**B032 LITERAL MUST BE SPECIFIED - word**

A literal must be specified for the indicated word.

**B033 INCOMPATIBLE WORK FILE FOR CHECKPOINT**

The file specified on the FILE parameter must be virtual.
B034 CANNOT BE ENCLOSED IN PARENTHESES - word

The indicated word cannot be enclosed in parentheses.

B035 CANNOT BE ENCLOSED IN APOSTROPHEES - word

The indicated word cannot be enclosed in apostrophes.

B036 NUMBER MUST BE POSITIVE INTEGER - word

The indicated word must be a positive integer.

B037 MUST BE ALPHA LITERAL - word

The indicated word must be an alphanumeric literal.

B038 MUST BE NUMERIC LITERAL - word

The indicated word must be a numeric literal.

B039 QUALIFICATION REQUIRED

The specified name cannot be uniquely identified. Qualify the field or record.

B040 FILE CANNOT HAVE FIELDS - file-name

Fields cannot be defined for files with the PRINTER attribute. Fields cannot be defined immediately following an IDMS FILE statement.

B041 IMPROPER USE OF PARENTHESES - word

Parentheses cannot be specified in the indicated context.

B042 MORE QUALIFICATION REQUIRED

The qualified name cannot be uniquely identified. Provide additional qualification.
B043  STATEMENT CANNOT HAVE LABEL - statement

A statement label is invalid for the indicated statement.

B044  OPERAND IS MISSING

A required operand is missing for the current statement as it is coded.

B045  CHECKPOINTED USER AREA EXCEEDS MAXIMUM ALLOWED BY IMS - length

IMS enables seven (7) user areas of 32K each to be checkpointed. The combination of USING field areas and internal CA-Easytrieve/Plus data which must be checkpointed exceeds this value. The supplemented message shows the length that was attempted. You should decrease the number of USING field areas.

B046  CANNOT PERFORM A STATEMENT LABEL

You coded a PERFORM command that referenced a statement label. A PERFORM command must reference a PROC label. A PROC label must be immediately followed by another sentence consisting only of the word PROC.

B047  CANNOT GOTO A PROC LABEL

You coded a GOTO command that referenced a PROC label. A GOTO command must reference a statement label. A statement label cannot be followed by another sentence consisting only of the word PROC.

B048  FIELDNAME NOT IN FILE - field-name

The indicated field-name is not contained in the specified file.

B049  PARAMETER IGNORED - word

The indicated word is not processed by the CA-Easytrieve/Plus syntax check. A message is generated for each parameter skipped during syntax check. Syntax check termination occurs whenever the syntax of a statement becomes unrecognizable.
B050  DUPLICATE STATEMENT LABEL - label-name

The indicated label is a duplicate of a previous label. A statement label must be
unique within a procedure or the main body of a job. A procedure label must be
unique within a job activity or report subactivity.

B051  NO COMMON FIELDS FOUND

The MOVE LIKE command did not find any fields with the same names in the
two files.

B052  MORE THAN ONE COMMON FIELD - field

An error was detected while processing a MOVE LIKE statement. For a field in
the TO file, more than one field of the same name was found in the FROM file.

B053  FILE SHOULD BE IS OR VS - file-name

A READ, WRITE, or POINT statement references an un-keyed file. The file must
be either ISAM or VSAM.

B054  NOT A VALID FILE - file-name

An invalid file was specified for the current statement.

B055  INVALID LENGTH, TYPE OR DECIMAL PLACES - word

There is an inconsistency in the current field definition, or the field’s type is not
valid in the statement referencing the field. For example: an A-type field cannot
have any decimal places specified, a two-byte packed field can have a maximum
of three decimal places, or a comma was omitted between two host variables,
causing the second to be treated as a null indicator.

B056  UNRESOLVED LABEL REFERENCE - label-name

The indicated label was referenced on a statement, but a corresponding label was
never specified. This is a deferred message generated at the end of each job
activity. The program must be scanned to locate the invalid reference.
B057  INVALID LABEL REFERENCE

The indicated label has a reference which extends outside the allowed scope of reference for a label.

B058  UNRESOLVED REPORT REFERENCE

A PRINT statement specified a report-name for which a corresponding report was never specified.

B059  PREMATURE END OF FILE

An end-of-file was detected (on the CA-Easytrieve/Plus source program input file) before CA-Easytrieve/Plus could identify a valid program. Verify that the JCL statements which associate a data set with SYSIN (SYSIPT) are valid. This message is generated when a null source program is encountered.

B060  MISSING END-PROC STATEMENT

The current procedure was not terminated properly with an END-PROC statement.

B061  REPORT LINE OVERFLOWED BY - amount

The current report output line overflowed the LINESIZE by the amount specified. Solutions include:

- Reduce the width of one or more line items.
- Reduce the number of fields on the line.
- Reduce the SPACE value.
- Increase the LINESIZE value if possible.

For detailed information on the above option, see the subsection titled “Line Item Positioning” in Chapter 11, “Report Processing.”

B062  FIELD REFERENCED IN UNAVAILABLE FILE - file-name

One or more fields were referenced in the identified file, but the file is not used within the job activity. This is a deferred message which is generated at the end of each job activity. This message is always accompanied by one or more B063 messages that identify which fields were referenced in the unavailable file.
B063 FIELD REFERENCED WAS - field-name

This message always accompanies the B062 message and identifies which field(s) were referenced in the unavailable file.

B064 NUMBER OF KEYS MUST BE SAME FOR ALL FILES - file-name

The same number of corresponding keys must be specified for each file defined within a synchronized file group.

B065 JOB OR SORT STATEMENT INVALID AFTER JOB
- JOB
- SORT

A JOB or SORT statement immediately follows a JOB statement. There is no job activity defined for the preceding JOB statement.

B066 INVALIDFILENAME, FILENAME SUBSTITUTED - filename

The filename on a FILE statement was not defined properly. CA-Easytrieve/Plus has substituted a filename in order to continue processing the program. Correct the filename definition on the FILE statement.

B067 CONFLICTING OR DUPLICATE OPTION - option

The specified option either is not permitted, or has been specified more than once.

B068 PARAMETER INVALID WITH 'ADJUST' - COL

The COL parameter must be used with the REPORT NOADJUST parameter.

B069 INVALID RECORD LENGTH - word

The indicated record length is not valid.

B070 VALUE NOT WITHIN ACCEPTABLE RANGE - word

The indicated value is not within the range required for its usage. Refer to the description of the current statement for the valid range.
B071  INVALID MACRO SUBSTITUTION WORD - word

The indicated word is not a valid substitution word. Either the format of the name is incorrect, the name has not been previously defined, or the name is too long.

B072  EXCESSIVE OR MISPLACED POSITIONAL PARAMETER - word

One of the following conditions exists:
1. The number of positional parameters specified for the macro invocation exceeds the number defined in the macro definition.
2. A positional parameter was specified after a keyword parameter on the macro invocation statement.
3. A keyword parameter was misspelled, which caused it to be interpreted as a misplaced positional parameter.

B073  NUMERIC VALUE MUST BE INTEGER - word

The indicated word must be an integer.

B074  IMPROPER USE OF AMPERSAND IN MACRO

An ampersand was used incorrectly in a macro definition. One of the following conditions exists:
1. A substitution word was not preceded by either a CA-Easytrieve/Plus delimiter or the macro variable concatenation character (a period).
2. An attempt was made to define an ampersand within a literal. You must code two ampersands to define the literal ampersand; this is the same basic rule that is used for the apostrophe.

B075  UNDEFINED SUBSTITUTION WORD - word

The indicated word is used as a substitution word in the macro body, but is not defined in the macro prototype.

B076  REMAINDER OF STATEMENT IS IGNORED

A previous syntax error has occurred that makes it impossible to continue the syntax check of the current statement. The rest of the parameters on the statement are ignored.
B077  INVALID 'DO' / 'END-DO' PAIRING - count

An END-DO statement is either missing or incorrectly placed. Every DO WHILE statement must be delimited by an END-DO statement. The value, count, indicates how many DO WHILE statements were not delimited by an END-DO. Look for previous error messages B173 or B185 indicating that the nesting level was in error.

B078  INVALID 'FILE EXIT USING' PARAMETER - word

The indicated word is not valid as a USING parameter. File fields cannot be specified for USING parameters on FILE exits.

B079  'PRESIZE' OVERFLOWED, INCREASE IT

The compiler work file's record length is too small for the current job. The value used is provided in the parameter listing at the end of the compile output. Increase that value by 512 using the PARM PRESIZE parameter, and then rerun the job.

B080  PARENTAGE IS INCORRECT - record-name

The parent of this RECORD cannot be found. Either the parent name on this RECORD statement is incorrect, or the RECORD statement for the parent is missing.

B081  INCORRECT QUALIFICATION - name

The specified name is either incorrectly qualified or the qualifier used is invalid. A colon was used instead of a period in an SQL INCLUDE.

B082  NAME IS UNDEFINED - name

The specified name cannot be found.

B083  CANNOT PERFORM PROC WITHIN SAME PROC

A procedure contained a PERFORM statement which references the procedure itself. This constitutes a recursive call of the procedure and is not permitted in CA-Easytrieve/Plus.
**B084 PARAMETER INVALID FOR IDMS - word**

This parameter is invalid on an IDMS statement.

**B085 NAME TOO LONG**

The length of a name exceeded: 40 characters for field names, eight (8) characters for OS/390 file names, seven (7) characters for VSE file names, 16 characters for IDMS record names, eight (8) characters for IDMS PROGRAM, DB, NODE, SCHEMA, or SUBSCHEMA names.

**B086 INVALID LENGTH - name**

The length of this field is not valid. When this message is received just prior to a JOB statement or following a SORT statement, you used RESET and OCCURS to define an array greater than 65,520 bytes for the specified field name. RESET cannot be used with an array greater than 65,520 bytes in length.

**B087 NULL LITERAL INVALID**

A literal must have at least one character.

**B088 LITERAL NOT FOLLOWED BY DELIMITER**

The specified literal was not terminated properly.

**B089 INVALID MACRO NAME - name**

The specified name on an MSTART statement for an instream macro is invalid. The name must be no more than eight characters long, and the first character must be alphabetic.

**B090 NAME IS RESERVED - name**

The referenced name is a reserved keyword. Your use of the keyword is invalid.

**B091 INVALID TABLE FILE**

Either the table is not in ascending sequence, or there is a duplicate ARG.
B092  WORKAREA NOT VALID FOR THIS FILE

   Workarea is not supported for this file type.

B093  FILE ORGANIZATION REQUIRES DISK DEVICE

   A disk is required for the type of file.

B094  INVALID RECORD FORMAT

   Either the record format is not valid, or is inconsistent with the blocksize specified.

B095  NOT SUPPORTED IN DOS

   The requested option is not supported in the VSE (DOS) environment.

B096  LITERAL TOO LONG

   Literals are limited to 254 characters.

B097  LENGTH INVALID FOR TYPE

   The length attribute is invalid for the type of field requested.

B098  NOT A VALID TYPE

   The type attribute is not valid.

B099  DECIMAL SPECIFICATION TOO LARGE

   The number of decimal places cannot exceed the length of the field.

B100  LOCATION INVALID

   You attempted to qualify a field, used in a definition, with an invalid file. Work qualified fields cannot have an offset. File qualified fields cannot be defined in working storage.
**B101 IMPROPER FIELD OVERLAY**

The overlay field cannot be longer than or extend past the overlaid field.

**B102 TOO MANY MASKS DEFINED**

Only 192 unidentified masks can be defined in any one CA-Easytrieve/Plus run.

**B103 OVERLAY CONFLICTS WITH QUALIFICATION**

You attempted to overlay a qualified field onto a field from another file.

**B104 INDEX FIELD REQUIRED - name**

The specified name in the INDEX parameter of a DEFINE statement has been previously defined as a working storage field. Only names not previously defined, or previously defined as index names, are used in the INDEX parameter.

**B105 VALUE NOT ALLOWED FOR THIS FIELD**

The VALUE keyword is not permitted for this field.

**B106 NAME INVALID FOR TABLE FILE**

The only valid names for a TABLE file are ARG and DESC.

**B107 NUMBER MUST BE NON-NEGATIVE INTEGER**

You must use a zero or positive integer in this field.

**B108 FILE ORGANIZATIONS INCOMPATIBLE**

You cannot copy from a flat file to a database file, or from a record to a database file.

**B109 DEVICE NOT ALLOWED FOR VIRTUAL FILE**

You cannot specify a device for a virtual file.
B110  RECORD FORMAT REQUIRED FOR VIRTUAL FILE

The record format is required for a virtual file.

B111  SPECIFIED DEVICE NOT VALID FOR TABLE FILE

The specified device does not support table files.

B112  FILE ORGANIZATION NOT VALID FOR INSTREAM TABLES

A file organization keyword cannot be specified for an instream table file.

B113  DECIMAL PLACES NOT ALLOWED

An integer value is required in this field.

B114  NUMERIC FIELD REQUIRED

You tried to use an alphanumeric literal or a field name where a numeric field is required.

B115  FIELD IS READ/ONLY

You attempted to update a read only field.

B116  INVALID QUALIFIER

Either the name you attempted to define is syntactically incorrect, or you attempted to qualify a field name with another field name, or you had too many qualifiers, or you attempted to qualify a field with a reserved word other than WORK.

B117  INVALID LENGTH FOR TABLE FIELD

The length for the ARG and/or DESC fields cannot exceed 254.

B118  NAME MISSING

You referenced a null value within parentheses.
B119  INVALID SUBSCRIPT QUALIFIER
The specified subscript was not valid, or a literal was used where literals are not permitted.

B120  SUBSCRIPT INVALID ON A SUBSCRIPTED FIELD
You cannot subscript a field with a subscripted field.

B121  NAME EXPECTED AFTER A QUALIFIER
Qualification was started and not completed.

B122  NAME MUST BE A FILE OR FIELD - name
The name entered is not a field or a file.

B123  NAME NOT DEFINED IN FILE OR RECORD
The specified name cannot be found within a file or a record.

B124  NAME CANNOT BE QUALIFIED
You attempted to qualify a name that cannot be qualified. Verify that you are referencing the correct CA-Easytrieve/Plus variable.

B125  SUBSCRIPTS NOT ALLOWED
You incorrectly specified a subscript where subscripted fields are not permitted.

B126  TOO MANY SUBSCRIPTS
Only three subscripts are permitted.

B127  NOT A REPORT PROCEDURE NAME
While in a REPORT section, you attempted to PERFORM a procedure that is not a valid REPORT procedure. See Chapter 11, “Report Processing,” for a list of valid report procedures.
B128 INCORRECT NUMBER OF SUBSCRIPTS

You attempted to reference a subscripted field. The number of subscripts is dependent on how you defined the field in the library section. See Chapter 13, “Table and Array Processing,” for more details.

B129 INDEXED FIELD NOT ALLOWED

While using a variable as a subscript identifier, you referenced an indexed field. Indexed fields are not permitted to be referenced on certain statements, such as the USING parameter of a SORT statement.

B130 SORT FIELD CANNOT EXCEED 255 BYTES

The maximum length of a sort field is 255 bytes. Use more than one adjacent field when it is necessary to sort on long fields.

B131 EXIT NOT ALLOWED FOR INSTREAM TABLES

A file exit is not permitted on an instream table file.

B132 APOSTROPHE NOT PRECEDED BY SPACE

Apostrophes must be preceded by a space or a left parenthesis.

B133 COMMA NOT FOLLOWED BY SPACE

A comma must be followed by a space.

B134 ALL OTHER PARAMETERS IGNORED FOR DATABASE

Only the DL/I or IDMS keywords (and their associated sub-keywords) are valid for database file definitions.

B135 REPORT FILE LENGTH EXCEEDED

The length of a report work file record has exceeded the record length specified on the FILE statement, or if VFM is being used to spool the report, the report work file record length has exceeded 65535.
B136  SUM FILE LENGTH EXCEEDED

The length of the sumfile record has exceeded the record length specified on the FILE statement for the SUMFILE, or if VFM is being used for the sumfile, the sumfile record length has exceeded 65535.

B137  LITERAL CONTAINS DBCS DATA FROM MULTIPLE DBCS CODE SYSTEMS-word

The identified word contains the shift codes of more than one DBCS code system. You cannot mix DBCS code systems in the one literal.

B138  DBCS HEX LITERAL MUST DEFINE AN EVEN NUMBER OF BYTES

A DBCS hexadecimal literal must contain an even number of bytes. This means that it must be defined using a multiple of four hexadecimal characters.

B139  RELATIONAL OPERATOR MUST BE EQ OR NE

Series tests and range tests must use the relational operator EQ or NE.

B140  INVALID OR UNSUPPORTED DBCS CODE SYSTEM-word

The specified DBCS code system is not defined for your installation.

B141  EXTENDED REPORTING PRINTER NAME NOT DEFINED-word

The printer name defined for the EXTENDED keyword is not defined in the extended reporting options module.

B142  INVALID FILE DEFINITION FOR EXTENDED REPORTING PRINTER

An extended reporting printer file must be a sequential file. It cannot have a device type of CARD or PUNCH. It cannot be a TABLE file.

B143  UNSUPPORTED DBCS CODE SYSTEM FOR EXTENDED REPORTING PRINTER

The DBCS code system assigned to an extended reporting printer was not defined in the DBCS options module.
**B144 INVALID OR UNSUPPORTED USE OF DBCS DATA-word**

The identified field or literal containing Double Byte data is invalid or unsupported in the context in which it is being used.

**B145 INVALID OR UNSUPPORTED CONVERSION LITERAL IDENTIFIER-word**

The source conversion identifier has not been defined in the DBCS options module and this cannot be converted.

**B146 INVALID LITERAL LENGTH RETURNED BY CONVERSION LITERAL**

The length of a source conversion literal returned by the conversion routine is invalid. It must be between 1 and 254 and have an even length if the literal is all DBCS characters.

**B147 INVALID OR UNSUPPORTED DBCS CODE SYSTEM RETURNED BY SOURCE CONVERSION ROUTINE**

The DBCS code system returned by a Source Conversion Routine is not defined in the DBCS Options module.

**B148 CONVERSION ERROR - message**

A Source Conversion Routine error was indicated. The message text was returned to CA-Easytrieve/Plus by the conversion routine.

**B149 MIXED FIELDS NOT VALID FOR DBCS CODE SYSTEM**

MIXED fields can only be defined for a DBCS code system that supports either Wrapping or Header shift codes.

**B150 SORT NOT DEFINED FOR DBCS CODE SYSTEM**

The DBCS sort options are only supported for IBM and JEF DBCS code systems and only if a KANJI sort has been defined in the DBCS options module.

**B151 INVALID FONT INDEX VALUE**

A font index value must have a value of 1 through 256.
B152  DBCS CODE SYSTEM OF SORT-IN AND SORT-OUT NOT EQUAL

For KANJI sort, the DBCS code system of the sort input file must match the DBCS code system of the sort output file.

B153  ITEM DOES NOT MATCH DATA TYPE OF FONT - word

The data format of the print item does not match the data format of the font to be used to print the item.

B154  MAXIMUM OVERPRINT RECORDS EXCEEDED FOR PRINT LINE

A print line generated by the indicated statement requires more overprint records than is supported by the assigned extended reporting printer.

B155  PRINT ITEM OVERLAPS EXISTING PRINT ITEM - item

For extended reporting printers, print items (fields and literals) cannot overlap or print on top of one another.

B156  FONT DEFINED FOR HEADING LITERAL NOT FOUND - word

The font number defined for one of the headings of the item identified as word, is not defined in the extended reporting options module for the assigned extended reporting printer.

B157  FONT NOT DEFINED - font-number

The font number was not found in the extended reporting options module for the assigned extended reporting printer.

B158  INVALID USE OF DBCS DATA IN HEADING - word

One of the heading lines of the item identified as word contains DBCS data and the assigned printer does not support DBCS data.

B159  POSITION FOR PAGE COUNT OCCUPIED BY PRINT ITEM

When positioning the page count on title line 1, it was detected that a print item already occupies that position.
B160  OPTION ONLY SUPPORTED FOR EXTENDED REPORTING PRINTER - word

You can only use a font index value when the report or print line is directed to an extended reporting printer that is not a line printer.

B161  LINESIZE TOO SMALL TO POSITION TAG LITERAL

The value of LINESIZE is too small to position the TAG literal on the first summary line of the report. Increase the LINESIZE value or decrease SPACE or the size of the TAG literal.

B162  EXCESSIVE PARAMETERS SPECIFIED ON STATEMENT, REMAINDER IGNORED

The statement requires a fixed number of parameters. This value was exceeded for the indicated statement.

B163  MAXIMUM RECORD LENGTH EXCEEDED FOR EXTENDED REPORTING PRINTER

A print record generated by CA-Easytrieve/Plus for the indicated statement exceeds the maximum record size defined for the assigned extended reporting printer.

B164  MAXIMUM DATA LENGTH EXCEEDED FOR EXTENDED REPORTING PRINTER

A print record generated by CA-Easytrieve/Plus for the indicate statement contains more print data then is supported by the assigned extended reporting printer.

B165  UNABLE TO POSITION LINE COMPLEX ITEM AT EXACT PRINT POSITION - item-number

The elements on each line of a Line Complex must be positioned at exactly the same print position. Due to the positioning of other print elements on the same print record, one of the elements of a line complex could not be positioned at the same print position. You must vary the fonts and character widths of elements on the print lines to enable CA-Easytrieve/Plus to position these elements correctly.

B166  ITEM EXCEEDS MAXIMUM BYTE COUNT SUPPORTED BY A FONT - item-number

The number of bytes in the identified print item exceeds the data byte count of the assigned font. Reduce the width of the item.
B167  MAXIMUM PAGE EXCEEDED BY PAGESIZE OR LINESIZE VALUE - word

The indicated word is too large for the maximum form size defined for the associated extended reporting printer.

B168  SPACE REPLACE CHARACTER NEEDED TO USE ASSOCIATED FONT WITH PRINT ITEM - item-number

The indicated print item occurring on the specified CA-Easytrieve/Plus statement is positioned on an OVERPRINT record that is not the first OVERPRINT record and the print item MUST have a space that will print at the associated fonts width. The requirement for a space that prints at the associated fonts width is due to the item containing MIXED format data or the item is a field on a detail or summary line that must print as spaces (due to DTLCTL and SUMCTL options).

B169  RELATIONAL OPERATOR IS MISSING

The IF statement coded is missing the relational operator between the subject and the object.

B170  EXPRESSION NOT ALLOWED

Expressions are not allowed in series or range tests.

B171  SUBJECT MUST BE A FIELD OR FILE

The subject of a comparison must be either a field or a file. Literals are not permitted.

B172  SYNCHRONIZED FILE PROCESSING NOT ACTIVE

You coded a MATCH or DUP statement but did not code a JOB statement specifying synchronized file processing.

B173  NO MATCHING 'IF' STATEMENT

An END-IF statement was encountered without a valid matching IF statement, or at the same nesting level as a DO WHILE statement.
**B174  MORE THAN ONE 'ELSE' STATEMENT**

You coded more than one ELSE statement within a single IF/END-IF construct.

**B175  CONDITION IS INCOMPLETE**

The IF statement does not contain enough information to construct a valid comparison.

**B176  OBJECT OF CONDITION IS MISSING**

The object of a comparison, or a condition test, was not specified.

**B177  INVALID FIELD CONDITION**

The subject is incompatible with the condition test or the condition test is undefined.

**B178  SECOND OBJECT OF RANGE IS MISSING**

A range test was detected, but the second object is missing.

**B179  FIRST OBJECT OF RANGE IS MISSING**

A range test was detected, but the first object is missing.

**B180  ARITHMETIC OPERATOR IS MISSING**

Two operands were encountered that were not separated by an arithmetic operator.

**B181  INVALID ARITHMETIC OPERATOR**

The arithmetic operator specified is not valid.

**B182  HEXADECIMAL LITERAL REQUIRED**

When the object of a BIT test is a literal, the literal must be a hexadecimal literal.
B183 'AND' OR 'OR' IS MISSING

Compound comparisons must be joined with an 'AND' or an 'OR'.

B184 SUBJECT OF CONDITION IS MISSING

The subject of a comparison is missing.

B185 NO MATCHING 'DO' STATEMENT

An END-DO statement was encountered without a valid matching DO WHILE statement, or at the same nesting level as an IF statement.

B186 FILE NAME REQUIRED

The name specified is not a file name.

B187 RECORD NAME REQUIRED

The name specified is not a record name.

B188 FIELD NAME REQUIRED

The name specified is not a field name.

B189 PROGRAM NAME REQUIRED

The name specified is not a program name.

B190 MASK NAME REQUIRED

The name specified is not a mask name.

B191 STATEMENT OR PROCEDURE NAME REQUIRED

The name specified is not a statement or a procedure name.

B192 CURSOR NAME REQUIRED

The name specified is not a previously defined CURSOR name.
B196  FILE OR RECORD NAME REQUIRED

The name specified is not a file or record name.

B197  FILE, RECORD OR FIELD NAME REQUIRED

The name specified is not a file, record, or field name.

B198  RECORD OR FIELD NAME REQUIRED

The name specified is not a record or field name.

B199  FILE OR FIELD NAME REQUIRED

The name specified is not a file or field name.

B200  FIELD OR TABLE NAME REQUIRED

The name specified is not a field or table name.

B201  LOGICAL RECORD NOT ALLOWED

You specified a logical record name where it is not permitted.

B202  NAME MUST BE A LOGICAL RECORD

You failed to specify a logical record name. You cannot specify a field name or literal.

B203  FIND STATEMENT NOT VALID WITH LOGICAL RECORD

A logical record name cannot be specified on an IDMS FIND statement.

B204  SQL - supplemental

An SQL error has occurred. The supplemental message gives detailed information on the reason for the error as returned from the SQL interface.
B205  QUALIFYING RECORD NAME NOT VALID

The record name specified as a qualifier of a field in the WHERE parameter is not valid.

B206  OPERAND FOLLOWING PREFIX + OR - MISSING

Following a leading + (plus) or - (minus) sign, a field name, numeric literal, or an expression enclosed in parentheses must be coded. Either the end of the expression or another operator was found instead.

B207  CLOSING RIGHT PARENTHESIS MISSING

The right parenthesis that closes the nested expression was omitted.

B208  NUMERIC HEXADECIMAL LITERAL MUST HAVE SAME LENGTH AS SUBJECT - literal

When comparing a numeric field to a hexadecimal literal, the length of the literal must be the same as the length of the numeric field.

B209  OPERAND FOLLOWING + OR - MISSING

Following a + (addition) or - (subtraction) operator, a field name, numeric literal, or an expression enclosed within parentheses must be coded. Either the end of the expression or another operator was found instead.

B210  OPERAND FOLLOWING * OR / MISSING

Following an * (multiplication) or / (division) operator, a field name, numeric literal, or an expression enclosed within parentheses must be coded. Either the end of the expression or another operator was found instead.

B211  ELEMENT DOES NOT EXIST IN DICTIONARY

The specified name is not defined in the IDD.

B212  ELEMENT DOES NOT EXIST IN SUBSCHEMA

The specified name is not defined as part of the subschema that contains the logical record being accessed.
B213 ELEMENT NOT UNIQUE IN SUBSCHEMA

The subschema definition contains more than one definition of the specified name.

B214 FIELD USED IN ARITHMETIC EXPRESSION IS NOT NUMERIC

The indicated field was used as an operand in an arithmetic expression. The data type of the operand is alphanumeric. Operands used in arithmetic expressions must be numeric.

B215 OPERAND OF A LOGICAL CONNECTIVE IS INVALID

The operands of a logical connective (AND/OR/NOT) must be either a comparison, a DBA defined name, or a Boolean expression enclosed in parentheses.

B216 FIELD USED IN MATCHES/CONTAINS TEST IS NOT ALPHANUMERIC

The indicated field was used as an operand in a MATCHES/CONTAINS test. The data type of the operand is numeric. Operands used in MATCHES/CONTAINS tests must be alphanumeric.

B217 DIVISOR EXCEEDS MAXIMUM SIZE OF 8 BYTES

The second operand of a division operator (/) in the WHERE parameter cannot exceed eight (8) bytes in length.

B218 INVALID NESTED CONDITION

A nested condition in the WHERE parameter is invalid.

B219 INVALID EXPRESSION IN PARENTHESES

An expression enclosed in parentheses is invalid.

B220 LOGICAL RECORD CANNOT HAVE FIELDS - record name

A LOGICAL-RECORD cannot have fields associated with it. All fields that are defined as part of a LOGICAL-RECORD must follow an ELEMENT-RECORD statement that follows the LOGICAL-RECORD statement.
B221  ELEMENT RECORD MUST FOLLOW LOGICAL RECORD - record name

An ELEMENT-RECORD statement was coded following a FILE statement without an intervening LOGICAL-RECORD statement. ELEMENT-RECORD statements must follow a LOGICAL-RECORD statement or another ELEMENT-RECORD statement.

B222  DATA BASE FILE NOT SPECIFIED AS INPUT

The file that contains the logical record specified by the SELECT statement must be specified on the INPUT parameter of the JOB statement.

B223  ELEMENT RECORD NOT PART OF LOGICAL RECORD - record name

The name specified on the ELEMENT-RECORD statement does not match any of the record names specified for the logical record in the data dictionary.

B224  ELEMENT RECORD ALREADY DEFINED - record name

The specified element record name has already been defined for this logical record.

B225  SUBSCHEMA NAME NOT FOUND - subschema name

While attempting to locate the entry for a logical record in the data dictionary, CA-Easytrieve/Plus was unable to locate an entry for the subschema name specified on the FILE statement.

B226  RECORD NAME NOT FOUND - record name

The record name specified on a LOGICAL-RECORD statement is not defined as being part of the subschema specified on the FILE statement.

B227  RECORD NOT VALID FOR THIS FILE ORGANIZATION

A RECORD statement only follows a FILE statement that specifies an IDMS or DL/I database. A LOGICAL-RECORD or an ELEMENT-RECORD statement can only follow a FILE statement that specifies an IDMS database.
Syntax Diagnostic Messages

B228  WHERE PARAMETER SYNTAX ERROR -- REMAINDER OF WHERE PARAMETER IGNORED

A syntax error prevented the WHERE parameter from being fully processed. The unprocessed portion has been ignored. This message follows the message for the original syntax error.

B229  VARYING ALLOWED ONLY ON ALPHA, KANJI, OR MIXED FIELDS

You tried to define a varying field with a data format other than A, K, or M. A varying field cannot be defined as numeric. See Chapter 5, “Data Definition.”

B230  INVALID INDICATOR ARRAY

The indicator array specified as an SQL host variable is not valid. Indicator arrays must be two-byte binary fields occurring more than one time.

B231  RESET INVALID WITH REDEFINE

The RESET keyword must only be specified on simple definitions. RESET fields cannot be redefined and cannot be part of a redefinition.

B232  RESET ONLY VALID FOR 'W' WORKING STORAGE

The RESET keyword is only valid on W-type working storage fields.

B233  INVALID USE OF VARCHAR FIELD

A VARYING field is not valid in this context.

B254  LOCATION REQUIRES A W, S, * OR INTEGER

The LOCATION clause specified where the first field must be defined. It must be W for a W-type working storage field, S for an S-type working storage field, * for the next available position in the current FILE, or an integer specifying the position within the current FILE of the first field to be defined.

B255  SQL INCLUDE STATEMENT CANNOT FOLLOW SQL LOGIC STATEMENT

The SQL INCLUDE statement cannot follow any other SQL statement except for another SQL INCLUDE statement, and the SQL INCLUDE statement must be coded in the library section.
B256  SQL PROCESSING BYPASSED DUE TO PREVIOUS ERROR

A severe error has occurred while processing a previous SQL statement. The previous error prevents further SQL processing. Correct the error listed previously to enable SQL processing to continue.

B257  INVALID USE OF BIND PARAMETER

The BIND parameter is only valid if you have set SQLBIND option in your CA-Easytrieve/Plus options table to ANY or blanks (the default).

B258  'WHEN' STATEMENT REQUIRED AFTER 'CASE' STATEMENT

You must follow a 'CASE' statement with a 'WHEN' statement. You can insert comments and compiler directives between a 'CASE' and 'WHEN' pair, but no statements are permitted between them.

B259  FIELD LENGTH MAY NOT EXCEED 254 BYTES

You specified a field with a length greater than 254 bytes. The compiler only accepts fields with a length of 254 or fewer bytes.

B260  LITERAL LENGTH MUST BE EQUAL TO FIELD LENGTH

You specified a literal whose length was not exactly equal to the length of the subject field. The compiler requires the lengths to be exactly equal.

B261  SERIES AND RANGES MUST BE UNIQUE

Within a 'CASE' structure, the series and ranges you specify for 'WHEN' statements must be unique across all the structure's 'WHEN' statements. A series value can not be duplicated. A range can not be duplicated. A series value can not fall within a range. A range can not fall within or overlap another range.

B262  INVALID 'CASE' / 'END-CASE' PAIRING - count

An END-CASE statement is either missing or incorrectly placed. Every CASE statement must be delimited by an END-CASE statement. The value (count) indicates how many CASE statements were not delimited by an END-CASE. Look for previous error messages B173, B185 or B263 indicating that the nesting level was in error.
B263 NO MATCHING ‘CASE’ STATEMENT

An END-CASE statement was encountered without a valid matching CASE statement or an END-CASE statement at the same nesting level as a DO WHILE or IF statement.

B264 MUST BE A CONTROL FIELD-word

The fieldname specified has not been coded on the CONTROL statement.

B265 EVEN VALID ONLY ON PACKED FIELD

The EVEN subparameter is valid only on packed data fields.

B266 MUST BE A NULLABLE FIELD

The subject of the IF statement must have a null indicator associated with it.

B267 MUST NOT BE A NULLABLE FIELD

The identified field may not have a null indicator associated with it.

B268 COMMAND NOT ALLOWED IN HOST SERVER

The PRINT command is not available in this product.

B269 INVALID DATA TYPE RETURNED FROM CATALOG INTERFACE

The data type returned from the CA-Pan/SQL interface is not recognized.

B270 SQL- supplemental

An SQL warning has been issued. The supplemental message gives detailed information on the reason the warning was returned from the CA-Pan/SQL interface. This does not affect RETURN-CODE.

B290 WHEN NOT ALLOWED AFTER OTHERWISE

OTHERWISE is used when no WHEN comparisons are met. Therefore, a WHEN statement cannot follow an OTHERWISE statement.
B291  MORE THAN ONE 'OTHERWISE' STATEMENT

You coded more than one OTHERWISE statement within a single CASE/END-CASE construct.

B297  PLANNAME CANNOT BE THE SAME AS LINKNAME

The SQL planname for the STATIC SQL application plan cannot be the same as the linked program name.

B298  INVALID OPTIONS TABLE

The version of the CA-Easytrieve/Plus options table does not agree with the version of CA-Easytrieve/Plus that you are running. Refer to the CA-Easytrieve/Plus Installation Guide for information on generating the options table.

B299  SELECTABLE UNIT IS NOT AVAILABLE

The selectable option that you have requested is not available at this time. Please check with your system programmer to be sure the CA-Easytrieve/Plus installation was complete and that you are pointing to all necessary execution libraries.

B300  MIX OF IDD AND * EZTPIDD NOT ALLOWED

IDD statements and * EZTPIDD statements are not supported concurrently. Recode the EZTPIDD statements into IDD statements.

B301  IDD IDMS ERROR

The IDD statement's access into CA-IDMS has resulted in an unexpected return code. Resolve any other error messages and try again. If the messages persist, contact Computer Associates Technical Support.

B302  RECORD NAME IN SELECT NOT FOUND

The SELECT CLAUSE of an IDD SUBSCHEMA or IDD FILE statement specified a record name not found in the given dictionary.
B303 UNABLE TO RESOLVE IDD ENTITY

The SUBSCHEMA, FILE, or RECORD requested was found in the dictionary. However, either the version was incorrect, or the given program name was not authorized for the entity, not within the given schema, or not registered as being valid.

B304 IDD ENTITY WAS NOT FOUND

The SUBSCHEMA, FILE, or RECORD specified was not found in the given dictionary.

B305 NO REQUEST WAS ISSUED FOR IDD

An internal error has occurred in the IDD statement processor. Please contact Computer Associates Technical Support.

B306 USERID REQUIRED FOR SQL/DS

The userid parameter was not specified on the PARM statement. It is required for SQL/DS programs.

B307 ONLY VALID WITHIN LIBRARY DEFINITION SECTION

The statement issued is only valid in the Library Definition Section of a CA-Easytrieve/Plus program.

B308 ONLY VALID WITHIN ACTIVITY DEFINITION SECTION

The statement issued is only valid in the Activity Definition Section of a CA-Easytrieve/Plus program.

B309 KEYVALUE PARAMETER NOT CONSISTENT WITH RECORD KEY PARAMETER

Beginning with release 5.2 of CA-Easytrieve/Plus, multiple CALC keys are supported. The syntax supporting multiple CALC keys on the RECORD statement and the keyvalue clause of the RETRIEVE statement was changed. While the pre-5.2 release syntax of these two statements is still supported, mixing of the old and the new syntax is not permitted.
**B310  MAXIMUM OCCURS VALUE IS 50**

The maximum OCCURS value for the ‘TO’ parameter of a HOSTDISK file is 50.

**B311  MUST BE DEFINED IN ‘S’ WORKING STORAGE**

Only fields that specify a location parameter of ‘S’ are permitted.

**B312  FIELD MUST BE ALPHABETIC**

Only fields that specify a type of ‘A’ are permitted.

**B313  OCCURS VALUE MUST MATCH ‘TO’ FIELD’S OCCURS VALUE**

The occurs value of the HOSTDISK file's validate field must match the occurs value of file's TO field.
The following list and explanation of operational diagnostic messages is generated by DQSCGEN.

**CGEN010E  COMMAND PROGRAM GENERATION FILE IS EMPTY**

This error occurs if the CGENDATA data file was not correctly defined in the JCL.

**CGEN020E  PREMATURE END OF COMMAND PROGRAM GENERATION FILE - NO STATIC STATEMENTS FOUND**

This error is due to an invalid GENDATA file.

**CGEN024E  PREMATURE END OF COMMAND PROGRAM GENERATION FILE - MORE SEGMENTS WERE INDICATED BUT NONE WERE FOUND**

This internal error is due to invalid data in the GENDATA file.

**CGEN025E  CONTINUATION CODE ERROR - MORE SEGMENTS ARE INDICATED BUT STATEMENT LENGTH HAS BEEN EXCEEDED**

This internal error is due to an invalid record in the GENDATA file.

**CGEN026E  NO CONTINUATION OF RECORDS BUT STATEMENT LENGTH HAS NOT BEEN REACHED**

This internal error is due to an invalid record in the GENDATA file.

**CGEN030E  INVALID RECORD TYPE CODE - EXPECTING x INSTEAD FOUND y**

This error is caused by an empty CGENDATA file due to no SQL statements in the program.

**CGEN040E  UNSUPPORTED DATA TYPE CODE OF "x" HAS BEEN ENCOUNTERED**

This error occurs if a datatype was passed to DQSCGEN which may be valid for CA-Easytrieve/Plus but is not valid for DB2. Usually, a datatype of NUMERIC was passed to DQSCGEN.
CGEN050E  INVALID DECIMAL SCALE LENGTH "x" FOR DATA TYPE "y" HAS BEEN ENCOUNTERED

This error occurs if SCALE is less than zero or is greater than PRECISION.

CGEN051E  INVALID DECIMAL PRECISION OF "x"

This error occurs if the decimal precision is less than 1 or greater than 31.

CGEN060E  INVALID HOST VARIABLE LIST NUMBER OF ENTRIES

This internal error occurs if the host variable list counter does not match the number of detail entries.

CGEN070E  INVALID USER STATEMENT LENGTH

This internal error occurs if the length is less than zero (0).
IDD Interface Messages

The following list and explanation of operational diagnostic messages is generated by the CA-Easytrieve/Plus IDD interface. Note that these do not have a message ID.

******* ERROR ON

{BIND }
{READY} AT nn
{FIND }
{USE }

An internal error has occurred. Call Computer Associates Technical Support for assistance.

******* EXPECTED CONTINUATION NOT RECEIVED

A statement continuation was indicated, but end-of-file on the CA-Easytrieve/Plus IDD source input or a non-IDD statement has been detected.

******* IDD-NAME NOT FOUND OR UNAUTHORIZED

The SUBSCHEMA, FILE, or RECORD name was not found or is unauthorized in the IDMS dictionary. Verify the spelling and rerun your job.

******* NOT A VALID NAME

The indicated name is not valid or is used out of context.

******* NUMBER MUST BE POSITIVE INTEGER

The indicated word must be a positive integer.

******* PARAMETER IS INVALID

The indicated word is invalid as used in the current statement.

******* PARAMETER IS TOO LARGE

The value of the word is too large. Refer to the statement description for the valid range.
******* REQUIRED PARAMETER IS NOT CODED

Additional parameters are required. That is, parameter(s), subparameter(s), or their associated value(s) are missing. Refer to the statement syntax description for the correct information.

******* VERSION PARAMETER IS INVALID

The version parameter specified is invalid. Refer to the statement syntax description for correct information.

******* UNBALANCED PARENTHESES

Parentheses must be balanced across a statement. (See Chapter 3, “System Overview: Rules of Syntax.”)
SQL Supplemental Diagnostic Messages

The following messages are supplemental messages to the CA-Easytrieve/Plus diagnostic messages B204, or A046. Some error explanations reference message manuals. You should refer to the correct manuals for the version of SQL you are running. These manuals are:

- SQL/DS Messages and Codes for VSE (SH24-5019),
- SQL/DS Messages and Codes for VM/SP (SH24-5070),
- IBM Database 2 Messages and Codes (SC26-4113),
- ORACLE Error Messages and Codes Manual (3605).

SQL WARNING, CODE IS xxxx

A positive SQL code was returned from a “PREPARE” of the statement. You should look up the SQLCODE in the SQL message manuals.

xxx CURSORS, MAXIMUM USEABLE CURSORS AT RUNTIME ARE yy

More cursors were defined by the user program than were specified by installation parameters for the interface. Refer to Chapter 3 in the CA-Easytrieve/Plus Installation Guide to increase the number of predefined interface CURSORs.

SQL ERROR, SQL CODE IS xxxx

A negative SQL code was returned from a “PREPARE” of the statement. You should look up the SQLCODE in the IBM SQL message manuals.

KEYWORD IS INVALID OR AN UNSUPPORTED COMMAND: xxxxxxxx

The first word of the SQL statement must be an SQL command keyword.

STATEMENT CONTAINS INVALID CHARACTER OR TOKEN: xxxxxxxx

A secondary keyword of a command is not valid. In the case of a cursor name, the cursor name contains invalid characters.

THE OBJECT OF THE DECLARE STATEMENT IS NOT CORRECT

The object of the ‘DECLARE CURSOR’ must be a SELECT statement or an INSERT statement (SQL/DS only).
ANOTHER KEYWORD WAS EXPECTED

The SQL statement is incomplete; additional keywords were expected but not found.

CURSOR NAME MUST BE SPECIFIED

A valid cursor name was not found immediately after the SQL command word.

END OF STATEMENT EXPECTED

Extra characters were found beyond the valid SQL statement.

ERROR LIMIT EXCEEDED, PROCESSING OF STATEMENT SUSPENDED

Too many errors were encountered when trying to process the SQL statement. Further processing of the statement is suspended.

MESSAGE DATA: xxxxxxxx

This is a supplemental message for a previous message that displayed the SQLCODE. This message shows the information to be inserted in the message text when the SQLCODE is looked up in the message manual.

KEYWORD IN ERROR : xxxxxxxx

This is a supplemental message for a previous syntax error message. The invalid keyword or token is displayed.

SQL ERRORS FOUND

An ACCESS MODULE was not created for this program because SQL statement errors were encountered.

EMPTY OR INVALID ACCESS MODULE

An ACCESS MODULE was not created because the ACCESS MODULE does not contain any SQL table processing statements, that is, FETCH, INSERT, DELETE.
CURSOR NAME xxxxxxxxx PREVIOUSLY DEFINED

A cursor name is “DECLARED” only once.

“OPEN” COMMAND REQUIRED FOR CURSOR xxxxxxxxx

A cursor was referenced in a FETCH, UPDATE, or DELETE without an OPEN statement having been executed.

“DECLARE” STATEMENT REQUIRED FOR CURSOR xxxxxxxxx

The cursor name was used in an SQL statement without having been defined by an SQL “DECLARE CURSOR” statement.

“DECLARE FOR INSERT” REQUIRED FOR CURSOR xxxxxxxxx

The cursor name was found in an SQL “PUT” statement without having been defined by an SQL ‘DECLARE FOR INSERT’.

“DECLARE FOR SELECT” REQUIRED FOR CURSOR xxxxxxxxx

The cursor name was found in an SQL “FETCH” statement without having been defined by an SQL ‘DECLARE FOR SELECT’.

CURSOR xxxxxxxxx NOT REFERENCED IN AN SQL “FETCH” COMMAND

A cursor has been DECLAREd for “SELECT” but has not been referenced in a “FETCH” statement, or a cursor has been referenced in an UPDATE or DELETE “WHERE CURRENT OF CURSOR-NAME”, but has not been referenced in a “FETCH” statement.

CURSOR xxxxxxxxx NOT REFERENCED IN AN SQL “PUT” COMMAND

A cursor has been declared for “INSERT” but has not been referenced in a “PUT” statement.

CONFLICTING USE OF CURSOR xxxxxxxxx

A cursor name has been declared for “INSERT” use with a “PUT” statement, yet the same cursor name is being referenced in either a FETCH, UPDATE, or DELETE “WHERE CURRENT OF CURSOR-NAME”.

START UP ERROR REPORTED BY INIT

The SQL interface was unable to initialize itself. When attempting to terminate the interface, resources were unable to be released due to the error in the initialization.

ERROR WHILE EXECUTING AN INTERNAL SQL STATEMENT. SQL CODE IS xxxx

A severe error has occurred in the SQL interface.

- For DB2, the error is related to the Call Attach facility.
- For SQL/DS, the installation of the application plan probably was not successful.

Look up the SQLCODE in the IBM message manuals.

INSUFFICIENT STORAGE TO LOAD xxxxxxxxx

Not enough storage was available to load the module.

MODULE “xxxxxxxx” NOT FOUND

The module was not found.

UNABLE TO LOAD MODULE “xxxxxxxx”

An error occurred loading a module. Look for IBM error messages that accompany this message.

UNABLE TO OBTAIN GLOBAL STORAGE

Storage was not available for the SQL interface to run.

ALL INTERNAL CURSORS HAVE BEEN USED, NO CURSORS REMAINING FOR DECLARE

During the execution of the user module, more cursors were defined and concurrently opened than were predefined by the interface during the installation. User should refer to the installation documentation for information about increasing the number of predefined cursors for the interface.
CURSOR xxxxxxx MUST BE DECLARED BEFORE EXECUTING RELATED CURSOR STATEMENTS

A cursor must be defined in an SQL “DECLARE” statement before it can be referenced in any subsequent SQL statements.

SQL CONNECT ERROR, SQLCODE = xxxx

An explicit “CONNECT” was executed by the interface on behalf of the programmer in order to preprocess the SQL statements. A non-zero SQL code was returned. User should verify that the proper authorization exists for the given userid to access the SQL/DS subsystem. Further processing of the user program is suspended.

SQL CREATE ACCESS MODULE ERROR, SQLCODE = xxxx

An SQL “CREATE PROGRAM” was executed by the interface on behalf of the programmer in order to create an ACCESS MODULE for the SQL statements. A non-zero SQL code was returned to the interface. The user should refer the SQL code to his/her system’s programmer. Further processing of the user program is suspended.

RELEASE LEVEL MISMATCH BETWEEN DB2 AND THE CALL ATTACH FACILITY

The release level of the CALL ATTACH facility and that of the DB2 subsystem do not match. The interface is unable to execute in this environment. Further processing of the user program is suspended.

DB2 SUBSYSTEM “xxxx” IS NOT ACTIVE

The DB2 subsystem ID that was specified by the user is not currently active. The interface is unable to execute in this environment. Further processing of the user program is suspended.

DB2 SUBSYSTEM “xxxx” DOES NOT EXIST

The interface was unable to establish a connection to the DB2 subsystem ID that was specified by the user. The user should correct the subsystem ID and rerun the job. Further processing of the user program is suspended.

PLAN NAME “xxxxxxxxx” NOT AUTHORIZED

The user has not been GRANTed authorization to execute the plan. Further processing of the user program is suspended.
DB2 ERROR IN SUBSYSTEM “zzzz”; RETURN CODE xxxx, REASON CODE X’yy’

An unanticipated error has been encountered when attempting to use the DB2 ‘CALL ATTACH’ facility. Contact your system programmer. Further processing of the user program is suspended.

PLAN “xxxxxxxx” NOT FOUND

The plan name specified does not exist within the DB2 subsystem.

MAXIMUM CONNECTIONS TO DB2 SUBSYSTEM “xxxx” EXCEEDED

The maximum number of concurrent connections has been exceeded. Retry the job later. Connection limits are specified during DB2 installation for TSO, Batch, and Call Attachment environments.

DB2 DENIED ACCESS TO SUBSYSTEM “xxxx”. DB2 EXECUTING IN RESTRICTED ACCESS MODE

A request to connect to a DB2 subsystem has been rejected. DB2 was started in restricted access mode. Only userids authorized to perform maintenance functions are permitted access.

INTERNAL ERROR xxxx yyyyyyyyy

An internal error has occurred within the SQL interface. xxxx is a code defining the error. yyyyyyyyy is text data associated with the error.

STATIC COMMAND PROGRAM xxxxxxxx NOT FOUND

The program requested an SQL static-only execution but the command program could not be found. Verify that the secondary steps ran successfully.

TIME STAMPS DO NOT AGREE

The CA-Pan/SQL time stamp of the SQL static command program does not match that of the CA-Easytrieve/Plus program. Verify the correct running of the secondary steps.

xxxxxxxx FOUNDED INSTEAD OF xxxxxxxx

The SQL command program was found to contain the wrong program. Verify the correct running of the secondary steps.
-818 RETURNED FOR STATIC COMMAND

DB2 detected a time stamp problem between the DBRM in the application plan and the command program. The GENDATA file created by the CA-Easytrieve/Plus SQL program needs to be reprocessed. Rerun the DB2BIND proc.

-911 RETURNED FOR COMMAND PROGRAM

DB2 detected an authorization problem between the user and the application plan. The userid executing the application plan was not granted RUN authority by the userid who created it. Either execute with the same userid that created the plan, or grant access on the plan to the userid that needs to execute it.

INSUFFICIENT STORAGE TO LOAD xxxxxxxxx

There was not enough storage to load the command program. Increase your storage size (REGION for OS/390, GETVIS for VSE).

UNABLE TO EXECUTE USING STATIC SQL

The execution mode is STATIC-ONLY. However, the user program cannot be executed using static SQL at this time. The reason is reported in the next error message.

EXECUTION MODE CHANGED TO DYNAMIC EXECUTION DUE TO ERROR

The OPEN for the GENDATA DD statement failed. The execution mode is changed to DYNAMIC to permit the remainder of the program to be compiled. Verify the presence of the GENDATA DD statement. Look on the console log for information relating to the OPEN failure.

NUMBER OF HOST VARIABLES EXCEEDS 999

An SQL statement cannot have more than 999 host variables. If necessary, convert the SQL statement into multiple statements.

DELETE FAILED FOR STATIC COMMAND PROGRAM xxxxxxxxx

CA-Pan/SQL was unable to delete the named command program. Check the console log for DB2 error messages.
FETCH MUST BE EXECUTED PRIOR TO UPDATE OR DELETE WHERE CURRENT OF CURSOR

A cursor must be executed in an SQL FETCH statement before it can be referenced in any subsequent SQL statement.

THE CURSOR MUST BE DECLARED FOR UPDATE IN ORDER TO UPDATE OR DELETE WHERE CURRENT OF

A cursor must be defined in an SQL DECLARE statement before it can be referenced in any subsequent SQL statement.

OS/390 OPEN ERROR ON GENDATA FILE, OPEN RC=xxxxxxxx

The open on the GENDATA FILE failed for the given reason. Report the error to your systems programmer.

INSUFFICIENT FILE SPACE FOR FILE=GENDATA

A PUT to the GENDATA file failed due to insufficient space. Increase the disk space for file GENDATA.

OS/390 CLOSE ERROR ON GENDATA FILE, OPEN RC=xxxxxxxx

The close on the GENDATA FILE failed for the given reason. Report the error to your systems programmer.
Introduction

This appendix contains a list of CA-Easytrieve/Plus symbols and reserved words. Reserved words are keywords that make up the CA-Easytrieve language, so they cannot be used as labels or identifiers.

The reserved words are listed in alphabetical order. Associated with each symbol is one or more references. The references describe the various ways in which you can use the symbol. An R in the column after the symbol indicates it is reserved.

Symbol References

<table>
<thead>
<tr>
<th>Special Symbol</th>
<th>Reserved</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Syntax delimiter (period)</td>
<td>Macro parameter concatenation (period)</td>
</tr>
<tr>
<td>&lt;</td>
<td>Conditional expression</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>Conditional expression</td>
<td></td>
</tr>
<tr>
<td>(</td>
<td>Syntax delimiter (left parenthesis)</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>Syntax delimiter (colon)</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>Assignment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuation of statements and words</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DISPLAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LINE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TITLE</td>
<td></td>
</tr>
<tr>
<td>&amp;</td>
<td>Macro variable prefix</td>
<td></td>
</tr>
<tr>
<td>Special Symbol</td>
<td>Reserved</td>
<td>Reference</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>Assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Comment statement</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFINE</td>
</tr>
<tr>
<td>)</td>
<td></td>
<td>Syntax delimiter (right parenthesis)</td>
</tr>
<tr>
<td>¬&lt;</td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POINT</td>
</tr>
<tr>
<td>¬&gt;</td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td>¬=</td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td>-</td>
<td></td>
<td>Assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Continuation of statements and words</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISPLAY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TITLE</td>
</tr>
<tr>
<td>**</td>
<td>R</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>/</td>
<td></td>
<td>Assignment</td>
</tr>
<tr>
<td>'</td>
<td></td>
<td>Syntax delimiter (single quote)</td>
</tr>
<tr>
<td>%</td>
<td></td>
<td>Macro invocation</td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POINT</td>
</tr>
<tr>
<td>,</td>
<td></td>
<td>Syntax delimiter (comma)</td>
</tr>
<tr>
<td>=</td>
<td></td>
<td>Assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Conditional expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td>POINT</td>
</tr>
<tr>
<td>@</td>
<td>R</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>
Reserved Words

The following list includes all CA-Easytrieve/Plus reserved words.

<table>
<thead>
<tr>
<th>Reserved Word</th>
<th>Reserved Word</th>
<th>Reserved Word</th>
<th>Reserved Word</th>
<th>Reserved Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER-BREAK</td>
<td>ERROR</td>
<td>LT</td>
<td>RETURN-CODE</td>
<td></td>
</tr>
<tr>
<td>AFTER-LINE</td>
<td>EXIT</td>
<td>MASK</td>
<td>ROLLBACK</td>
<td></td>
</tr>
<tr>
<td>AFTER-SCREEN</td>
<td>EXTERNAL</td>
<td>MATCHED</td>
<td>ROW</td>
<td></td>
</tr>
<tr>
<td>AIM</td>
<td>EZLIB</td>
<td>MEND</td>
<td>S</td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>F1...F36</td>
<td>MESSAGE</td>
<td>SCREEN</td>
<td></td>
</tr>
<tr>
<td>ATTR</td>
<td>FETCH</td>
<td>MOVE</td>
<td>SEARCH</td>
<td></td>
</tr>
<tr>
<td>BEFORE</td>
<td>FILE</td>
<td>MSGTART</td>
<td>SECONDARY</td>
<td></td>
</tr>
<tr>
<td>BEFORE-BREAK</td>
<td>FILE-STATUS</td>
<td>NE</td>
<td>SELECT</td>
<td></td>
</tr>
<tr>
<td>BEFORE-LINE</td>
<td>FIL</td>
<td>NEWPAGE</td>
<td>SEQUENCE</td>
<td></td>
</tr>
<tr>
<td>BEFORE-SCREEN</td>
<td>FINAL</td>
<td>NOMASK</td>
<td>SIZE</td>
<td></td>
</tr>
<tr>
<td>BUSHU</td>
<td>FIRST</td>
<td>NOPRINT</td>
<td>SKIP</td>
<td></td>
</tr>
<tr>
<td>BY</td>
<td>FIRST-DUP</td>
<td>NOT</td>
<td>SORT</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>FOR</td>
<td>NOVERIFY</td>
<td>SQL</td>
<td></td>
</tr>
<tr>
<td>CASE</td>
<td>GE</td>
<td>NOP</td>
<td>STOP</td>
<td></td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td>GET</td>
<td>OF</td>
<td>SYSDATE</td>
<td></td>
</tr>
<tr>
<td>CHK</td>
<td>GO</td>
<td>SYSDATE-LONG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHKP-STATUS</td>
<td>GOTO</td>
<td>OR</td>
<td>SYSDATE-LONG</td>
<td></td>
</tr>
<tr>
<td>CLEAR</td>
<td>GQ</td>
<td>OTHERWISE</td>
<td>SYSTIM</td>
<td></td>
</tr>
<tr>
<td>CLOSE</td>
<td>GT</td>
<td>PA1...PA3</td>
<td>SYSN</td>
<td></td>
</tr>
<tr>
<td>COLOR</td>
<td>HEADING</td>
<td>PAGE-COUNT</td>
<td>SYSIPT</td>
<td></td>
</tr>
<tr>
<td>COMMIT</td>
<td>HEX</td>
<td>PAGE-NUMBER</td>
<td>SYSSNAP</td>
<td></td>
</tr>
<tr>
<td>CONTROL</td>
<td>HIGH-VALUES</td>
<td>PARM-REGISTER</td>
<td>SYSPRINT</td>
<td></td>
</tr>
<tr>
<td>COPY</td>
<td>IDD</td>
<td>PATH-1D</td>
<td>SYSTIME</td>
<td></td>
</tr>
<tr>
<td>CURSOR</td>
<td>IDMS</td>
<td>PATTERN</td>
<td>TALLY</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>IF</td>
<td>PERFORMANCE</td>
<td>TERM-COLUMNS</td>
<td></td>
</tr>
<tr>
<td>DECLARE</td>
<td>IN</td>
<td>POINT</td>
<td>TERMINATION</td>
<td></td>
</tr>
<tr>
<td>DEFAULT</td>
<td>INSERT</td>
<td>POS</td>
<td>TERM-NAME</td>
<td></td>
</tr>
<tr>
<td>DEFINE</td>
<td>JOB</td>
<td>PRIMARY</td>
<td>TERM-ROWS</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>JUSTIFY</td>
<td>PRINT</td>
<td>TITLE</td>
<td></td>
</tr>
<tr>
<td>DENWA</td>
<td>KANJI-DATE</td>
<td>PROC</td>
<td>TO</td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>KANJI-DATE-LONG</td>
<td>PROCEDURE</td>
<td>TRANSFER</td>
<td></td>
</tr>
<tr>
<td>DL1</td>
<td>KANJI-TIME</td>
<td>PROGRAM</td>
<td>TRC</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td>KEY</td>
<td>PUT</td>
<td>UNIQUE</td>
<td></td>
</tr>
<tr>
<td>DUPLICATE</td>
<td>KEY-PRESSED</td>
<td>READ</td>
<td>UNTIL</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>KOKUGO</td>
<td>RECORD</td>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>ELSE</td>
<td>KUN</td>
<td>RECORD-COUNT</td>
<td>UPPERCASE</td>
<td></td>
</tr>
<tr>
<td>ELSE-IF</td>
<td>LAST-DUP</td>
<td>RECORD-LENGTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>END</td>
<td>LEM</td>
<td>REFRESH</td>
<td>USERID</td>
<td></td>
</tr>
<tr>
<td>END-CASE</td>
<td>LEVEL</td>
<td>RELEASE</td>
<td>VALUE</td>
<td></td>
</tr>
<tr>
<td>END-DO</td>
<td>LIKE</td>
<td>RENUM</td>
<td>VERIFY</td>
<td></td>
</tr>
<tr>
<td>END-IF</td>
<td>LINE</td>
<td>REPEAT</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>END-PROC</td>
<td>LINE-COUNT</td>
<td>REPORT</td>
<td>WHEN</td>
<td></td>
</tr>
<tr>
<td>ENDPAGE</td>
<td>LINE-NUMBER</td>
<td>REPORT INPUT</td>
<td>WHILE</td>
<td></td>
</tr>
<tr>
<td>ENDTABLE</td>
<td>LINK</td>
<td>RESHOW</td>
<td>WORK</td>
<td></td>
</tr>
<tr>
<td>ENTER</td>
<td>LIST</td>
<td>RESTART</td>
<td>WRITE</td>
<td></td>
</tr>
<tr>
<td>EOF</td>
<td>LOWVALUES</td>
<td>RETRIEVE</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>EQ</td>
<td>LQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introduction

The options table (EZTOPT) provides the basic parameters that control CA-Easytrieve/Plus operation. This appendix lists them in alphabetical order. Space is provided for you to note the options which were selected at system installation.

The PARM statement provides the ability to temporarily override some of your system's standard options. The PARM statement and its parameters alter the program environment only as long as the program is running. For more information on customizing your program's environment, refer to the PARM Statement in Chapter 4, “Environment Definition.”

Selectable Options

ABEXIT=xxxxxx

Indicates the type of processing that CA-Easytrieve/Plus performs when a program check occurs within a program. Valid values are NO, NOSNAP, and SNAP. Override is through the ABEXIT parameter of the PARM statement.

- **NO**—indicates that CA-Easytrieve/Plus should not process any program checks.
- **SNAP**—indicates that CA-Easytrieve/Plus should intercept any program checks in the range 1 through 11 and produce an Error Analysis Report. See Chapter 15, “System Facilities.”
- **NOSNAP**—performs the same functions as SNAP except that the CA-Easytrieve/Plus storage areas are not dumped.

Default - SNAP Standard -
ACROSS=nnn

Specifies the number of labels to print across the print line when using the LABELS parameter of the REPORT statement. The valid range is 1 to 127. Override is through the LABELS parameter of the REPORT statement.

Default - 4  
Standard -

ALTSEQ=(xxx[,modname])

Specifies the use of an alternate collating sequence table for the sort process. For detailed information on this facility, refer to the SORT Reference manual for your installation. This parameter is primarily used where the English alphabet is not used. The valid values for xxx are YES and NO.

- YES — indicates you want to use an alternate collating table.
- NO — indicates the facility is not used.
- modname — if YES is specified, provide the name of table module.

Override for both parameters is through the SORT parameter of the PARM statement.

Default - (NO,EZTPAQTT)  
Standard -

BLOCKO=x

(OS/390 only) Specifies whether a system-determined blocksize is used for files that do not have logical record length and blocksize coded. A zero is passed to the operating system which determines the optimum blocksize. This feature should be used only if your operating system supports the use of the IBM system-determined blocksize. Override is through the BLOCKSIZE parameter of the FILE statement.

- N — indicates that the system does not determine the blocksize for data sets. It must be specified through the JCL or FILE statement.
- D — indicates that the system determines the blocksize for disk and tape data sets. DSORG does not have to be coded in the JCL with this option.
- P — indicates that the system determines the blocksize for PRINTER data sets.
- A — indicates that the system determines the blocksize for disk, tape and PRINTER data sets.

Default - N  
Standard -
BUFNO=nnn

Specifies the number of I/O buffers for each sequential file. VFM, VSAM, and VSE sequential disk files assigned to FBA devices do not use this information. Valid values for VSE are 1 or 2; for OS/390, they are 1 to 255. Override is by the BUFNO parameter of the FILE statement.

Default - 2
Standard -

CALCDUP=xxx

Specifies whether CALC records with duplicate keys are to be retrieved for the CA-IDMS RETRIEVE statement. Valid values are YES and NO.
- YES—indicates that all root records with the same tickler file key are returned.
- NO—indicates that only the first of the duplicate records is returned.

Override is through the DUPS/NODUPS keyword of the RETRIEVE statement.

Default - NO
Standard -

CLIST=xxxxxxx

Specifies whether a condensed listing of the executable statements for each JOB and SORT should be produced. Valid values are CLIST and NOCLIST. Override is through the DEBUG parameter of the PARM statement.

Default - NOCLIST
Standard -

CMSVFM=xx

(CMS only) Specifies the file mode of the CMS minidisk used for the VFM work file when operating under VM/CMS. The minidisk must be accessed for read/write operations. The CMS file name used is yyyVFM yyyVFM xx where yyy is the value of the WKDSNPF option (the default is EZT). The CMS minidisk must be blocked at 1024.

xx must be one or two characters that are a valid CMS file mode. If only one character is specified, a blank is automatically supplied as the second character. No override is available.

Default - A1
Standard -
COMPNME='xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

Specifies the value which is centered in the title area of the compiler listing. This facility enables you to specify the name of your company (50 characters maximum). Enclose the name in single quotes. No override is available.

Default - COMPUTER ASSOCIATES, INTL. FIELD INSTALLATION

Standard -

COMPSTR=nnnnK

(VSE only) This option indicates the amount of storage that CA-Easytrieve/Plus makes available for loading user-supplied routines at compilation time. Examples of routines that CA-Easytrieve/Plus loads during compilation include DBCS code system conversion programs and Source Conversion routines.

A VSE error occurs during program loading if this space is too small. The valid range for this parameter is 0 to 4096. No override is available.

Default - 4K

Standard -

DATE=xxxxxx

Specifies the format of the date placed at the top of the compiler listing and also stored in the system defined SYSDATE field. Valid values are MMDDYY, YYMMDD, or DDMMYY where MM refers to the month, DD refers to the day, YY refers to the year. No override is available.

Note: Also, see the LONGDTE= option which is used to specify the four-digit year.

Default - MMDDYY

Standard -
DATEADJ=nnnn

This option indicates a base year that CA-Easytrieve/Plus uses to calculate the adjusted system date. CA-Easytrieve/Plus reads the system date and adds the current century to the year value in the system date. CA-Easytrieve/Plus then reduces this year value by the year coded as the DATEADJ value, giving a year that CA-Easytrieve/Plus uses as the YY portion of the SYSTEM-DATE. For example, a year value of 1985 is calculated to be year 30 if DATEADJ is set to 1955.

Valid values for this option are 0 to 9999. If the result of the calculation results in a negative number, then CA-Easytrieve/Plus uses a value of zero for the year portion of the system date. Should the year value exceed 99, then CA-Easytrieve/Plus uses the last two bytes for the year portion of the system date. No override is available.

Default - 0 Standard -

DEVICE=xxxx

(VSE only) Specifies the default device type for CA-Easytrieve/Plus user files. Valid values are 3330, 3340, 3350, 3375, 3380, 3390, FBA, TAPE, and DISK. DISK indicates that the value of the DISK option is used. Override is through the DEVICE parameter of the PARM statement or the DISK parameter of the FILE statement.

Default - DISK Standard -

DISK=xxxx

(VSE only) Specifies the default type of disk. This value is used when DISK is specified for the options VFMDEV, DEVICE, or MACDEV. It is also used when you specify the DISK option on a FILE statement. Valid values are 3330, 3340, 3350, 3375, 3380, 3390, or FBA. For programs executing on DOS AF2 or later, the disk device type is determined when the file is opened.

Default - 3380 Standard -
Indicates if the IBM DB2-DL/I batch support is being used to access DB2 and DL/I data from application programs with coordinated recovery. Valid values are YES and NO. A value of NO results in the DB2 Call Attach Facility being used by the SQL Interface for CA-Easytrieve/Plus. This option requires the use of SQL COMMIT and ROLLBACK commands to syncpoint the DB2 environment independent of the IMS environment.

A value of YES means DL/I is coordinating recovery of both DB2 and DL/I through a two-phase commit process. A commit point occurs when a program issues an IMS checkpoint call. Therefore, DB2 COMMIT and ROLLBACK commands are not valid and their execution results in a nonzero SQLCODE (-925 and -926).

**Note:** The user must code the correct JCL, in addition to specifying the applicable value for this parameter.

**Default - NO**

**Standard -**

**(VSE only)** Specifies whether DL/I is to use V command codes for automatic input. DL/I must be release 1.6 or later to use command codes in SSAs. Valid values are YES or NO. No override is available.

**Default - YES**

**Standard -**

Specifies whether a data map of the library and report files is generated. Valid values are DMAP and NODMAP. Override is through the DEBUG parameter of the PARM statement.

**Default - NODMAP**

**Standard -**

Specifies the number of lines per label. The valid range is 1 to 32767. Override is by the LABELS parameter of the REPORT statement.

**Default - 6**

**Standard -**
DTLCTL=xxxxx

Specifies the method of printing the value of control fields on detail lines in a control report. For specific information on the options available, refer to the DTLCTL parameter of the REPORT statement in Chapter 11, “Report Processing.” Valid values are EVERY, FIRST, or NONE. Override is through the DTLCTL parameter of the REPORT statement.

Default - FIRST

ENVIRON = {NONE}
{COBOL}

(OS/390 only) Instructs CA-Easytrieve/Plus to establish the proper execution environment prior to calling any COBOL subprograms. The environment is established prior to each JOB activity that contains a CALL statement and is terminated after the activity for which it was established. When set in the Options Table, it establishes the default used for all CA-Easytrieve/Plus programs. This default can be overridden using the ENVIRONMENT parameter of the PARM or JOB statement. For more information about the ENVIRONMENT parameter, see Chapter 14, “Subprograms.”

Default - NONE

EXITSTR=nnnnK

(VSE only) Indicates the amount of storage made available at execution time for user-called programs and non-VSAM I/O exits. VSAM I/O exits are loaded into GETVIS space. A VSE error occurs during program loading if this space is too small. The valid range for this parameter is 0 to 4096. Override is through the EXITSTR parameter of the PARM statement.

Default - 8K

FLDCHK=xxxxxxxx

Indicates whether CA-Easytrieve/Plus validates data references to each field during execution. A data reference is invalid if a field-name was referenced in a file which had no active record. An invalid reference can cause a program check or a reference to invalid data. Valid values are FLDCHK and NOFLDCHK. Override is through the DEBUG parameter of the PARM statement.

Default - FLDCHK
**Flow=xxxxxx**

Indicates whether the FLOW option should be active at execution time. FLOW traces statement execution. Valid values are FLOW and NOFLOW. This option or STATE is necessary if the statement number prefix is desired on execution-time diagnostic messages. Override is through the DEBUG parameter of the PARM statement.

Default - NOFLOW

**FlowSiz=nnnn**

Specifies the number of trace entries available for the FLOW option. Each entry requires 2 bytes of storage. The valid range is 1 to 4096. Override is through the DEBUG parameter of the PARM statement.

Default - 100

**IddExit=xxx**

Specifies whether the IDD interface is invoked as a modify SYSIN (SYSIPT) exit. Valid values are YES and NO. No override is available.

Default - NO

**Idmsnam=**

For internal use only.

Default -

**Lablsiz=nnn**

Specifies the number of print positions on a printed label. The valid range is 1 to the value of LINESIZ. Override is through the LABELS parameter of the REPORT statement.

Default - 30

**Linesiz=nnn**

Specifies the maximum length of a report line. This value also defines the length of the compiler output lines. The valid range is 120 to 204. Override is through the LINESIZE parameter of the REPORT statement, but the value specified by LINESIZE cannot be greater than the maximum set here. The override is only applicable to report output. The specified length does not include the carriage control character.

Default - 132
LIST=(xxxxxx,yyyyyy)

Indicates whether to list PARM and/or FILE statistics for each job. Valid values are PARM or NOPARM, FILE or NOFILE. Override is through the LIST parameter of the PARM statement.

Default - (PARM,FILE) Standard -

LONGDTE=xxx

Specifies whether SYSDATE or SYSDATE-LONG is the default date that appears on TITLE 01 on REPORTS.

- YES displays SYSDATE-LONG on reports and the compile listing.
- NO displays SYSDATE on reports and the compile listing.

Valid values are YES or NO. Override for reports is through LONGDATE or SHORTDATE on the REPORT statement.

Note: The format of SYSDATE-LONG respects the date format as specified by the DATE= option.

Default - NO Standard -

MAC#LIB=n

Specifies the number of CA-Panvalet or CA-Librarian libraries to be searched if the CA-Panvalet or CA-Librarian macro library support is used. The valid range for this parameter is 1 to 9. No override is available.

This value is used as a suffix to the name specified by MACDDN. Thus, if:

```
MACDDN  = PANDD
MAC#LIB = 2
```

In the JCL, two ddnames are:

```
PANDD1
PANDD2
```

Default - 1 Standard -

MACDDN=xxxxxxxx

Indicates the name used by the five standard CA-Easytrieve/Plus macro library facilities to reference the desired macro library. Following is a description of how each facility uses the option in a Macro Library.
CA-Panvalet or CA-Librarian

Defines the first five characters (usually PANDD) of the ddname used to reference each CA-Panvalet library, or the first six characters (usually MASTER) of the CA-Librarian ddname. In VSE, the library is also referenced through the SYS number. See the MACSYS# parameter for details on accessing alternate VSE library names.

PDS

(OS/390 only) defines the ddname used to reference the Partitioned Data Set (PDS) to be used as a macro library. This parameter must be a valid ddname.

SSL

(VSE only) MACDDN is not used for the SSL macro facility. Use a LIBDEF statement to define the source library to be searched for macro members.

VSAM

Defines the OS/390 ddname or VSE file-name used to reference the VSAM macro library. This parameter must be a valid OS/390 ddname or VSE DLBL.

No override is available.

Default - PANDD Standard -

MACDEV=xxxx

(VSE only) Specifies the device type for the CA-Panvalet or CA-Librarian macro library. Valid values are 3330, 3340, 3350, 3375, 3380, 3390, FBA, or DISK. If you specify DISK, the value specified for the DISK option is used. No override is available.

Default - DISK Standard -

MACRO=xxxx

Specifies the type of macro library support. Valid values for xxxx are NO, LIBR, PAN, PDS, SSL, VSAM, or USER. When the type is PAN, the optional modname can be specified to identify the interface routine. For type USER, the modname must be specified to identify the interface routine name.

When the type is SSL, the optional modname can be specified as the default macro subtype. The default subtype for CA-Easytrieve/Plus SSL members is Z. The subtype can be specified on the macro invocation statement. No other override is available.

Default - (PAN,PANMODI) Standard -
MACSYS#=nnn

(VSE only) Specifies the starting SYS number for CA-Panvalet or CA-Librarian macro libraries. The valid range is 0 to 240. No override is available. For CA-Panvalet, a value of 0 indicates that the SYS number to be used is the one defined when the product was installed.

Default - 0 Standard -

MONEY=

Specifies the one character currency symbol used as the floating currency symbol in print edit masks. No override is available.

Default - $ Standard -

NEWPAGE=xxx

Indicates whether a skip to channel 1 (top of form) should occur on the first line of every printed label. Use of this parameter requires a special carriage tape or FCB. Valid values are YES and NO. A value of NO can be overridden through the LABELS parameter of the REPORT statement. A value of YES cannot be overridden.

Default - NO Standard -

NUMERIC=(xxxxxx,yyyyyy)

Specifies the numeric edit characters for thousands and decimal point respectively. The x and y can be COMMA or PERIOD. For the European form of a number, code NUMERIC=(PERIOD,COMMA). No override is available.

Default - (COMMA,PERIOD) Standard -
NUMWORK={  } {DA}

Specifies the number of work areas used by the sort. This option has different meanings for OS/390 and VSE, as indicated below.

- **VSE**—enables two types of sort work data sets -- SD and DA. An SD type is a single extent data set; there can be from one to eight data sets. To specify the use of SD type data sets, code NUMWORK=n, where n is in the range 1 to 8. The DA type is a single data set which can have multiple extents. To specify the use of a DA type data set, code NUMWORK=DA.

- **OS/390**—indicates whether the work data sets should be dynamically allocated by the sort program. If the value of nn is 0, data sets are not dynamically allocated; they must be defined by DD statements. This technique is required if your operating system or your sort program does not support dynamic allocation of data sets. A value of 1 to 31 indicates the sort program should dynamically allocate the specified number of work data sets.

Override is through the WORK parameter of the PARM and SORT statements.

Default - 3

**PAGESIZ=(lllll,ddddd)**

lllll, LINE page size, specifies the maximum report page size for LINE statements. The valid range for LINE page size is 1 to 32767. This value also defines the page size of the compiler output lines. ddddd, DISPLAY page size, specifies the maximum report page size for report procedure DISPLAY statements. The valid range for DISPLAY page size is 0 to 32767. A value of zero for DISPLAY page size indicates an infinite report page size for report procedure DISPLAYs. Override is through the PAGESIZE parameter of the REPORT statement.

Default - 58,0

**PAGEWRD=xxxxxxxxxx**

Specifies the desired spelling for the English word PAGE for non-English language installations. The specified word replaces the word PAGE in the first title line of each report and at the top of each page of the CA-Easytrieve/Plus compiler output listing. The value must be from 1 to 10 characters long. No override is available.

Default - PAGE

**PLACE=xxxxxx**

For internal use only.

Default - none
PMAP=xxxxxx

Indicates whether a listing of the machine language program that CA-Easytrieve/Plus generates is produced. Valid values are PMAP and NOPMAP. Override is by the DEBUG parameter of the PARM statement.

Default - NOPMAP  Standard -

PRESIZE=nnnnn

Specifies the record length of the compiler work file. The valid range is 512 to 32767. Override is through the PRESIZE parameter of the PARM statement. If the value is too small for a particular program, CA-Easytrieve/Plus issues error message B079. This parameter should be increased only when it becomes unacceptable to override using the PARM statement. Determine the size required by use of the PARM statement override.

Default - 512  Standard -

PREPNME=xxxxxxxx

Specifies the default name of the access module to be created for SQL. Override is through the PREPNAME parameter of the PARM statement.

Default - EASYPLUS  Standard -

REWIND=xxxxxx

(VSE only) Specifies the method used for tape positioning. Valid values are:

- YES — specifies that the tape is rewound before and after it is used.
- UNLOAD — specifies that the tape is rewound before use and unloaded after it is used.
- NORWD — specifies that the tape should not be positioned before or after use.

Override is through the TAPE parameter of the FILE statement.

Default - YES  Standard -

SCANCOL=(mm,nn)

Establishes the inclusive column bounds scanned for CA-Easytrieve/Plus source input. The value for mm is the scan start column; the value for nn is the scan stop column. The valid range for both values is 1 to 80, where mm is less than nn. No override is available.

Default - (1,72)  Standard -
SEPDATEx

Specifies the one-character date separator used in the current date for all CA-Easytrieve/Plus listings and reports. The character, represented by the lowercase x, can be any character. No override is available.

Default = / Standard -

SEPTIMEx

Specifies the one-character time separator used in the current time for all CA-Easytrieve/Plus listings and reports. The character, represented by the lowercase x, can be any character. No override is available.

Default = . Standard -

SINXIT=modname

Specifies the name of a user supplied SYSIN/SYSIPT exit routine. The modname must be a valid program name. No override is available. More information on the SYSIN/SYSIPT exit capability can be found in the CA-Easytrieve/Plus Installation Guide, Appendix F, “Unit Record Exits.”

Default - none Standard -

SKIP=nnn

Specifies the number of blank lines inserted before each LINE 01 is printed, except for the first LINE 01 after a page heading. The valid range is 0 to 255 or PAGESIZ, whichever is smaller. Override is through the SKIP parameter of the REPORT statement.

Default - 0 Standard -

SORTMSG=xxxxxxxx

Specifies the level of messages output from your sort program. Valid values are NONE, ALL, CRITICAL, and DEFAULT. DEFAULT causes the sort to output the level of messages which was specified when your sort was installed. Override is through the SORT parameter of the PARM statement.

Default - DEFAULT Standard -

SORTMSR=xxxxxxxx

Specifies the routing of sort messages. Valid values are PRINTER and CONSOLE. Override is through the SORT parameter of the PARM statement. This parameter is ignored if you specify SORTMSG=DEFAULT.

Default - PRINTER Standard -
**SORTNAM=modname**

Defines the name of the sort program on your operating system. The modname must be a valid program name. No override is available.

Default - SORT    Standard -

**SORTOPT=(wwww,xxxxxxx,yyyyyy,zzzzzzz)**

(VSE only) Specifies the setting of options available for the VSE sort program. Refer to your installation's sort reference manual for detailed information.

- wwww — can be TP or NOTP. The TP option requests that the merge order of the sort be limited to reduce contention for I/O resources. This reduces the impact on any active TP systems.
- xxxxxxx — can be REAL or VIRTUAL. VIRTUAL inhibits the sort from fixing pages. The option VIRTUAL is required when CA-Easytrieve/Plus is run in an ICCF interactive partition. If CA-Easytrieve/Plus is executed in the batch environment, change the default to REAL for better performance.
- yyyyyy — can be DIAG or NODIAG. DIAG specifies that messages containing diagnostic information are produced.
- zzzzzzz — can be ERASE or NOERASE. ERASE specifies that work data sets used during a sort are erased at the end of the sort. ERASE is ignored for work data sets on tape.

All of these options can be overridden through the SORT parameter of the PARM statement. We recommend that you modify only the xxxxxxx option since changes to any other options tend to degrade performance. Select these options through the override on the PARM statement only when necessary.

Default - (NOTP,VIRTUAL,NODIAG,NOERASE)

Standard -

**SORTPRT=xxxxxx**

(OS/390 only) Specifies a valid ddname for sort messages. No override is available.

Default - SYSOUT    Standard -
SORTRLS=nnnnK

Specifies the amount of free storage to be made available after the sort is invoked. This space may be required when the input or output file is controlled by an exit and the exit needs to allocate storage either explicitly or implicitly (for example, open a file). The valid range is 0 to 1024. Override is through the PARM statement.

Default - 0K
Standard -

SORTSZ={mmmmK}
{MAX}
{MAX-nnnnK}

Specifies the maximum amount of storage that the sort program is allowed to use. If the value exceeds the amount of storage available, the amount available is used. Valid values for mmmm and nnnn are 16 to 4096. MAX indicates that the sort can use all available storage. '-nnnnK' is the amount of storage released after the MAX amount has been reserved. Override is through the SORT parameter of the PARM statement.

Default - MAX
Standard -

SORTWK#=(nnn,...)

(VSE only) Specifies the sort work data set logical assignments for SD type work data sets. You can specify one to eight assignments; nnn can be from 001 to 240, or 255. A value of 255 indicates that the sort should use the logical unit assigned during sort program installation. See your sort reference manual for details. Override is through the SORT parameter of the PARM statement.

Default - (255,255,255,255,255,255,255,255)
Standard -

SORTWRK=xxxxx

(OS/390 only) Specifies the sort work device type for sort programs which dynamically allocate their sort work data sets. The value of xxxxx can be any particular or generic device name which is valid for your operating system. Override is through the SORT parameter of the PARM statement.

Default - SYSDA
Standard -

SPACE=nnn

Specifies the number of spaces to be inserted between fields specified on TITLE and LINE statements. 0 to LINESIZ - 2 is the valid range. Override is through the SPACE parameter of the REPORT statement.

Default - 3
Standard -
**SPREAD=xxx**

Indicates whether each line item (column) of a report is separated as far as possible from adjacent line items. Valid values are YES and NO. Override is through the SPREAD parameter of the REPORT statement.

Default - NO  
Standard -

**SPRTXIT=modname**

Specifies the name of a user-supplied SYSPRINT/SYSLST exit routine. The modname must be a valid program name. No override is available. More information on the SYSPRINT/SYSLST exit capability can be found in the CA-Easytrieve/Plus Installation Guide, Appendix F, “Unit Record Exits.”

Default - none  
Standard -

**SQLBIND=xxxxxx**

Specifies the type of SQL bind processing for the execution of CA-Easytrieve/Plus DB2 programs. The default value is blank, which means that the value for SQLBIND is determined from the BIND parameter of the PARM statement of the CA-Easytrieve/Plus program. Valid values are:

- **DYNAMIC**—indicates that SQL statements are to be executed “dynamically.” The rules of DB2 Dynamic Bind apply to authorization checking. Refer to the explanation of BIND on the PARM statement in the this guide. No override is available.

- **STATIC**—indicates that SQL statements are to be executed “statically.” STATIC BIND results in the generation of a “static-command-program” which is preprocessed by the DB2 preprocessor and a DB2 Plan is bound into the DB2 system catalog. No override is available.

- **ANY**—indicates that STATIC processing is to be attempted first. If STATIC processing is not able to take place due to an incomplete static environment, then processing switches to DYNAMIC. ANY can be overridden by the PARM statement.

Default - blank  
Standard -
**SQLSYNTAX=xxxxxxx**

Specifies the type of syntax checking to be performed on the SQL statements. Valid values are:

- **FULL** — SQL statements are fully syntax checked using the facilities of the underlying DBMS. FULL syntax checking results in the SQL statement undergoing a “dynamic prepare.”

- **PARTIAL** — SQL statements are checked for valid keywords, no connection is made to the DBMS unless an INCLUDE statement is coded for an SQL table. A value of PARTIAL does not permit the program to execute until it has undergone FULL syntax checking.

- **NONE** — NONE applies to the DB2 SQL Interface. A value of NONE results in PARTIAL syntax checking being performed. However, it requires an SQLBIND value of STATIC-ONLY. NONE enables your program to execute without FULL syntax checking being performed. This option permits your DB2 program to bypass “dynamic prepares” and authorization checking at compilation.

Default - FULL  
Standard -

**SSID=**

Specifies a valid connection ID for the given database management system. The valid formats are as follows:

- **DB2** — xxxx/yyyyyyyy where xxxx is a valid DB2 subsystem ID and yyyyyyyyy is a valid DB2 location ID. Neither value is required. Either one can be specified. If a location ID is specified, it must be preceded by a slash, such as SSID=/yyyyyyyy.

  If no SSID is specified, the DB2 subsystem is obtained from the DB2 default module DSNHDECP found through the DB2 library, specified in your JOBLIB or STEPLIB JCL.

- **SQL/DS** — yyyyyyyyy where yyyyyyyyy is a valid database ID.

Override is through the SSID parameters of the PARM statement.

Default - blank  
Standard -

**STATE=xxxxxxx**

Specifies whether to maintain the statement number of the last statement executed. This option or FLOW is necessary if the statement number prefix is desired on execution-time diagnostic messages. Valid values are STATE and NOSTATE. Override is through the DEBUG parameter of the PARM statement.

Default - STATE  
Standard -
STORMAX=nnnnK

Specifies the maximum amount of storage which CA-Easytrieve/Plus uses. The valid range is 60 to 4096. The amount used is the lower of the STORMAX value or the actual available free space. No override is available.

Default - 1024K for VSE Standard -
Default - 4096K for OS/390 Standard -

SUMCTL=xxxx

Specifies the technique of annotating summary lines in a control report. For specific information, refer to the SUMCTL parameter of the REPORT statement in Chapter 11, “Report Processing,” of this guide. Valid values are ALL, HIAR, TAG, and NONE. Override is through the SUMCTL parameter of the REPORT statement.

Default - HIAR Standard -

SUMSPAC=n

Specifies the number of additional print positions to be reserved for printing summed fields in a report. The additional space is used to prevent an overflow condition when the summed field exceeds its defined size. The valid range is 0 to 9. Override is through the SUMSPACE parameter of the REPORT statement.

Default - 3 Standard -

SYSTEM=xxxxxx

Specifies the host operating system under which the CA-Easytrieve/Plus system will be running. Valid options are DOSVSE or OS. DOSVSE refers to version AF2 or higher.

Default - DOSVSE for VSE Standard -
Default - OS for OS/390 Standard -

TALYSIZ=nn

Specifies the size of the TALLY field in digits. The valid range is 1 to 18. The value of SUMSPAC is added to TALYSIZ to determine the effective size of the printed field. Using the defaults for SUMSPAC and TALYSIZ, the effective size of TALLY is five digits. Override is through the TALLYSIZE parameter of the REPORT statement.

Default - 2 Standard -
**TBLMAX=nnnnn**

Specifies the maximum number of entries for a table which is loaded from an external file (not instream). The valid range is 0 to 32767. Override is through the TABLE parameter of the FILE statement. If the value that you specify for TBLMAX is excessively high, then the total amount of storage required for CA-Easytrieve/Plus is inflated. If the value that you specify for TBLMAX is too low, then CA-Easytrieve/Plus issues a diagnostic message as the file is loaded. Specify a value that is adequate for 90 percent to 95 percent of the tables, then use the FILE statement override for particularly large tables.

Default - 256  
Standard -

**TITLSKP=nnn**

Specifies the number of blank lines to be inserted between the last title line and the first heading line (or the first data line if NOHEADING is specified). The valid range is 0 to 255 or PAGESIZ, whichever is smaller. Override is through the TITLESKIP parameter of the REPORT statement.

Default - 3  
Standard -

**UPDTDLI=xxx**

Specifies whether the DL/I update function codes DLET, ISRT, or REPL are to be allowed on the DL/I statement. Valid options are YES or NO. No override is available.

Default - NO  
Standard -

**UPDTIDD=xxx**

Specifies whether the dictionary is to be updated with program compilation statistics. Valid options are YES or NO. When set to YES, override is through the RETRIEVE parameter on the IDD NAME statement. When set to NO, no override is available.

Default - NO  
Standard -

**UPDTIDM=xxx**

Specifies whether the CA-IDMS update functions CONNECT, DISCONNECT, ERASE, MODIFY, READY (for UPDATE), or STORE are permitted on the CA-IDMS statement. Valid options are YES or NO. No override is available.

Default - NO  
Standard -
UPDTVS=xxx

Specifies whether the VSAM update functions are to be permitted. Valid values are YES or NO. If your site has a security product in place, set this option to YES. No override is available.

Default - NO Standard -

USERMSK=(id,'mask',...)

Specifies a list of installation defined edit masks. USERMSK provides the ability to predefine commonly used edit masks. Each mask is defined as a pair of values; the first value is the mask identifier, and the second value describes the mask just as it would be defined in an CA-Easytrieve/Plus field definition. Valid mask identifiers are A through Y. Using letters at the end of the alphabet avoids conflicts with programmer coded masks. Enclose the mask in single quotes as follows:


Default - none Standard -

VERFILE= {HIGHEST} {LOWEST} {nnnn}

(IDD only) Specifies the version of the non-database file that you want to retrieve. Valid values are HIGHEST, LOWEST, and a specific version number nnnn. Override is through the VERSION parameter of the IDD FILE statement.

Default - HIGHEST Standard -

VERPGM = {HIGHEST} {LOWEST} {nnnn}

(IDD only) Specifies the version of the program that is accessing the dictionary. Valid values are HIGHEST, LOWEST, and a specific version number nnnn. Override is through the VERSION parameter of the IDD NAME statement.

Default - HIGHEST Standard -

VERREC= {HIGHEST} {LOWEST} {nnnn}

(IDD only) Specifies the version of the record that you want to retrieve. Valid values are HIGHEST, LOWEST, and a specific version number nnnn. Override is through the VERSION parameter of the IDD RECORD statement.

Default - HIGHEST Standard -
VERSCHM= \{HIGHEST\} \{LOWEST\} \{nnnn\}  

(IDD only) Specifies the version of the schema owning the subschema that you want to retrieve. Valid values are HIGHEST, LOWEST, and a specific version number nnnn. Override is through the VERSION parameter of the IDD SUBSCHEMA statement.

Default - HIGHEST       Standard -  

VFMDEV=xxxxxxx  

Specifies the device type of the VFM overflow file. Valid values are 3330, 3340, 3350, 3375, 3380, 3390, FBA, DISK, MEMORY, or SAMV. DISK uses the device-type defined in the DISK parameter. Specify SAMV if the virtual file manager is on a VSAM managed SAM data space. Override is through the VFM parameter of the PARM statement. SAMV cannot be specified on the PARM statement.

When specifying SAMV, define the cluster as follows:

```
DEFINE CLUSTER
  (NAME(xxxxxx.xxxxx.xxxxx)  
  NONINDEXED
  REUSE
  NOALLOCATION
  RECORDFORMAT(V)
  RECORDSIZE(512 4096)
  CYL(n,n)
  VOLUME(volume))
  CATALOG(vsam.usercat.file)
```

Default - DISK       Standard -  

VFMSPAC=nnnnK  

Specifies the maximum amount of storage used by VFM for its buffer pool. The valid range is 6 to 4096. Specify a value that is adequate for 90 percent to 95 percent of the programs. Use the following formula:

\[(number \text{ of virtual files } + 1) \times 8K\]

Override is through the VFM parameter of the PARM statement.

Default - 64K       Standard -  

WKDSNPF=xxx  

Specifies the CA-Easytrieve/Plus three-character prefix used for all internal work files. A user file name cannot begin with the specified prefix. The value must be three characters that are valid as a file-name prefix. No override is available.

Default - EZT       Standard -
**XREF=xxxxxx**

Indicates whether a cross-reference listing is produced. Valid entries are LONG, SHORT, and NOXREF. LONG requests a cross-reference listing which includes unreferenced names. SHORT requests a listing that contains only referenced names. NOXREF inhibits production of a cross-reference listing. Override is through the DEBUG parameter of the PARM statement.

Default - NOXREF  
Standard -
Introduction

Appendix D illustrates how to use CA-Easytrieve/Plus to solve a variety of basic data processing problems. The emphasis is placed on reading data files and printing reports. The examples are excerpts from the CA-Easytrieve/Plus Application Guide which contains additional examples.

The input data for these examples are the Inventory and Personnel sample files described on the following pages. The field definitions for the files are contained in the macros listed. The field definitions are not repeated for each example; you can refer to the original field definitions as required.

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

Coding Conventions

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 conventions help make the programs more readable. Your choice of similar guidelines in the development of your programs permit the logic and structure of the programs to be easily discerned. A liberal supply of meaningful comments can also make program maintenance much easier.
Inventory Sample File

The macro listing $\%INVMSTR$ below provides the field definitions for the Inventory file in the following examples. Refer to this page when studying examples which use the Inventory file.

MACRO

* INVENTORY MASTER FIELD DEFINITIONS

* PART-INFO               1  43 A
  PART-DESCRIPTION               1 35 A -
  PART-NUMBER                   36  8 N  MASK '999-99-999' -

* 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 -
  ITEM-REORDER-POINT            66  4 N 0 -
  ITEM-LAST-SALE-DATE           70  6 N  MASK(D '99/99/99') -
  ITEM-LAST-INVENTORY-DATE      76  6 N  MASK D -
  ITEM-LAST-INVENTORY-QUANTITY  82  4 N 0 -
  ITEM-MFGD-COMMODITY-GROUP     86  3 P -
  ITEM-WEIGHT-POUNDS            89  2 P 0 MASK 'ZZ9 #' -

* LAST-PURCHASE-INFO     91 13 A
  LAST-PURCHASE-QUANTITY        91  3 P 0 -
  LAST-PURCHASE-PRICE           94  4 P 2 -
  LAST-PURCHASE-DATE            98  6 N  MASK D -

* VENDOR-INFO           104  17 A
  VENDOR-NUMBER                104  8 N  MASK '99-99-999' -
  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 JOB below lists the contents of the Personnel sample file. The field definitions are embedded as macro &PERSNL. These field definitions are not repeated in the examples. Refer to this section when studying an example which uses the Personnel file.

```
1 *
2 * PERSONNEL MASTER FILE LISTING
3 *
4 FILE PERSNL FB(150 1800)
5 %PERSNL
6 *
7 * TEST FILE FIELD DEFINITIONS
8 *
9 REGION 1 1 N
10 BRANCH 2 2 N
11 SSN 4 5 P MASK '999-99-9999' - HEADING('SOCIAL' 'SECURITY' 'NUMBER')
12 EMP# 9 5 N HEADING('EMPLOYEE' 'NUMBER')
13 NAME 17 20 A HEADING('EMPLOYEE NAME')
14 NAME-LAST 8 A HEADING('LAST NAME')
15 NAME-FIRST +8 12 A HEADING('FIRST NAME')
16 ADDRESS 37 39 A
17 ADDR-STREET 37 20 A HEADING('STREET')
18 ADDR-CITY 57 12 A HEADING('CITY')
19 ADDR-STATE 69 2 A HEADING('STATE')
20 ADDR-ZIP 71 5 N HEADING('ZIP CODE')
21 PAY-NET 90 4 P 2 HEADING('NET PAY')
22 PAY-GROSS 94 4 P 2 HEADING('GROSS PAY')
23 DEPT 98 3 N
24 DATE-OF-BIRTH 103 6 N MASK(Y 'Z9/99/99') - HEADING('DATE OF BIRTH')
25 TELEPHONE 117 10 N MASK '(999) 999-9999' - HEADING('TELEPHONE NUMBER')
26 SEX 127 1 N HEADING('SEX CODE')
27 * 1 - FEMALE
28 * 2 - MALE
29 MARITAL-STAT 128 1 A HEADING('MARITAL STATUS')
30 * M - MARRIED
31 * S - SINGLE
32 JOB-CATEGORY 132 2 N HEADING('JOB CATEGORY')
33 SALARY-CODE 134 2 N HEADING('SALARY CODE')
34 DATE-OF-HIRE 136 6 N MASK Y - HEADING('DATE OF HIRE')
35 *
36 JOB INPUT PERSNL
37 PRINT PERSNL-LIST
38 *
39 *
40 REPORT PERSNL-LIST SKIP 1 SPACE 1 LINESIZE 80
41 *
42 TITLE 'NEW PERSONNEL SAMPLE FILE LISTING'
43 *
44 HEADING REGION ('R' 'G' 'N')
45 HEADING BRANCH ('BRCH')
46 HEADING EMP# ('EMPL NUMBER')
47 HEADING SSN ('SOCIAL SECURITY NUMBER/' 'TELEPHONE')
48 HEADING PAY-GROSS ('PAY - ' 'GROSS/' 'NET')
49 HEADING SEX ('SEX/' 'M/S')
50 HEADING DEPT ('DPT/' 'J*C/' 'S*C')
51 HEADING DATE-OF-BIRTH ('DATE OF ' 'BIRTH/' 'HIRE')
52 *
```
<table>
<thead>
<tr>
<th>Line</th>
<th>POS</th>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>53</td>
<td>1</td>
<td>REGION</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BRANCH</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSN</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EMP#</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NAME</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>+2</td>
<td>PAY-GROSS</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEPT</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATE-OF-BIRTH</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SEX</td>
<td>-</td>
</tr>
<tr>
<td>54</td>
<td>-1</td>
<td>TELEPHONE</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 5</td>
<td>ADDR-STREET</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 6</td>
<td>PAY-NET</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 7</td>
<td>JOB-CATEGORY</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 8</td>
<td>DATE-OF-HIRE</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 9</td>
<td>MARITAL-STAT</td>
<td>-</td>
</tr>
<tr>
<td>55</td>
<td>POS 5</td>
<td>ADDR-CITY</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>ADDR-STATE</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADDR-ZIP</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>POS 7</td>
<td>SALARY-CODE</td>
<td>-</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/ GROSS/</td>
<td>DATE OF BIRTH/ SEX/</td>
<td>EMPL NUMBER/ EMPLOYEE NAME</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------</td>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>1 01</td>
<td>025-30-5228 12267 WIMN GLORIA 430 M ST SW 107 BOSTON MA 02005</td>
<td>373.60 903 5/22/30 1</td>
<td>(617) 332-2762</td>
</tr>
<tr>
<td>1 02</td>
<td>121-16-6413 11473 BERG NANCY 3710 JENIFER ST N W BALTIMORE MD 21055</td>
<td>759.20 943 8/15/31 1</td>
<td>(301) 636-8995</td>
</tr>
<tr>
<td>1 03</td>
<td>228-58-8307 02688 CORNING GEORGE 3208 S 5TH TRENTON NJ 08621</td>
<td>146.16 915 10/12/52 2</td>
<td>(609) 444-7688</td>
</tr>
<tr>
<td>1 02</td>
<td>256-52-8737 00370 NAGLE MARY 826 D STREET SE BALTIMORE MD 21834</td>
<td>554.40 935 1/13/43 1</td>
<td>(301) 636-8995</td>
</tr>
<tr>
<td>1 04</td>
<td>281-36-2873 01963 ARNOLD LINDA 1569 COLONIAL TERR NEW YORK NY 10012</td>
<td>445.50 911 8/29/42 1</td>
<td>(212) 451-4840</td>
</tr>
<tr>
<td>1 03</td>
<td>298-34-4755 11602 MANHART VIRGINIA 1305 POTOMAC ST N W TRENTON NJ 08621</td>
<td>344.80 914 10/13/38 1</td>
<td>(609) 444-3094</td>
</tr>
<tr>
<td>1 04</td>
<td>322-30-0050 11931 TALL ELAINE 1412 36TH ST NW NEW YORK NY 10091</td>
<td>492.26 917 12/25/40 1</td>
<td>(212) 451-4531</td>
</tr>
<tr>
<td>1 01</td>
<td>452-52-1419 02200 BRANDOW LYDIA 3616 B ST S E BOSTON MA 02111</td>
<td>804.64 918 9/14/47 1</td>
<td>(617) 332-6701</td>
</tr>
<tr>
<td>1 04</td>
<td>554-70-3189 11357 LARSON RODNEY 610 H ST SW NEW YORK NY 10059</td>
<td>283.92 911 2/11/39 2</td>
<td>(212) 451-7382</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY</td>
<td>PAY -</td>
<td>DPT/</td>
</tr>
<tr>
<td>---</td>
<td>------------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>G</td>
<td>NUMBER/</td>
<td>EMPL</td>
<td>GROSS/</td>
</tr>
<tr>
<td>N</td>
<td>BRCH</td>
<td>TELEPHONE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>1</td>
<td>04</td>
<td>577-20-0461</td>
<td>11467</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(212) 773-0799</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>01</td>
<td>579-12-0813</td>
<td>11376</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(813) 796-1189</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>579-50-4818</td>
<td>11710</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(404) 832-8801</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>03</td>
<td>008-28-7725</td>
<td>04234</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(202) 715-0404</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>03</td>
<td>120-32-5734</td>
<td>03416</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(202) 715-0389</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>04</td>
<td>190-32-2101</td>
<td>00445</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(904) 986-0034</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>03</td>
<td>212-48-5461</td>
<td>00577</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(202) 715-1914</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY G NUMBER/</td>
<td>EMPLOYER NUMBER</td>
<td>EMPLOYEE NAME</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------</td>
<td>-----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>2 05</td>
<td>235-72-1049</td>
<td>(919) 489-1614</td>
<td>VETTER DENISE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7311 KEYSTONE LANE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RALEIGH NC 27591</td>
</tr>
<tr>
<td>2 02</td>
<td>284-36-5652</td>
<td>(404) 832-1776</td>
<td>KRUSE MAX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2161 N PIERCE STREET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ATLANTA GA 30345</td>
</tr>
<tr>
<td>2 05</td>
<td>310-44-5370</td>
<td>(919) 489-5531</td>
<td>LOYAL NED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>17 KENNEDY STREET</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RALEIGH NC 27516</td>
</tr>
<tr>
<td>2 03</td>
<td>362-48-0393</td>
<td>(202) 715-1832</td>
<td>DENNING RALPH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1629 16TH ST NW APT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WASHINGTON DC 20005</td>
</tr>
<tr>
<td>3 01</td>
<td>570-10-5594</td>
<td>(816) 581-1352</td>
<td>WEST KATHY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1728 IRVING ST N W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KANSAS CITY KS 66015</td>
</tr>
<tr>
<td>3 02</td>
<td>577-09-1160</td>
<td>(214) 941-1441</td>
<td>THOMPSON JANICE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7752 EMERSON RD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DALLAS TX 75235</td>
</tr>
<tr>
<td>3 02</td>
<td>578-38-7587</td>
<td>(214) 941-1585</td>
<td>SMOTH CINDY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4128 18TH STREET NE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DALLAS TX 75219</td>
</tr>
<tr>
<td>3 03</td>
<td>578-54-3178</td>
<td>(312) 646-0934</td>
<td>NORIDGE DEBBIE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4264 E CAPITOL NE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CHICAGO IL 60652</td>
</tr>
<tr>
<td>R</td>
<td>SOCIAL SECURITY NUMBER/</td>
<td>EMPLOYEE NAME</td>
<td>GROSS/</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------</td>
<td>----------------</td>
<td>--------</td>
</tr>
<tr>
<td>3</td>
<td>03 579-50-4170 (312) 646-1650</td>
<td>ROGERS PAT</td>
<td>329.00</td>
</tr>
<tr>
<td></td>
<td>03 208-28-2315 (612) 588-1900</td>
<td>GREEN BRENDA</td>
<td>365.60</td>
</tr>
<tr>
<td></td>
<td>01 231-68-9995 (816) 581-0031</td>
<td>KELLY KEITH</td>
<td>197.60</td>
</tr>
<tr>
<td></td>
<td>02 418-46-1872 (214) 941-0558</td>
<td>ISAAC RUTH</td>
<td>313.60</td>
</tr>
<tr>
<td></td>
<td>03 467-56-4149 (312) 646-1891</td>
<td>STRIDE ANN</td>
<td>386.40</td>
</tr>
<tr>
<td></td>
<td>02 215-36-5852 (214) 986-1901</td>
<td>LACH LORRIE</td>
<td>310.40</td>
</tr>
<tr>
<td></td>
<td>03 228-46-5157 (312) 588-5118</td>
<td>EPERT LINDA</td>
<td>310.40</td>
</tr>
</tbody>
</table>

D–8 CA-Easytrieve/Plus Reference Guide
<table>
<thead>
<tr>
<th>R</th>
<th>SOCIAL SECURITY</th>
<th>G</th>
<th>NUMBER/</th>
<th>EMPL</th>
<th>PAY -</th>
<th>DPT/</th>
<th>DATE OF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>BRCH</td>
<td>TELEPHONE NUMBER</td>
<td>EMPL</td>
<td>GROSS/</td>
<td>J*C/</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>09481</td>
<td>OSMON</td>
<td>269-24-7428</td>
<td>09481</td>
<td>4201 CATHEDRAL AVE</td>
<td>628.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAMUEL</td>
<td>(312) 588-8995</td>
<td></td>
<td>20</td>
<td>7/22/50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>07231</td>
<td>GRECO</td>
<td>388-18-6119</td>
<td>07231</td>
<td>2195 CANTERBURY WAY</td>
<td>1,064.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LESLIE</td>
<td>(214) 399-7608</td>
<td></td>
<td>685.23</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>08262</td>
<td>CROCI</td>
<td>577-16-2985</td>
<td>08262</td>
<td>1666 C ST NE</td>
<td>376.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JUDY</td>
<td>(553) 444-1970</td>
<td></td>
<td>60</td>
<td>9/23/64</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>05805</td>
<td>REYNOLDS</td>
<td>061-30-8680</td>
<td>05805</td>
<td>4126 CROSSWICK TURN</td>
<td>174.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WILLIAM</td>
<td>(214) 399-4040</td>
<td></td>
<td>134.03</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>05807</td>
<td>PHILPS</td>
<td>090-22-9192</td>
<td>05807</td>
<td>2954 NORTHAMPTON ST</td>
<td>253.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SUE</td>
<td>(816) 836-3094</td>
<td></td>
<td>213.76</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>04589</td>
<td>YOUNG</td>
<td>118-34-8805</td>
<td>04589</td>
<td>1545 18 STREET NW</td>
<td>313.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JANE</td>
<td>(816) 836-4531</td>
<td></td>
<td>229.69</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>05914</td>
<td>MILLER</td>
<td>140-32-0779</td>
<td>05914</td>
<td>3 POOKS HILL RD APT</td>
<td>312.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>JOAN</td>
<td>(312) 588-6791</td>
<td></td>
<td>222.61</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>05525</td>
<td>TALUS</td>
<td>216-44-7756</td>
<td>05525</td>
<td>9331 CAROLINE AVENUE</td>
<td>460.80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RUTH</td>
<td>(206) 225-3828</td>
<td></td>
<td>279.56</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Examples**
<table>
<thead>
<tr>
<th>R</th>
<th>SOCIAL SECURITY NUMBER</th>
<th>EMPLOYEE NAME</th>
<th>GROSS</th>
<th>NET</th>
<th>J-C</th>
<th>BIRTH</th>
<th>SEX</th>
<th>DPT/ DATE OF HIRE</th>
<th>BRCH</th>
<th>TELEPHONE NUMBER</th>
<th>EMPLOYEE TELEPHONE NUMBER</th>
<th>EMPLOYEE ADDRESS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4 02</td>
<td>484-30-8293 (714) 771-0799</td>
<td>JOHNSON LISA</td>
<td>712.80</td>
<td>451.92</td>
<td>942</td>
<td>2/05/46</td>
<td>1</td>
<td>02</td>
<td>484-30-8293</td>
<td>(714) 771-0799</td>
<td>886 CONNECTICUT AVE SAN DIEGO CA 92045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 01</td>
<td>577-58-0363 (206) 225-9127</td>
<td>WARD MARINA</td>
<td>183.75</td>
<td>141.47</td>
<td>921</td>
<td>9/12/23</td>
<td>1</td>
<td>01</td>
<td>577-58-0363</td>
<td>(206) 225-9127</td>
<td>1725 H ST NE APT 2 SEATTLE WA 98101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 03</td>
<td>579-62-1768 (213) 493-5966</td>
<td>ZOLTAN JANET</td>
<td>125.00</td>
<td>25.00</td>
<td>924</td>
<td>9/14/68</td>
<td>S</td>
<td>03</td>
<td>579-62-1768</td>
<td>(213) 493-5966</td>
<td>2026 FORT DAVIS ST LOS ANGELES CA 90091</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 03</td>
<td>060-26-8978 (213) 493-8979</td>
<td>JONES ALFRED</td>
<td>804.80</td>
<td>560.63</td>
<td>940</td>
<td>11/13/39</td>
<td>2</td>
<td>03</td>
<td>060-26-8978</td>
<td>(213) 493-8979</td>
<td>2870 BELMONT ROAD N LOS ANGELES CA 90052</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 02</td>
<td>104-20-0956 (714) 771-9876</td>
<td>HAFER ARTHUR</td>
<td>121.95</td>
<td>96.64</td>
<td>911</td>
<td>8/03/40</td>
<td>2</td>
<td>02</td>
<td>104-20-0956</td>
<td>(714) 771-9876</td>
<td>806 CONNECTICUT AVE SAN DIEGO CA 92031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 04</td>
<td>448-24-6593 (415) 278-1753</td>
<td>JUDAR PAULA</td>
<td>591.20</td>
<td>459.57</td>
<td>943</td>
<td>10/12/50</td>
<td>1</td>
<td>04</td>
<td>448-24-6593</td>
<td>(415) 278-1753</td>
<td>433 46TH ST N W SAN FRANCISCO CA 94041</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 03</td>
<td>537-03-4093 (213) 725-6495</td>
<td>WALTERS KAREN</td>
<td>424.00</td>
<td>282.45</td>
<td>901</td>
<td>3/17/71</td>
<td>M</td>
<td>03</td>
<td>537-03-4093</td>
<td>(213) 725-6495</td>
<td>1022 S KENSINGTON P LOS ANGELES CA 90030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 01</td>
<td>558-44-7609 (206) 225-8456</td>
<td>RYAN PAMELA</td>
<td>399.20</td>
<td>291.70</td>
<td>914</td>
<td>12/12/49</td>
<td>1</td>
<td>01</td>
<td>558-44-7609</td>
<td>(206) 225-8456</td>
<td>1717 R N W #301 SEATTLE WA 98109</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GETDATE Macro

The macro %GETDATE, listed below, obtains the system date and strips out the slashes. The macro is defined with a single positional parameter -- the receiving field. All other required fields are defined within the macro.

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-LAST75 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-FIRST 6       . * MOVE TO USER FIELD
Basic Examples

Example 1 - Employees in Region 1

The Personnel Department 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 via 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 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

MM/DD/YY               EMPLOYEES IN REGION 1          PAGE      1

<table>
<thead>
<tr>
<th>FIRST NAME</th>
<th>LAST NAME</th>
<th>EMPLOYEE NUMBER</th>
<th>BRANCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLORIA</td>
<td>WIMN</td>
<td>12267</td>
<td>01</td>
</tr>
<tr>
<td>NANCY</td>
<td>BERG</td>
<td>11473</td>
<td>02</td>
</tr>
<tr>
<td>GEORGE</td>
<td>CORNING</td>
<td>02688</td>
<td>03</td>
</tr>
<tr>
<td>MARY</td>
<td>NAGLE</td>
<td>00370</td>
<td>02</td>
</tr>
<tr>
<td>LINDA</td>
<td>ARNOLD</td>
<td>01963</td>
<td>04</td>
</tr>
<tr>
<td>VIRGINIA</td>
<td>MANHART</td>
<td>11602</td>
<td>03</td>
</tr>
<tr>
<td>ELAINE</td>
<td>TALL</td>
<td>11931</td>
<td>04</td>
</tr>
<tr>
<td>LYDIA</td>
<td>BRANDOW</td>
<td>02200</td>
<td>01</td>
</tr>
<tr>
<td>RODNEY</td>
<td>LARSON</td>
<td>11357</td>
<td>04</td>
</tr>
<tr>
<td>JULIE</td>
<td>BYER</td>
<td>11467</td>
<td>04</td>
</tr>
</tbody>
</table>
```
Example 2 - Proposed Salary Schedules

The Personnel Department 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:** The use of the BEFORE-BREAK procedure 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 *  2 *   EXAMPLE 2 3 *
4 FILE PERSNL FB(150 1800) 5 %PERSNL
```
Control Branch

Title 1 'PROPOSED SALARY SCHEDULE FOR REGION 4 EMPLOYEES'

Title 2 'DETAIL BY BRANCH -- DESCENDING PAY-GROSS'

Line 1 Branch name=last pay-gross raise-dollars raise-percent - new-salary

Before-break. proc

RAISE-PERCENT = RAISE-PERCENT / TALLY + .005

End-proc

Report summary-by-category summary linesize 78

Sequence branch job-category

Control branch job-category

Title 1 'PROPOSED SALARY SCHEDULE FOR REGION 4'

Title 2 'SUMMARY BY JOB-CATEGORY AND BRANCH'

Line 1 Branch job-category pay-gross new-salary raise-dollars

Proposed salary schedule for Region 4 employees page 1

Detail by branch -- descending pay-gross

<table>
<thead>
<tr>
<th>BRANCH</th>
<th>LAST NAME</th>
<th>GROSS</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></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>JOHNSON</td>
<td>712.80</td>
<td>49.90</td>
<td>7.00</td>
<td>762.70</td>
</tr>
<tr>
<td></td>
<td>HAFER</td>
<td>121.95</td>
<td>10.98</td>
<td>9.00</td>
<td>132.93</td>
</tr>
<tr>
<td>02</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>

Proposed salary schedule for Region 4 employees page 1

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>60.88</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>
Example 3 - Employee Letters

The Personnel Department decided to accept the proposed salary adjustments and wishes to generate letters to all employees informing them of the salary adjustment. In addition to the letter, a mailing label must be generated. Order letters and mailing labels 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. Notice the ease with which the letter is specified. By including the parameters SKIP 1 and PAGESIZE 40, we ensure only one letter per page. The mailing labels are generated by specifying their content. The ACROSS 2 parameter allows 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.

```
1 *
2 *   EXAMPLE 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)') -
   MASK S
38 NEW-SALARY     W 4 P 2   HEADING('PROPOSED' 'SALARY') -
   MASK S
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-IF
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 -
   NOHEADING NOADJUST  SPACE 1  PAGESIZE 40  SKIP 1
62 SEQUENCE  ADDR-ZIP
63 LINE   1  COL 1 'ABC SYSTEMS, INC.'  COL 60 SYSDATE
64 LINE   3  NAME-FIRST NAME-LAST
65 LINE   4  ADDR-STREET
66 LINE   5  ADDR-CITY ADDR-STATE ADDR-ZIP
67 LINE   7  'DEAR' NAME-FIRST
68 LINE   9  'IT IS WITH GREAT PLEASURE THAT ABC SYSTEMS IS'
69 LINE  10  'PROVIDING YOU A SALARY INCREASE EFFECTIVE ON'
70 LINE  11  'YOUR NEXT PAY CHECK. THE INCREASE REFLECTS YOUR'
71 LINE  12  'EFFORTS IN MAKING ABC SYSTEMS THE LEADER IN THE'
72 LINE  13  'FIELD OF FINANCIAL COMPUTER SYSTEMS.'
73 LINE  15  'IN YOUR PARTICULAR CASE THE INCREASE IS' RAISE-PERCENT -
```
'%' 
74 LINE 16 'OF YOUR GROSS SALARY OF' OLD-SALARY '.  THIS EQUATES' 
75 LINE 17 'TO' RAISE-DOLLARS ', OR A NEW GROSS SALARY OF' - 
76 LINE 19 'NEW-SALARY '.  
77 LINE 20 'YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.' 
78 LINE 23 'SINCERELY,' 
79 LINE 27 'FRANK K. WILLIAMS' 
80 LINE 28 'PRESIDENT' 
81 * 
82 REPORT MAILING-LABEL LABELS ACROSS 2 
83 SEQUENCE ADDR-ZIP 
84 LINE 1 NAME-FIRST NAME-LAST 
85 LINE 2 ADDR-STREET 
86 LINE 3 ADDR-CITY ADDR-STATE ADDR-ZIP 

ABC SYSTEMS, INC.                      MM/DD/YY 
KAREN WALTERS 
1022 S KENSINGTON PK 
LOS ANGELES CA 90030 

DEAR KAREN 

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.                                           MM/DD/YY

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.                                           MM/DD/YY

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
KAREN    WALTERS  
1022 S KENSINGTON PK  
LOS ANGELES    CA   90030

ALFRED     JONES  
2670 BELMONT ROAD NW  
LOS ANGELES    CA   90052

JANET      ZOLTAN 
2026 FORT DAVIS ST S 
LOS ANGELES    CA   90091

ARTHUR     HAFER  
806 CONNECTICUT AVE 
SAN DIEGO      CA   92031

LISA       JOHNSON  
806 CONNECTICUT AVE 
SAN DIEGO      CA   92045

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

RUTH       TALUS  
9331 CAROLINE AVENUE 
SEATTLE        WA   98003

PAMELA     RYAN  
1717 R N W #301 
SEATTLE        WA   98009

MARINA     WARD  
1725 H ST NE APT 2 
SEATTLE        WA   98105
Example 4 - 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), 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. Notice the redefinition of the ZIP code field to allow 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 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
```
Example 5 - Tally Reports

The Personnel Department wants tallies on various fields within the personnel file. Each tally report is to list 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 that 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 * EXAMPLE 5
2 * FILE PERSNL FB(150 1800)
3 %PERSNL
4   SEX-CODE W 6 A HEADING 'SEX'
5   GROSS-RANGE W 3 P HEADING ('SALARY RANGE' 'HUNDRED $ INCR')
6   TOTAL-EMPLOYEES S 3 P 0
7   PERCENT W 3 P 2 HEADING('PERCENT' 'OF' 'TOTAL')
8 *
9 JOB
10   TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1
11 *
12   IF SEX EQ 1                      . * SET PROPER SEX CODE
13     SEX-CODE = 'FEMALE'
14   ELSE
15     SEX-CODE = 'MALE'
16   END-IF
17 *
18 PRINT SEX-MARITAL-STAT-RPT       . * PRINT REPORT
19 *
20 PRINT JOB-CATEGORY-RPT            . * PRINT REPORT
21 *
22 PRINT SALARY-CODE-RPT             . * PRINT REPORT
23 *
24 *
25 *
26 *
27 *
28 *
29 *
30 *
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
```
Basic Examples

59  GROSS-RANGE = PAY-GROSS / 100.00  . * CALCULATE GROSS SALARY
61  GROSS-RANGE = GROSS-RANGE * 100  . * RANGE
63  PRINT GROSS-PAY-RPT  . * PRINT THE REPORT
65  *
66  PRINT CITY-RPT  . * PRINT THE CITY REPORT
68  *
69  REPORT SEX-MARITAL-STAT-RPT SUMMARY LINESIZE 78
70  SEQUENCE SEX-CODE MARITAL-STAT  . * SORT REPORT
72  CONTROL SEX-CODE MARITAL-STAT  . * BREAK SPECIFICATION
74  TITLE 1 'TALLY OF EMPLOYEES BY SEX AND MARITAL STATUS'
75  LINE 1 SEX-CODE MARITAL-STAT TALLY PERCENT
76  BEFORE-BREAK. PROC  . * CALCULATE PERCENT
79  PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
80  END-PROC
81  *
82  REPORT JOB-CATEGORY-RPT SUMMARY LINESIZE 78
83  SEQUENCE JOB-CATEGORY  . * SORT REPORT
85  CONTROL JOB-CATEGORY  . * BREAK SPECIFICATION
87  TITLE 1 'TALLY OF EMPLOYEES BY JOB CATEGORY'
88  LINE 1 JOB-CATEGORY TALLY PERCENT
92  PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
93  END-PROC
94  *
95  REPORT SALARY-CODE-RPT SUMMARY LINESIZE 78
96  SEQUENCE SALARY-CODE  . * SORT REPORT
98  CONTROL SALARY-CODE  . * BREAK SPECIFICATION
100 TITLE 1 'TALLY OF EMPLOYEES BY SALARY CODE'
101 LINE 1 SALARY-CODE TALLY PERCENT
105 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
106 END-PROC
107  *
108 REPORT GROSS-PAY-RPT SUMMARY LINESIZE 78
109 SEQUENCE GROSS-RANGE D  . * SORT REPORT
111 CONTROL GROSS-RANGE  . * BREAK SPECIFICATION
113 TITLE 1 'TALLY OF EMPLOYEES BY GROSS SALARY RANGE'
114 HEADING PAY-GROSS ('AVERAGE' 'GROSS' 'SALARY')
115 LINE 1 GROSS-RANGE TALLY PERCENT PAY-GROSS
119 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
120 PAY-GROSS = PAY-GROSS / TALLY + .005
121 END-PROC
122  *
123 REPORT CITY-RPT SUMMARY LINESIZE 78
124 SEQUENCE ADDR-CITY  . * SORT REPORT
126 CONTROL ADDR-CITY  . * BREAK SPECIFICATION
128 TITLE 1 'TALLY OF EMPLOYEES BY HOME CITY'
130 LINE 1 ADDR-CITY TALLY PERCENT
133 PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
134 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>
<tr>
<td></td>
<td></td>
<td>48</td>
<td>100.00</td>
</tr>
</tbody>
</table>

### 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>
<tr>
<td>48</td>
<td></td>
<td>100.00</td>
</tr>
</tbody>
</table>
### 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>

<table>
<thead>
<tr>
<th>SALARY CODE</th>
<th>TALLY</th>
<th>PERCENT OF TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td></td>
<td>100.00</td>
</tr>
</tbody>
</table>

### 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 HUNDRED $ INCR</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>800</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>700</td>
<td>3</td>
<td>6.25</td>
</tr>
<tr>
<td>600</td>
<td>1</td>
<td>2.08</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>4.17</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>8.33</td>
</tr>
<tr>
<td>300</td>
<td>17</td>
<td>35.42</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>20.83</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>14.58</td>
</tr>
<tr>
<td>1</td>
<td>2.08</td>
<td>13.80</td>
</tr>
<tr>
<td>48</td>
<td>100.00</td>
<td>376.63</td>
</tr>
<tr>
<td>CITY</td>
<td>TALLY</td>
<td>PERCENT OF TOTAL</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<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 FRANCISCO</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>
Example 6 - Phone Number Selection

The National Federation of Business and Professional Women's Clubs is recruiting for a chapter in the Chicago area. They 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 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  
-------------------------------------------------------------------------------  
MM/DD/YY CHICAGO AREA WOMEN AND TELEPHONE NUMBERS PAGE 1  
FIRST LAST TELEPHONE  
NAME NAME NUMBER  
LINDA EPERT (312) 588-5118  
JOAN MILLER (312) 588-6701  
DEBBIE NORIDGE (312) 646-0934  
PAT ROGERS (312) 646-1650  
ANN STRIDE (312) 646-1891  
```
Example 7 - Salary Tally Report

The Personnel Department requested that the Salary Range Report produced in Example 5 be expanded to include a bar chart 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 allows us to modify the contents of a summary line prior to printing (a common requirement in many control reports).

```
1 * 2 * EXAMPLE 7
3 * 4 FILE PERSNL FB(150 1800)
5 %PERSNL
35 GROSS-RANGE W 3 P 65 HEADING ('SALARY RANGE' 'HUNDRED $ INCR')
36 TOTAL-EMPLOYEES S 3 P 0
37 PERCENT W 3 P 2 65 HEADING ('PERCENT' 'OF' 'TOTAL')
38 BAR-GRAPH S 30 A 65 HEADING ('PERCENT OF EMPLOYEES' -
39 ASTERISKS S 30 A 65 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
```
<table>
<thead>
<tr>
<th>Salary Range</th>
<th>Tally</th>
<th>Percent of Employees</th>
<th>Percent of Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>1</td>
<td>2.08</td>
<td>*</td>
</tr>
<tr>
<td>800</td>
<td>2</td>
<td>4.17</td>
<td>**</td>
</tr>
<tr>
<td>700</td>
<td>3</td>
<td>6.25</td>
<td>***</td>
</tr>
<tr>
<td>600</td>
<td>1</td>
<td>2.08</td>
<td>*</td>
</tr>
<tr>
<td>500</td>
<td>2</td>
<td>4.17</td>
<td>**</td>
</tr>
<tr>
<td>400</td>
<td>4</td>
<td>8.33</td>
<td>****</td>
</tr>
<tr>
<td>300</td>
<td>17</td>
<td>35.42</td>
<td>*********************</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>20.83</td>
<td>**********</td>
</tr>
<tr>
<td>100</td>
<td>7</td>
<td>14.58</td>
<td>********</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.08</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>
Example 8 - File Expansion

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.

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, all five data moves are performed with one MOVE LIKE statement. The new fields are easily initialized with the MOVE statement.
1 *
2 *     EXAMPLE 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       90  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 *     JOB   FINISH  WRAP-UP
26     MOVE  LIKE  PERSIN TO PERSOUT    . * MOVE LIKE NAMED FIELDS
27     MOVE  ZERO  TO  NEW-1 NEW-2 NEW-3 NEW-4 NEW-7
28     MOVE  SPACE TO  NEW-5 NEW-6
29     PUT  PERSOUT                     . * OUTPUT THE REFORMATTED FILE
30 *
31 WRAP-UP.  PROC
32     DISPLAY NEWPAGE 'TOTAL INPUT RECORDS = ' RECORD-COUNT(PERSIN)
33     DISPLAY SKIP 2 'TOTAL OUTPUT RECORDS = ' RECORD-COUNT(PERSOUT)
34 END-PROC
35
36 TOTAL INPUT RECORDS =            48
37
38 TOTAL OUTPUT RECORDS =            48
39
40
**Example 9 - 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 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.

Notice that 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, notice that 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.
EXAMPLE 9

FILE PERSNL FB(150 1800)

* AVERAGE-GROSS W 4 P 2  HEADING('AVERAGE' 'GROSS' 'SALARY')
* TOTAL-GROSS S 6 P 2
* PERCENT-GROSS W 3 P 2  HEADING('PERCENT OF' 'COMPANY' 'GROSS')
* PERCENT-TALLY W 3 P 2  HEADING('PERCENT OF' 'COMPANY' 'EMPLOYEES')
* SALARY-RATIO W 3 P 3  HEADING('RATIO OF' '%-GROSS /' '%-TALLY')

TOTAL-EMPLOYEES S 3 P 0

REGION-TEXT W 10 A  HEADING('COMPANY' 'REGION')

FILE RGNID TABLE INSTREAM.

DEFINE INSTREAM REGION TABLE.

ARG 1 1 N.
DESC 3 10 A.

DEFINE TABLE SPECIAL FIELD IDS.

REGION.

1 NORTHEAST
2 SOUTHEAST
3 CENTRAL
4 WEST

ENDTABLE.

JOB.

TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1.  * CALCULATE TOTAL EMPLOYEES
AVERAGE-GROSS = PAY-GROSS  .  * AVERAGE = GROSS FOR EACH RECD
TOTAL-GROSS = TOTAL-GROSS + PAY-GROSS

TOTAL-GROSS FOR COMP.

SEARCH TABLE FOR MATCHING REGION INFORMATION.

SEARCH RGNID WITH REGION GIVING REGION-TEXT.

PRINT AVG-SALARY-RPT.  * PRINT THE REPORT

REPORT AVG-SALARY-RPT SUMMARY LINESIZE 78

SEQUENCE REGION  .  * SORT REPORT

CONTROL REGION-TEXT  .  * BREAK SPECIFICATION

TITLE 1 'AVERAGE GROSS SALARY BY REGION'

HEADING TALLY ('NUMBER' 'OF' 'EMPLOYEES')

LINE 1 REGION-TEXT TALLY PERCENT-TALLY -

AVERAGE-GROSS PERCENT-GROSS SALARY-RATIO

BEFORE-BREAK. PROC ... CALCULATE PERCENT

PERCENT-GROSS = AVERAGE-GROSS * 100 / TOTAL-GROSS + .005

AVERAGE-GROSS = AVERAGE-GROSS / TALLY + .005

PERCENT-TALLY = TALLY * 100 / TOTAL-EMPLOYEES + .005

SALARY-RATIO = PERCENT-GROSS / PERCENT-TALLY + .0005

END-PROC.

-------------------------------------------------------------------------------

MM/DD/YY AVERAGE GROSS SALARY BY REGION PAGE 1

<table>
<thead>
<tr>
<th>COMPANY REGION</th>
<th>NUMBER OF EMPLOYEES</th>
<th>PERCENT OF COMPANY EMPLOYEES</th>
<th>AVERAGE GROSS SALARY</th>
<th>PERCENT OF COMPANY GROSS</th>
<th>RATIO OF %-GROSS / %-TALLY</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORTHEAST</td>
<td>10</td>
<td>20.83</td>
<td>460.12</td>
<td>25.45</td>
<td>1.222</td>
</tr>
<tr>
<td>SOUTHEAST</td>
<td>10</td>
<td>20.83</td>
<td>246.98</td>
<td>13.66</td>
<td>.656</td>
</tr>
<tr>
<td>CENTRAL</td>
<td>19</td>
<td>39.58</td>
<td>378.08</td>
<td>39.74</td>
<td>1.004</td>
</tr>
<tr>
<td>WEST</td>
<td>9</td>
<td>18.75</td>
<td>424.83</td>
<td>21.15</td>
<td>1.128</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>100.00</td>
<td>376.63</td>
<td>100.00</td>
<td>1.000</td>
</tr>
</tbody>
</table>
Example 10 - 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, and Marketing. 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 of 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. Notice the use of the HEADING statement to supply alternate report headings for the specified fields; this is the only way to change the heading for TALLY. Also note that printing is suppressed for the summary line in the MISSING-DEPT-CODE report.
EXAMPLE 10

FILE PERSNL FB(150 1800)

DEPT-NAMES W 15 A HEADING ('DEPARTMENT' 'NAME')

FILE DPTCODE TABLE . * TABLE FILE DEFINITION

ARG 1 3 N DESC 5 15 A

JOB

IF REGION NE 3 . * SELECT REGION 3 EMPLOYEES
GO TO JOB . * SKIP ALL OTHERS
END-IF

SEARCH DPTCODE WITH DEPT GIVING DEPT-NAMES

IF NOT DPTCODE . * IF NO DEPT NAME PRESENT
DEPT-NAMES = '*NO TABLE ENTRY' . * INDICATE MISSING ENTRY
PRINT MISSING-DPT-CODE . * OUTPUT ERROR REPORT
END-IF

PRINT ALPHA-LIST . * PRINT ALPHA LISTING
PRINT RPT-BY-DEPT . * OUTPUT REPORT BY DEPARTMENT

REPORT ALPHA-LIST LINESIZE 78
SEQUENCE NAME-LAST NAME-FIRST
TITLE 'CENTRAL REGION EMPLOYEES'
LINE NAME-LAST NAME-FIRST SSN DEPT DEPT-NAMES

REPORT RPT-BY-DEPT SUMCTL NONE LINESIZE 78
SEQUENCE DEPT-NAMES NAME-LAST . * SEQUENCE BY DEPT AND NAME
CONTROL DEPT-NAMES . * CONTROL BREAK ON DEPT
TITLE 'CENTRAL REGION EMPLOYEES BY DEPARTMENT'
HEADING TALLY ('NUMBER' 'OF' 'EMPLOYEES')
LINE DEPT-NAMES BRANCH NAME-LAST NAME-LAST TALLY

REPORT MISSING-DPT-CODE SUMMARY LINESIZE 78
SEQUENCE DEPT . * SEQUENCE BY DEPT
CONTROL FINAL NOPRINT DEPT . * CONTROL BREAK ON DEPT
TITLE 'CENTRAL REGION MISSING DEPARTMENT DESCRIPTIONS'
HEADING TALLY ('NUMBER' 'OF' 'EMPLOYEES')
LINE DEPT TALLY
<table>
<thead>
<tr>
<th>LAST</th>
<th>FIRST</th>
<th>SOCIAL NUMBER</th>
<th>DEPT</th>
<th>DEPARTMENT</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>BRENDA</td>
<td>288-28-2315</td>
<td>911</td>
<td>DATA PROCESSING</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>MILLER</td>
<td>JOAN</td>
<td>149-38-0779</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>661-36-8689</td>
<td>911</td>
<td>MARKETING</td>
</tr>
<tr>
<td>ROGERS</td>
<td>PAT</td>
<td>579-38-4179</td>
<td>924</td>
<td>PERSONNEL</td>
</tr>
<tr>
<td>SMOTH</td>
<td>CINDY</td>
<td>778-38-7587</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>MM/DD/YY</td>
<td>CENTRAL REGION EMPLOYEES BY DEPARTMENT</td>
<td>PAGE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------------</td>
<td>------</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DEPARTMENT NAME</th>
<th>BRANCH</th>
<th>LAST NAME</th>
<th>FIRST NAME</th>
<th>NUMBER OF EMPLOYEES</th>
</tr>
</thead>
<tbody>
<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>1</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>2</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>SMOTH</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>6</td>
</tr>
<tr>
<td>PRINTING</td>
<td>01</td>
<td>KELLY</td>
<td>KEITH</td>
<td>1</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>3</td>
</tr>
<tr>
<td></td>
<td>03</td>
<td>OSMON</td>
<td>SAMUEL</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MM/DD/YY</th>
<th>CENTRAL REGION MISSING DEPARTMENT DESCRIPTIONS</th>
<th>PAGE</th>
<th>1</th>
</tr>
</thead>
</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>
Example 11 - 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 requested an inventory report, ordered by the city in which the parts are located. The groups by location 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.

```plaintext
1 *
2 *   EXAMPLE 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
```
<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>000-19-944</td>
<td>PANEL, SOLAR</td>
</tr>
<tr>
<td></td>
<td>000-53-190</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>000-70-750</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&quot; X 96&quot;)</td>
</tr>
<tr>
<td></td>
<td>000-60-680</td>
<td>BATTERIES, ELECTRIC DRY CELL</td>
</tr>
<tr>
<td></td>
<td>001-78-260</td>
<td>AIR BRAKES</td>
</tr>
<tr>
<td></td>
<td>001-79-080</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&quot; X 72&quot;)</td>
</tr>
<tr>
<td>MAMMOND</td>
<td>000-19-360</td>
<td>WALLBOARD, FIBERBOARD (48&quot; X 96&quot;)</td>
</tr>
<tr>
<td>MEMPHIS</td>
<td>001-84-260</td>
<td>BUMPERS</td>
</tr>
<tr>
<td></td>
<td>001-85-260</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>
Example 12 - Expanded Inventory Report

After reviewing the previous report, the Materials Department decided they want 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.

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.

```
1 * 2 *   EXAMPLE 12 3 * 4 FILE   INVMSTR   FB(200 3000) 5 %INVMSTR 44 * 45 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 * 51 ITEM-EXT-VALUE = ITEM-SELLING-PRICE * ITEM-LAST-INVENTORY-QUANTITY 52 TOTAL-EXT-VALUE = TOTAL-EXT-VALUE + ITEM-EXT-VALUE 53 * 54 PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE 55 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 - 63 ITEM-LAST-INVENTORY-QUANTITY ITEM-SELLING-PRICE - 64 ITEM-EXT-VALUE 66 REPORT   SMY-BY-CITY   SUMMARY    LINESIZE 80 67   SEQUENCE  LOCATION-CITY 68   CONTROL   LOCATION-CITY 69   TITLE   1 'INVENTORY VALUE SUMMARY BY CITY' 70   LINE   1 LOCATION-CITY ITEM-EXT-VALUE PERCENT 71 BEFORE-BREAK. PROC 72     PERCENT = ITEM-EXT-VALUE * 100 / TOTAL-EXT-VALUE + .005 73 END-PROC
```
<table>
<thead>
<tr>
<th>CITY</th>
<th>PART NUMBER</th>
<th>LAST INVENTORY</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>
</tr>
<tr>
<td>CHICAGO</td>
<td>000-16-490</td>
<td>238</td>
<td>49.95</td>
<td>11,888.10</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-18-944</td>
<td>854</td>
<td>54.99</td>
<td>46,961.46</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-53-100</td>
<td>181</td>
<td>879.65</td>
<td>159,270.95</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-79-740</td>
<td>81</td>
<td>870.80</td>
<td>70,470.00</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-81-190</td>
<td>35</td>
<td>389.19</td>
<td>13,648.25</td>
</tr>
<tr>
<td>E MOLIN</td>
<td>000-82-150</td>
<td>134</td>
<td>197.80</td>
<td>26,785.26</td>
</tr>
<tr>
<td>HAMMOND</td>
<td>000-78-750</td>
<td>358</td>
<td>425.80</td>
<td>152,150.00</td>
</tr>
<tr>
<td>HAMMOND</td>
<td>358</td>
<td>425.80</td>
<td></td>
<td>152,150.00</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-15-980</td>
<td>3,150</td>
<td>14.29</td>
<td>45,813.50</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-51-260</td>
<td>14,389</td>
<td>15.25</td>
<td>219,432.25</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-60-680</td>
<td>654</td>
<td>54.00</td>
<td>35,904.60</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-63-320</td>
<td>385</td>
<td>59.88</td>
<td>23,653.80</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-64-800</td>
<td>459</td>
<td>43.59</td>
<td>19,136.01</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-74-900</td>
<td>86</td>
<td>31.59</td>
<td>2,716.74</td>
</tr>
<tr>
<td>INDIANP</td>
<td>000-81-400</td>
<td>189</td>
<td>81.45</td>
<td>8,878.05</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>
</tr>
<tr>
<td>KANS CT</td>
<td>2,218</td>
<td>8.99</td>
<td></td>
<td>19,939.82</td>
</tr>
<tr>
<td>MAMMOND</td>
<td>000-19-360</td>
<td>2,810</td>
<td>18.95</td>
<td>53,249.50</td>
</tr>
<tr>
<td>MAMMOND</td>
<td>2,810</td>
<td>18.95</td>
<td></td>
<td>53,249.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CITY</th>
<th>PART NUMBER</th>
<th>LAST INVENTORY</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>
</tr>
<tr>
<td>MEMPHIS</td>
<td>001-85-260</td>
<td>2,210</td>
<td>195.59</td>
<td>432,050.00</td>
</tr>
<tr>
<td>MEMPHIS</td>
<td>001-86-600</td>
<td>3,403</td>
<td>159.88</td>
<td>544,071.64</td>
</tr>
<tr>
<td>MEMPHIS</td>
<td>001-87-800</td>
<td>3,952</td>
<td>55.95</td>
<td>221,114.40</td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>000-11-576</td>
<td>88</td>
<td>119.66</td>
<td>10,520.08</td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>000-12-268</td>
<td>805</td>
<td>38.88</td>
<td>31,296.40</td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>000-62-270</td>
<td>245</td>
<td>98.97</td>
<td>24,247.65</td>
</tr>
<tr>
<td>ST PAUL</td>
<td>000-12-440</td>
<td>819</td>
<td>243.69</td>
<td>199,582.11</td>
</tr>
<tr>
<td>ST PAUL</td>
<td>000-13-325</td>
<td>799</td>
<td>159.66</td>
<td>127,568.34</td>
</tr>
<tr>
<td>ST PAUL</td>
<td>1,618</td>
<td>403.35</td>
<td></td>
<td>327,150.45</td>
</tr>
<tr>
<td></td>
<td>80,833</td>
<td>4,443.52</td>
<td></td>
<td>3,122,762.70</td>
</tr>
<tr>
<td>CITY</td>
<td>EXTENDED VALUE</td>
<td>PERCENT OF TOTAL VALUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHICAGO</td>
<td>547,409.45</td>
<td>17.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E MOLIN</td>
<td>317,135.92</td>
<td>10.16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HAMMOND</td>
<td>152,150.00</td>
<td>4.87</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDIANP</td>
<td>377,188.75</td>
<td>12.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KANS CT</td>
<td>19,939.82</td>
<td>.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAMMOND</td>
<td>53,249.50</td>
<td>1.71</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEMPHIS</td>
<td>1,262,462.68</td>
<td>40.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUSKEGN</td>
<td>66,076.13</td>
<td>2.12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST PAUL</td>
<td>327,150.45</td>
<td>10.48</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3,122,762.70</td>
<td>100.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 13 - Error Correction

After reviewing the Inventory by City report in Example 12, 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 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

MM/DD/YY            INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL         PAGE      1

PART NUMBER    CITY     UPDATE-STATUS
000-19-360    MAMMOND      BEFORE
000-19-360    HAMMOND      AFTER
```
Example 14 - Inventory Reduction

A new accountant for the company wishes to reduce the inventory of truck parts (commodity group 19720) by 15 percent. She thinks that this can save a substantial amount of money (since the interest rate is so high) and, therefore, requests a report which indicates how much could be saved. The 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.
FILE INVMSTR FB(200 3000)

`%INVMSTR`

```plaintext
1 *
2 *   EXAMPLE 14
3 *
4 FILE INVMSTR FB(200 3000)
5 *
6 MIN-STOCK-LEVEL W 4 P 0
7 STOCK-REDUCTION-QUANT W 4 P 0
8 PROPOSED-STOCK-QUANT W 4 P 0
9   HEADING('PROPOSED' 'STOCK' 'QUANTITY')
10 STOCK-VALUE-SAVINGS W 5 P 2  HEADING('STOCK VALUE' 'SAVINGS')
11 STOCK-INT-SAVINGS W 5 P 2  HEADING('STOCK INTEREST' 'SAVINGS')
12 *
13 JOB
14 IF ITEM-MFGD-COMMODITY-GROUP NE 19720 . * REJECT UNWANTED RECDS
15 END-IF
16 MIN-STOCK-LEVEL = 1.2 * ITEM-REORDER-POINT + .5
17 IF ITEM-LAST-INVENTORY-QUANTITY LE MIN-STOCK-LEVEL
18     PROPOSED-STOCK-QUANT = 0 . * NO REDUCTION IF ALREADY AT MIN
19     PERFORM REDUCTION-REPORT
20     GOTO JOB
21 END-IF
22 *  
23 STOCK-REDUCTION-QUANT = .15 * ITEM-LAST-INVENTORY-QUANTITY
24 PROPOSED-STOCK-QUANT = .
25 IF PROPOSED-STOCK-QUANT LT MIN-STOCK-LEVEL
26     PROPOSED-STOCK-QUANT = MIN-STOCK-LEVEL
27     PROPOSED-STOCK-QUANT = ITEM-LAST-INVENTORY-QUANTITY -
28     PROPOSED-STOCK-QUANT
29 END-IF
30 PERFORM REDUCTION-REPORT
31 *
32 REDUCTION-REPORT. PROC
33   STOCK-VALUE-SAVINGS = STOCK-REDUCTION-QUANT * LAST-PURCHASE-PRICE
34   STOCK-INT-SAVINGS = .015 * STOCK-VALUE-SAVINGS
35 PRINT SAVINGS-REPORT
36 END-PROC
37 *
38 REPORT SAVINGS-REPORT SKIP 1 LINESIZE 80
39 SEQUENCE STOCK-VALUE-SAVINGS D
40 CONTROL
41 TITLE 1 'STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720'
42 LINE 1 PART-NUMBER ITEM-LAST-INVENTORY-QUANTITY -
43     PROPOSED-STOCK-QUANT -
44     STOCK-VALUE-SAVINGS STOCK-INT-SAVINGS
45 LINE 2 PART-DESCRIPTION
```
<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>FENDERS</td>
<td></td>
<td></td>
<td></td>
<td></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>DOORS</td>
<td></td>
<td></td>
<td></td>
<td></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>HUBS</td>
<td></td>
<td></td>
<td></td>
<td></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>BUMPERS</td>
<td></td>
<td></td>
<td></td>
<td></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>AIR BRAKES</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001-79-000</td>
<td>385</td>
<td>328</td>
<td>1,707.15</td>
<td>25.60</td>
</tr>
<tr>
<td>AXLE SHAFTS</td>
<td></td>
<td></td>
<td></td>
<td></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>BRAKE DRUMS</td>
<td></td>
<td></td>
<td></td>
<td></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>DRIVE SHAFTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001-84-900</td>
<td>86</td>
<td>86</td>
<td>.00</td>
<td>.00</td>
</tr>
<tr>
<td>CYLINDER SLEEVES</td>
<td>11,622</td>
<td>9,897</td>
<td>99,920.76</td>
<td>1,498.77</td>
</tr>
</tbody>
</table>

STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720

LAST PROPOSED INVENTORY STOCK STOCK VALUE SAVINGS STOCK INTEREST SAVINGS
Example 15 - 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 Example 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,” of the CA-Easytrieve/Plus Application Guide.

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.
FILE INVMSTR FB(200 3000)

FILE NEWMSTR FB(200 3000)

FILE UPDTBL TABLE INSTREAM

JOB

SEARCH UPDTBL WITH PART-NUMBER GIVING TABLE-DESC

IF UPDTBL . * IF MATCH FOUND
PRINT AUDIT-TRAIL . * OUTPUT AUDIT TRAIL
PERFORM EXCESS-CHECK . * CHECK FOR LARGE QUANT VARIATION
ITEM-LAST-INVENTORY-DATE = NEW-DATE . * UPDATE DATE AND
ITEM-LAST-INVENTORY-QUANTITY = NEW-QUANT . * QUANTITY
END-IF

PUT NEWMSTR FROM INVMSTR . * OUTPUT UPDATED FILE

EXCESS-CHECK. PROC
IF NEW-QUANT < .8 * ITEM-LAST-INVENTORY-QUANTITY
PERCENT-DROP = 100 - 
- (NEW-QUANT * 100 / ITEM-LAST-INVENTORY-QUANTITY)
PRINT MGMT-WARNING . * IF UNUSUAL DROP IN QUANTITY
END-IF . * INFORM THE MANAGEMENT
END-PROC

REPORT AUDIT-TRAIL LINESIZE 80
TITLE 1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
LINE 1 PART-NUMBER ITEM-LAST-INVENTORY-DATE -
- ITEM-LAST-INVENTORY-QUANTITY -
NEW-DATE NEW-QUANT

REPORT MGMT-WARNING LINESIZE 80
TITLE 1 'INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY'
LINE 1 PART-NUMBER LOCATION-CITY -
- ITEM-LAST-INVENTORY-QUANTITY -
NEW-QUANT PERCENT-DROP
### 11/20/83 INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL

<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>4,027</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>

### 11/20/83 INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>CITY</th>
<th>LAST INVENTORY QUANTITY</th>
<th>NEW INVENTORY QUANTITY</th>
<th>PERCENT DROP IN INVENTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-85-400</td>
<td>INDIANP</td>
<td>109</td>
<td>73</td>
<td>33.02</td>
</tr>
<tr>
<td>001-83-800</td>
<td>INDIANP</td>
<td>439</td>
<td>311</td>
<td>29.15</td>
</tr>
<tr>
<td>001-84-200</td>
<td>MEMPHIS</td>
<td>653</td>
<td>472</td>
<td>27.71</td>
</tr>
</tbody>
</table>
Example 16 - Reorder Notification Report

The Materials Department needs a program which automatically reorders parts when quantities get below a specified level. The program should provide three reports:

1. A master activity report for the materials department,
2. A set of purchase orders to initiate the ordering, and
3. 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 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. Notice the use of 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 via LINE statements. Final totals are suppressed. A new page and renumbering are requested at each vendor control break. Note the use of control variables 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 on the title line.
FILE INVMSTR FB(200 3000)
%INVMSTR

PO# S 10 N HEADING('PURCHASE' 'ORDER' 'NUMBER')
PO-DATE PO# 6 N
PO-SEQ PO# +6 4 N
EXTENDED-TOTAL W 5 P 2 HEADING('EXTENDED' 'TOTAL')

FILE NEWMSTR FB(200 3000)

JOB

IF ITEM-LAST-INVENTORY-QUANTITY > 4.0 * ITEM-REORDER-POINT
   PUT NEWMSTR FROM INVMSTR . OUTPUT NEW MASTER RECORD IF NO
   GOTO JOB . CHANGE, AND GET NEXT RECORD
   END-IF

IF ITEM-LAST-INVENTORY-QUANTITY < ITEM-REORDER-POINT
   LAST-PURCHASE-QUANTITY = 1.2 * LAST-PURCHASE-QUANTITY
   END-IF

%GETDATE PO-DATE . GET DATE IN MMDDYY FORMAT
LAST-PURCHASE-DATE = PO-DATE . SET NEW PURCHASE DATE
EXTENDED-TOTAL = LAST-PURCHASE-QUANTITY * LAST-PURCHASE-PRICE

PRINT ACTIVITY-REPORT . PRINT MASTER ACTIVITY REPORT
PRINT PURCHASE-ORDERS . PRINT PURCHASE ORDERS
PRINT RECEIVING-REPORTS . PRINT RECEIVING REPORTS
PUT NEWMSTR FROM INVMSTR . OUTPUT UPDATED FILE

REPORT ACTIVITY-REPORT SKIP 1 SUMCTL TAG LINESIZE 80
SEQUENCE VENDOR-NUMBER PART-NUMBER
CONTROL VENDOR-NUMBER
SUM EXTENDED-TOTAL
TITLE 1 'PURCHASE ORDER ACTIVITY BY VENDOR'
HEADING LAST-PURCHASE-QUANTITY 'QUANTITY'
HEADING LAST-PURCHASE-PRICE ('ESTIMATED' 'PRICE')
LINE 1 VENDOR-NUMBER PART-NUMBER -
      LAST-PURCHASE-QUANTITY LAST-PURCHASE-PRICE -
      EXTENDED-TOTAL
LINE 2 VENDOR-LOCATION-CITY -2 VENDOR-LOCATION-STATE -
      POS 2 PART-DESCRIPTION

REPORT PURCHASE-ORDERS NOADJUST SKIP 1 SUMCTL NONE LINESIZE 80
SEQUENCE VENDOR-NUMBER PART-NUMBER
CONTROL FINAL NOPRINT VENDOR-NUMBER RENUM
SUM EXTENDED-TOTAL
TITLE 1 COL 25 'ABC COMPANY'
TITLE 2 COL 23 'PURCHASE ORDER'
TITLE 4 COL 1 'PO#' PO#
TITLE 6 COL 1 'VENDOR' VENDOR-NUMBER
TITLE 7 COL 10 VENDOR-LOCATION-CITY -2 VENDOR-LOCATION-STATE
HEADING LAST-PURCHASE-QUANTITY 'QUANTITY'
HEADING LAST-PURCHASE-PRICE ('ESTIMATED' 'PRICE')
LINE 1 PART-NUMBER +10 -
      LAST-PURCHASE-QUANTITY LAST-PURCHASE-PRICE -
      EXTENDED-TOTAL
LINE 2 PART-DESCRIPTION
BEFORE-BREAK. PROC
PO-SEQ = PO-SEQ + 1 . INCREMENT PO NUMBER
END-PROC
### Basic Examples

127 *
128 REPORT RECEIVING-REPORTS LINESIZE 80
129 SEQUENCE LOCATION-CITY VENDOR-NUMBER PART-NUMBER
130 CONTROL FINAL NOPRINT LOCATION-CITY RENUM NOPRINT
131 TITLE 1 'RECEIVING REPORT FOR' LOCATION-CITY 'WAREHOUSE'
132 HEADING LAST-PURCHASE-QUANTITY 'QUANTITY'
133 LINE 1 VENDOR-NUMBER PART-NUMBER LAST-PURCHASE-QUANTITY
134 *

---

**11/20/83**  
**PURCHASE ORDER ACTIVITY BY VENDOR**  
**PAGE 1**

<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>
<tr>
<td>001-79-000</td>
<td>600</td>
<td>29.95</td>
<td></td>
<td>17,970.00</td>
</tr>
<tr>
<td>MILW WI</td>
<td>AXLE SHAFTS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>001-85-400</td>
<td>300</td>
<td>40.00</td>
<td></td>
<td>12,000.00</td>
</tr>
<tr>
<td>MILE WI</td>
<td>DRIVE SHAFTS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDOR-NUMBER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>47,940.00</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>BAY CIT MI</td>
<td>CYLINDER SLEEVES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDOR-NUMBER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>9,774.00</td>
</tr>
<tr>
<td>00-03-4-091</td>
<td>001-84-200</td>
<td>1,000</td>
<td>45.50</td>
<td>45,500.00</td>
</tr>
<tr>
<td>PHIL PA</td>
<td>BUMPERS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDOR-NUMBER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>45,500.00</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>
<tr>
<td>LVILLE KY</td>
<td>DESKS, STEEL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VENDOR-NUMBER TOTAL</td>
<td></td>
<td></td>
<td></td>
<td>70,448.40</td>
</tr>
</tbody>
</table>
## Basic Examples

### 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></td>
<td></td>
<td></td>
<td></td>
<td>901.34</td>
</tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td>60,359.04</td>
</tr>
<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></td>
<td></td>
<td></td>
<td></td>
<td>23,950.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>258,872.78</td>
</tr>
</tbody>
</table>

---

### PURCHASE ORDER

**PO#** 1120810000  
**VENDOR** 00-00-0-562  
**MILW WI**

<table>
<thead>
<tr>
<th>PART NUMBER</th>
<th>QUANTITY</th>
<th>ESTIMATED PRICE</th>
<th>EXTENDED TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>001-78-200</td>
<td>600</td>
<td>29.95</td>
<td>17,970.00</td>
</tr>
<tr>
<td>001-79-000</td>
<td>600</td>
<td>29.95</td>
<td>17,970.00</td>
</tr>
<tr>
<td>001-85-400</td>
<td>300</td>
<td>40.00</td>
<td>12,000.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>47,940.00</td>
</tr>
<tr>
<td>Part Number</td>
<td>Quantity</td>
<td>Price</td>
<td>Total</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>001-84-900</td>
<td>600</td>
<td>16.29</td>
<td>9,774.00</td>
</tr>
<tr>
<td>CYLINDER SLEEVES</td>
<td></td>
<td></td>
<td>9,774.00</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>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>
<tr>
<td>BUMPERS</td>
<td></td>
<td></td>
<td>45,500.00</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Part Number</th>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-81-190</td>
<td>360</td>
<td>195.69</td>
<td>70,448.40</td>
</tr>
<tr>
<td>DESKS, STEEL</td>
<td></td>
<td></td>
<td>70,448.40</td>
</tr>
<tr>
<td>Date</td>
<td>Receiving Report for</td>
<td>Warehouse</td>
<td>Page</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>11/20/83</td>
<td>E Molin</td>
<td>E MOLIN</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>VENDOR</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER</td>
<td>NUMBER</td>
<td>QUANTITY</td>
</tr>
<tr>
<td>10-03-0-443</td>
<td>000-81-190</td>
<td>360</td>
<td></td>
</tr>
<tr>
<td>34-89-7-210</td>
<td>000-53-100</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>65-49-8-318</td>
<td>000-82-150</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11/20/83</th>
<th>Receiving Report for</th>
<th>INDIANP</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VENDOR</td>
<td>PART</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER</td>
<td>NUMBER</td>
<td>QUANTITY</td>
</tr>
<tr>
<td>00-00-0-562</td>
<td>001-78-200</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>00-00-0-562</td>
<td>001-79-000</td>
<td>600</td>
<td></td>
</tr>
<tr>
<td>00-00-0-562</td>
<td>001-85-400</td>
<td>300</td>
<td></td>
</tr>
<tr>
<td>00-00-9-128</td>
<td>001-84-900</td>
<td>600</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11/20/83</th>
<th>Receiving Report for</th>
<th>MEMPHIS</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VENDOR</td>
<td>PART</td>
<td>QUANTITY</td>
</tr>
<tr>
<td>00-03-4-091</td>
<td>001-84-200</td>
<td>1,000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11/20/83</th>
<th>Receiving Report for</th>
<th>MUSKEGN</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VENDOR</td>
<td>PART</td>
<td>QUANTITY</td>
</tr>
<tr>
<td>54-96-3-251</td>
<td>000-11-576</td>
<td>1,008</td>
<td></td>
</tr>
</tbody>
</table>
CA-Easytrieve/Earl

The CA-Easytrieve/Earl file exit program is an assembly program that translates the CA-Easytrieve exit parameter list to the CA-Earl parameter list and invokes a specific CA-Earl program.

Syntax

```
FILE filename WORKAREA (lrecl)                             +
   EXIT ('EARLEXIT' USING (COMAREA, exitname, filename, +
                      PARM - REGISTER))
```

Parameters

filename

The DD name of the data file that the Exit reads.

lrecl

The record length of the file.

EARLEXIT

The name of the CA-Easytrieve/EARL file exit.

COMAREA

An 80-byte working storage field used as a communication area for the specific CA-Earl exit being used. COMAREA must be defined in the CA-Easytrieve/Plus program as an 80-byte alpha field.

exitname

The CA-Earl exit supplied by a specific Computer Associates product.

PARM-REGISTER

CA-Easytrieve predefined field that contains the value of register 1 on entry to the CA-Easytrieve/Plus program.
Usage Notes

COMAREA must be passed to the exit as an 80-byte alpha field. It is redefined into two parts. The first four bytes of COMAREA is an integer that contains the CA-Earl return code. Its values are:

- -1—End of File
- -2—Record not found
- 0—User cancel.

Redefine the remaining 76 bytes of COMAREA per the requirements of the specific exit. This area can be used to pass additional parameters required by the Exit or for user messages.

Please refer to the appropriate product documentation for information on supplied CA-Earl exits.

Sample CA-Easytrieve/Earl Exit

```plaintext
* FILE DEFINITION
*   FILE NAME: AIRPORTS
*   EARL EXIT NAME: EARLGET
*   RECORD LENGTH: 48
FILE AIRPORTS WORKAREA 48
EXIT (EARLEXIT USING (COMAREA, 'EARLGET ', 'AIRPORTS ', +
   PARM-REGISTER)) F (48)
   NAME 1 18 A
   CITY 20 16 A
   COUNTRY 37 3 A
   PASS 41 8 A
DEFINE COMAREA W 80 A
DEFINE EARLCOM-FLAG COMAREA 4 B
DEFINE EARLCOM-MSG COMAREA +4 76 A
JOB INPUT AIRPORTS
PRINT AIRPORTS
REPORT AIRPORTS
CONTROL
LINE 1
   NAME +
   CITY +
   COUNTRY +
   PASS
```
This section describes all files and JCL required to execute CA-Easytrieve/Plus in the OS/390 operating system.

- Files prefixed by SYS, KJ, and SORT are operating system-related files.
- Files prefixed by EZT are CA-Easytrieve/Plus related.
- Files prefixed by PAN are macro file related.

The EZT and PAN prefixes can be respecified by options WKDSPF and MACDDN.

**SYSIN**

(required except for execution-only operation)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Source statement input plus optional data input</td>
</tr>
</tbody>
</table>

**Characteristics**

- Fixed length, 80-bytes

**Considerations**

- Optional data input follows the END statement which delimits the source program input.

**SYSPRINT**

(required)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Compiler and default report output</td>
</tr>
</tbody>
</table>

**Characteristics**

- Fixed length, 121 to 204 bytes.
PANDD1

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provide access to CA-Easytrieve/Plus macros stored in a CA-Panvalet library. (If you are using a PDS or VSAM macro library, use file-name PANDD.)</td>
</tr>
</tbody>
</table>

MASTER

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provide access to CA-Easytrieve/Plus macros stored in a CA-Librarian library.</td>
</tr>
</tbody>
</table>

EZTVFM

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Work file space for the CA-Easytrieve/Plus Virtual File Manager</td>
</tr>
<tr>
<td>Characteristics</td>
<td>DASD file, fixed length, record length computed by VFM. Multiple extents allowed.</td>
</tr>
<tr>
<td>Considerations</td>
<td>VFM is used for work files during compilation, by report spool files, and by user VIRTUAL files during execution.</td>
</tr>
</tbody>
</table>

VFM attempts to buffer all data in storage. If there is insufficient storage to buffer all of the data, an EZTVFM file is required.

**Note:** The EZTVFM file must not span volumes.
You can define an EZTVFM file simply by specifying UNIT and SPACE information on a DD statement. The amount of space required is dependent on the amount of data processed by the VFM during execution. VFM maintains a 90 percent utilization of disk space; if the total number of bytes of data to be maintained by VFM at any one time is known, the formula for cylinder allocation of space is:

\[
\frac{\text{bytes of data}}{0.9 \times \text{track-length} \times \text{trks/cyl}}
\]

### SORTWKnn

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides sort work space for the SORT program</td>
</tr>
<tr>
<td>Considerations</td>
<td>Work files are required only for those systems which do not provide dynamic allocation. If the number of sort work units supplied to CA-Easytrieve/Plus on the SORT or PARM statement or in the options table is between 1 and 31, the DYNALLOC parameter of the OS/390 SORT statement indicates dynamic allocation of work data sets.</td>
</tr>
</tbody>
</table>

### SYSLIN

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Output file for CA-Easytrieve/Plus object modules; used as input to the linkage editor</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Fixed blocked 80/800.</td>
</tr>
</tbody>
</table>

### SYSOUT

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Sort message output</td>
</tr>
<tr>
<td>Characteristics</td>
<td>As required by the sort utility. Normally assigned as SYSOUT=A.</td>
</tr>
</tbody>
</table>
## STEPLIB

(Optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Supplies load modules required by CA-Easytrieve/Plus and its options not available elsewhere.</td>
</tr>
</tbody>
</table>

## sysctl

(IDMS CV)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Supplies control information to CA-IDMS central version.</td>
</tr>
</tbody>
</table>

## sysjrnl

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the CA-IDMS journal file. The journal is usually a tape file.</td>
</tr>
</tbody>
</table>

## sysidms

(IDMS)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the CA-IDMS (release 12.0 and above) environment parameters.</td>
</tr>
</tbody>
</table>

## idmsdb

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the areas comprising the database.</td>
</tr>
</tbody>
</table>
**idmsdict**

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the dictionary to be used for library definitions.</td>
</tr>
</tbody>
</table>

**SYSSNAP**

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides error analysis printout</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Variable blocked 125/882; normally, assigned to SYSOUT=A.</td>
</tr>
</tbody>
</table>

**SYSUDUMP**

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Abnormal error dump data set</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Normally, assigned to SYSOUT=A.</td>
</tr>
</tbody>
</table>

**userfiles**

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides access to files described by CA-Easytrieve/Plus FILE statements</td>
</tr>
<tr>
<td>Characteristics</td>
<td>As required by coding on the FILE statements.</td>
</tr>
</tbody>
</table>

The following list details of the additional file requirements for CA-Easytrieve/Plus when using the IBM Kanji/Chinese Sort/Merge Program Product.
### KJSRTBL

(optional - used only with IBM Kanji/Chinese Sort)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Defines the data sets containing Kanji sort tables</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Normally, an OS/390 PDS containing load module members</td>
</tr>
<tr>
<td>Considerations</td>
<td>Libraries are provided and maintained by IBM-supplied utilities.</td>
</tr>
</tbody>
</table>

### KJSYSOUT

(optional - used only with IBM Kanji/Chinese Sort)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Kanji/Chinese sort message output</td>
</tr>
<tr>
<td>Characteristics</td>
<td>As required by the sort utility (normally: RECFM=FBA, LRECL=121,BLKSIZE=1210). It is normally assigned to SYSOUT=A.</td>
</tr>
</tbody>
</table>

The following list details the additional requirements for CA-Easytrieve/Plus when using the FACOM Kanji Sort/Merge Program Product.

### KATTR

(optional - used only with FUJITSU Kanji/Chinese Sort)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Defines the data sets containing Kanji sort tables</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Normally, an OS/390 PDS containing load module members</td>
</tr>
<tr>
<td>Considerations</td>
<td>Libraries are provided and maintained by FUJITSU-supplied utilities.</td>
</tr>
</tbody>
</table>
JCL Examples: OS/390 JCL Systems

The following example illustrates the JCL necessary to compile and go with sort and external VFM work file.

```jcl
//jobname   JOB accounting.info
//STEPNAME  EXEC  PGM=EZTPA00,REGION=512K
//SYSPRINT  DD SYSOUT=A
//SYSSNAP   DD SYSOUT=A
//SYSSSORT  DD SYSOUT=A
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,1)
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,100))
//userfile  DD dd-parms
//SYSIN     DD *
    ...CA-Easytrieve/Plus source statements...
```

This example illustrates the JCL necessary to compile and link edit a load module to be executed later.

```jcl
//jobname   JOB accounting.info
//STEPNAME  EXEC  PGM=EZTPA00,REGION=512K
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,100))
//SYSLIN    DD UNIT=SYSDA,SPACE=(400,(100,50)),DISP=(,PASS),
    DSN=&&SYSLIN
//SYSIN     DD *
    PARM  LINK(TESTPGM)...
    ...CA-Easytrieve/Plus source statements ....
//LKED      EXEC  PGM=IEWL
//SYSPRINT  DD SYSOUT=A
//SYSLIN    DD DSN=&&SYSLIN,DISP=(OLD,DELETE)
//SYSLMOD   DD DSN=your.eztp.loadlib,DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(1,5))
```

The following example illustrates the JCL necessary to execute a previously compiled and link edited CA-Easytrieve/Plus program.

```jcl
//jobname   JOB accounting.info
//STEPNAME  EXEC  PGM=TESTPGM
//STEPLIB   DD ...
//SYSPRINT  DD SYSOUT=A
//SYSSNAP   DD SYSOUT=A
//SYSSSORT  DD SYSOUT=A
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,1)
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,100))
//userfile  DD dd-parms
//SYSIN     DD *(optional CARD input)
```
The following procedure can be catalogued and used for executing CA-Easytrieve/Plus with IMS.

```
//DLIBATCH PROC   MBR=TEMPNAME,PSB=,BUF=8,SPIE=0,TEST=0,EXCPVR=0,
//          RST=0,PRLD=,SRCH=0,CKPTID=,MON=N,LOGA=0
//G     EXEC PGM=DFSRRC00,REGION=512K,
//     PARM=(DLI,MBR,PSB,BUF,
//     &SPIE&TEST&EXCPVR&RST,&PRLD,&SRCH,&CKPTID,&MON,&LOGA)
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//     DD DSN=IMSVS.PGMLIB,DISP=SHR
//     DD DSN=your.eztp.loadlib,DISP=SHR
//IMS     DD DSN=IMSVS.PSBLIB,DISP=SHR
//     DD DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//EZTVFM   DD UNIT=SYSDA,SPACE=(4096,(100,100))
//SYSOUT   DD SYSOUT=A
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,1)
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,1)
```

When executing CA-Easytrieve/Plus programs with both IMS and DB2 statements (if you are running a version of DB2 prior to 1.3), change the JCL statement given above from:

```
//      PARM=(DLI,&MBR,&PSB,&BUF,
```

to:

```
//      PARM=(BMP,&MBR,&PSB,&BUF,
```

Also, add the IBM DB2 SSPGM library and the Pan/SQL option library to the STEPLIB statement.

The following JCL executes the above procedure:

```
//jobname JOB accounting.info
//EZTPIMS EXEC DLIBATCH,MBR=EZTPA00,PSB=yourpsbname
//PAYFILE DD DISP=SHR,DSN=IMS.DATA.BASE
//PAYFLOW DD DISP=SHR,DSN=IMS.DATA.BASE.OVERFLOW
//SYSIN DD *
//  CA-Easytrieve/Plus IMS statements follow
//  ...
//  /*
```

The following example illustrates the OS/390 JCL to execute CA-Easytrieve/Plus with CA-IDMS under central version.

```
//jobname JOB accounting.info
//stepname EXEC PGM=EZTPA00
//STEPLIB DD DISP=SHR,DSN=your.eztp.loadlib
//     DD DISP=SHR,DSN=cdms.loadlib
//SYSPRINT DD SYSOUT=A
//SYSPUT DD SYSOUT=A
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,1)
//sysctl DD DISP=SHR,DSN=cdms.sysctl
//EZTVFM DD UNIT=SYSDA,SPACE=(4096,(100,100))
//SYSIN DD *
//  ... CA-Easytrieve/Plus source statements ...
///*
```
The following example illustrates the OS/390 JCL to execute a previously compiled and link edited CA-Easytrieve/Plus program with CA-IDMS under central version.

```
//jobname JOB accounting.info
//stepname EXEC PGM=TESTPGM
//STEPLIB DD DISP=SHR,DSN=your.eztp.loadlib
//EZTFVM DD DISP=SHR,DSN=cdms.loadlib
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SORTWK01 DD UNIT=SYSDA, SPACE=(CYL,1)
//sysctl DD DISP=SHR,DSN=cdms.sysctl
//EZTVFM DD UNIT=SYSDA, SPACE=(4096,(100,100))
//SYSIN DD * (optional CARD input)
```

The following example illustrates the OS/390 JCL to execute CA-Easytrieve/Plus with CA-IDMS under local mode.

```
//jobname JOB accounting.info
//stepname EXEC PGM=EZTPA00
//STEPLIB DD DISP=SHR,DSN=your.eztp.loadlib
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SORTWK01 DD SYSOUT=A
//SORTWK01 DD UNIT=SYSDA, SPACE=(CYL,1)
//sysjrnl DD DISP=(NEW,KEEP), UNIT=TAPE,
//              DSN=cdms.tapejrnl
//idmsdict DD DISP=SHR, DSN=cdms.dictdb
//idmsdb DD DISP=SHR, DSN=your.database
//EZTVFM DD UNIT=SYSDA, SPACE=(4096,(100,100))
//SYSIN DD *
... CA-Easytrieve/Plus source statements ...
/*
```

The following example illustrates the OS/390 JCL to execute a previously compiled and link edited CA-Easytrieve/Plus program with CA-IDMS under local mode.

```
//jobname JOB accounting.info
//stepname EXEC PGM=TESTPGM
//STEPLIB DD DISP=SHR,DSN=your.eztp.loadlib
//SYSPRINT DD SYSOUT=A
//SYSOUT DD SYSOUT=A
//SORTWK01 DD UNIT=SYSDA, SPACE=(CYL,1)
//sysjrnl DD DISP=(NEW,KEEP), UNIT=TAPE,
//              DSN=cdms.tapejrnl
//idmsdict DD DISP=SHR, DSN=cdms.dictdb
//idmsdb DD DISP=SHR, DSN=your.database
//EZTVFM DD UNIT=SYSDA, SPACE=(4096,(100,100))
//SYSIN DD * (optional CARD input)
```
The following example illustrates the JCL necessary to execute CA-Easytrieve/Plus with DB2 using Dynamic SQL mode.

//jobname  JOB  accounting.info,USER=userid
//stepname EXEC PGM=EZTPA00
//STEPLIB  DD   DISP=SHR,DSN=your.eztp.loadlib
//         DD   DISP=SHR,DSN=your.pansql.loadlib
//         DD   DISP=SHR,DSN=your.db2.sspgm.lib
//SYSPRINT DD   SYSOUT=A
//SYSOUT   DD   SYSOUT=A
//SORTWK01 DD   UNIT=SYSDA,SPACE=(CYL,1)
//SYSSNAP  DD   SYSOUT=A
//SYSUDUMP DD   SYSOUT=A
//EZTFVM   DD   UNIT=SYSDA,SPACE=(4096,(100,100))
//SYSIN    DD   *
  ... CA-Easytrieve/Plus DB2 source statements ... 
/*

SSPGM names your IBM DB2 SSPGM library, which contains the programs DSNHLI and DSNALI.

For examples on how to run CA-Easytrieve/Plus using Static SQL, see the CA-Easytrieve/Plus DB2 and SQL/DS Interface Option guide.

**Macro Libraries**

When including macros in your program, you must have JCL that refers to your macro libraries. For example, PANDD1 provides access to CA-Easytrieve/Plus macros stored in a CA-Panvalet library. MASTER provides access to macros stored in a CA-Librarian library. If you are using a PDS or VSAM macro library, use file name PANDD. See the CA-Easytrieve/Plus Installation Guide, or your system administrator for details.
JCL Examples for the VSE Operating System

This section describes all files and JCL required for executing CA-Easytrieve/Plus in the VSE operating system.

- Files prefixed by SYS, KJ, and SORT are operating system-related files.
- Files prefixed by EZT are CA-Easytrieve/Plus related.
- Files prefixed by PAN are macro file related.

The EZT and PAN prefixes can be respecified by options WKSDFSP and MACDNN.

**SYSIPT**

(required for compilation only)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Source statement input plus optional data input</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Fixed length, 80-bytes</td>
</tr>
<tr>
<td>Considerations</td>
<td>Optional data input follows the END statement, which delimits the source program input. The file can be assigned to any device supported by DTFCP for input.</td>
</tr>
</tbody>
</table>

**SYSLST**

(required)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Compiler and default report output</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Fixed length, 121 to 133 bytes</td>
</tr>
<tr>
<td>Considerations</td>
<td>The file can be assigned to any device supported for output by DTFCP.</td>
</tr>
</tbody>
</table>
### PANDD1

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides access to CA-Easytrieve/Plus macros stored in a CA-Panvalet library. (If you are using a VSAM macro library, use file-name PANDD or if you are using an SSL macro library, use a LIBDEF to define the source library to be searched for macro members.)</td>
</tr>
</tbody>
</table>

### MASTER

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides access to CA-Easytrieve/Plus macros stored in a CA-Librarian library.</td>
</tr>
</tbody>
</table>

### EZTVFM

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Work file space for the Virtual File Manager</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Sequential DASD workfile, fixed length, record length determined by VFM</td>
</tr>
<tr>
<td>Considerations</td>
<td>VFM is used for work files during compilation, by report spool files, and by user VIRTUAL files during execution.</td>
</tr>
</tbody>
</table>

VFM attempts to buffer all data in storage. If there is insufficient storage to buffer all data, an EZTVFM data set is required.

The EZTVFM data set must be a single extent data set on DASD. The size of the EZTVFM data set depends on the amount of data to be maintained by the VFM during execution. VFM maintains at least 90 percent utilization of the disk space. If the total number of bytes to be maintained by VFM at any one time is known, the formula for tracks of disk space is:

\[
\frac{\text{bytes of data}}{0.9 \times \text{track-length}}
\]
### SORTWKn

*(optional)*

<table>
<thead>
<tr>
<th><strong>File-name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provide sort work space for the VSE sort</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Either DASD or tape files as required by the sort program</td>
</tr>
<tr>
<td>Considerations</td>
<td>If work files are DA, then this must match the options table NUMWORK specification or be overridden on the PARM statement.</td>
</tr>
</tbody>
</table>

### SYSLNK

*(optional)*

<table>
<thead>
<tr>
<th><strong>File-name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Output file for CA-Easytrieve/Plus object modules; used as input to the VSE linkage editor</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Fixed length, 82 bytes</td>
</tr>
<tr>
<td>Considerations</td>
<td>The file can be assigned to any device supported by DTFCP for output.</td>
</tr>
</tbody>
</table>

### userfiles

*(optional)*

<table>
<thead>
<tr>
<th><strong>File-name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Provides access to files described by CA-Easytrieve/Plus FILE statements</td>
</tr>
<tr>
<td>Characteristics</td>
<td>As required by coding on the FILE statements.</td>
</tr>
</tbody>
</table>

### CA-IDMS

*(CA-IDMS)*

<table>
<thead>
<tr>
<th><strong>File-name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Supplies the required IDMS PHASEs.</td>
</tr>
</tbody>
</table>
sysjrnl

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the IDMS journal file. The journal is usually a tape file. The file number is given by ( f ).</td>
</tr>
</tbody>
</table>

gsidms

(IDMS)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the CA-IDMS (release 12.0 and above) environment parameters.</td>
</tr>
</tbody>
</table>

idmsdb

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the areas comprising the database.</td>
</tr>
</tbody>
</table>

idmsdict

(IDMS local)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Identifies the dictionary to be used for library definitions.</td>
</tr>
</tbody>
</table>

UPSI

(optional)

<table>
<thead>
<tr>
<th>File-name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Supplies variable information to the CA-Easytrieve/Plus program or to IDMS.</td>
</tr>
</tbody>
</table>
JCL Examples: VSE JCL Systems

The following example illustrates the JCL necessary to compile and go with sort and external VFM work file.

*  $$ JOB JNM=jobname
// JOB jobname
// DLBL EZTP,'your.eztp.library',0,SD
// EXTENT SYS003,volser,1,0,start,lgth
// ASSGN SYS003,mmm
// LIBDEF PHASE,SEARCH=EZTP.sublib,TEMP
// ASSGN SYS001,...
// ASSGN SYS010,...
// ASSGN SYS008,...
// DLBL SORTWK1,,0,DA
// EXTENT SYS001.volser...start,lgth
// DLBL EZTVFM,,0,SD
// EXTENT SYS010.volser...start,lgth
// DLBL INREC,,0,SD
// EXTENT SYS008.volser...start,lgth
// EXEC EZTPA00,SIZE=512K
...CA-Easytrieve/Plus source statements...
/*
/&
* $$ EOJ

This example illustrates the JCL necessary to compile and link edit a phase to be executed later.

*  $$ JOB JNM=jobname
// JOB jobname
// DLBL EZTP,'your.eztp.library',0,SD
// EXTENT SYS003,volser,1,0,start,lgth
// ASSGN SYS003,mmm
// LIBDEF PHASE,CATALOG=EZTP.sublib,TEMP
// LIBDEF PHASE,SEARCH=EZTP.sublib,TEMP
// ASSGN SYS010,...
// DLBL EZTVFM,,0,SD
// EXTENT SYS010.volser...start,lgth
// OPTION CATAL
// EXEC EZTPA00,SIZE=512K
PARM LINK(TESTPGM)
...CA-Easytrieve/Plus source statements...
/*
// EXEC LNKEDT
/&
* $$ EOJ
The following example illustrates the JCL necessary to execute a previously compiled and link edited phase.

```
* $$ 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
  /*
  /*
  /* $ E0J
```

The following JCL can be used for executing CA-Easytrieve/Plus with DLI.

```
* $$ JOB JNM=jobname
  JOB EZTDLI
  UPSI 00000000
  ASSGN SYS011,nnn
  TLBL LOGOUT
  ASSGN SYS006,nnn
  DLBL INVPRT1,'INVENTORY',99/365,VSAM
  EXTENT SYS006.volser
  EXEC DLZRRC00,SIZE=512K
  DLI,EZTPA00,psbname
  /*
  /*
  /* $ E0J
```

When executing CA-Easytrieve/Plus programs with both DLI and SQL/DS statements, change the JCL statement given above from:

```
// EXEC DLZRRC00,SIZE=512K
```

to:

```
// EXEC DLZRRC00,SIZE=(AUTO,xxxK)
```

where xxxK is the amount of storage needed to execute your CA-Easytrieve/Plus program. xxxK is usually between 256K and 512K.
The following example illustrates the VSE JCL to execute CA-Easytrieve/Plus with CA-IDMS under central version.

```cl
* $$ JOB JNM=jobname
  // JOB     jobname
  // UPSI    b
  // DLBL    EZTP,'your.eztp.library'
  // EXTENT  ,volser
  // DLBL    IDMS,'cdms.library'
  // EXTENT  ,volser
  // LIBDEF PHASE,SEARCH=(EZTP.sublib,IDMS.sublib)
  // DLBL    SORTWK1,,0
  // EXTENT  SYS001.volser...start.length
  // DLBL    EZTVFM,.0,SD
  // EXTENT  SYS001.volser...start.length
  // EXEC    EZTPA00
  ... CA-Easytrieve/Plus source statements ...
/*
```

The following example illustrates the VSE JCL to execute CA-Easytrieve/Plus with CA-IDMS under local mode.

```cl
* $$ JOB JNM=jobname
  // JOB     jobname
  // UPSI    b
  // DLBL    EZTP,'your.eztp.library'
  // EXTENT  ,volser
  // DLBL    IDMS,'cdms.library'
  // EXTENT  ,volser
  // LIBDEF PHASE,SEARCH=(EZTP.sublib,IDMS.sublib)
  // ASSGN   SYS001.DISK,VOL=volser,SHR
  // DLBL    SORTWK1,,0
  // EXTENT  SYS001.volser...start.length
  // DLBL    idmsdb,'your.database',,DA
  // EXTENT  SYS004.volser...start.length
  // ASSGN   SYS004.DISK,VOL=volser,SHR
  // DLBL    idmsdb,'cdms.dictdb',,DA
  // EXTENT  SYS005.volser...start.length
  // ASSGN   SYS005.DISK,VOL=volser,SHR
  // DLBL    SYS009.cuu
  // TLBL    sysjrnl,'cdms.tapejrnl',,volser,,f
  // DLBL    EZTVFM,.0,SD
  // EXTENT  SYS010.volser...start.length
  // EXEC    EZTPA00
  ... CA-Easytrieve/Plus source statements ...
/*
```
The following example illustrates the VSE JCL to execute a previously compiled and link edited CA-Easytrieve/Plus program with CA-IDMS under central version.

* $$ JOB JNM=jobname
  // JOB      jobname
  // UPSI     b
  // DLBL     EZTP,'your.eztp.library'
  // EXTENT   .volser
  // DLBL     IDMS,'cdms.library'
  // EXTENT   .volser
  // LIBDEF   PHASE,SEARCH=(EZTP.sublib,IDMS.sublib)
  // ASSGN    SYS001.DISK,VOL=volser,SHR
  // DLBL     SORTWK1,,0
  // EXTENT   SYS001.volser...start,length
  // DLBL     EZTVFM,,0,SD
  // EXTENT   SYS010.volser...start,length
  // EXEC     TESTPGM
    ... CA-Easytrieve/Plus source statements ...
  /*

The following example illustrates the VSE JCL to execute a previously compiled and link edited CA-Easytrieve/Plus program with CA-IDMS under local mode.

* $$ JOB JNM=jobname
  // JOB      jobname
  // UPSI     b
  // DLBL     EZTP,'your.eztp.library'
  // EXTENT   .volser
  // DLBL     IDMS,'cdms.library'
  // EXTENT   .volser
  // LIBDEF   PHASE,SEARCH=(EZTP.sublib,IDMS.sublib)
  // ASSGN    SYS001.DISK,VOL=volser,SHR
  // DLBL     SORTWK1,,0
  // EXTENT   SYS001.volser...start,length
  // ASSGN    SYS004.DISK,VOL=volser,SHR
  // DLBL     idmsdb,'your.database',DA
  // EXTENT   SYS004.volser...start,length
  // ASSGN    SYS005.DISK,VOL=volser,SHR
  // DLBL     idmsdb,'cdms.dictdb',DA
  // EXTENT   SYS005.volser...start,length
  // ASSGN    SYS009.cuu
  // TLBL     sysjrnl,'cdms.tapejrnl',volser,f
  // DLBL     EZTVFM,,0,SD
  // EXTENT   SYS010.volser...start,length
  // EXEC     TESTPGM
    ... CA-Easytrieve/Plus source statements ...
  /*
The following example illustrates the JCL necessary to execute CA-Easytrieve/Plus with SQL/DS.

```jcl
* $$ JOB JNM=jobname
  // JOB jobname
  // UPSI b
  // DLBL EZTP,'your.eztp.library'
  // EXTENT .volser
  // ASSGN SYS001.DISK,VOL=volser,SHR
  // DLBL SORTWK1,,0
  // EXTENT SYS001.volser,,start,lgth
  // ASSGN SYS010.DISK,VOL=volser,SHR
  // DLBL EZTVFMI,,0,SD
  // EXTENT SYS010.volser,,start,lgth
  // DLBL SQLLIB,'your.pansql.library',0,SD
  // EXTENT .volser
  // LIBDEF PHASE,SEARCH=(SQLLIB.sublib,EZTP.sublib)
  // EXEC EZTPA00,SIZE=512K
  PARM USERID('user-id' 'password')
  ... CA-Easytrieve/Plus SQL/DS source statements ...
  /*
  */
  * $$ EOJ
```

**Note:** When executing a CA-Easytrieve/Plus program with SQL/DS for VSE, no JCL changes are required for Multiple User Mode. For the Single User Mode of SQL/DS for VSE, however, you should specify the `SIZE=(AUTO,xxxK)` parameter on your JCL EXEC statement, where `xxxK` is the amount of storage required for normal CA-Easytrieve/Plus execution. The `xxxK` value is usually between 256K and 512K.

### Macro Libraries

When including macros in your program, you must have JCL that refers to your macro libraries. For example, PANDD1 provides access to CA-Easytrieve/Plus macros stored in a CA-Panvalet library. MASTER provides access to macros stored in a CA-Librarian library. If you are using a VSAM macro library, use file name PANDD. If you are using an SSL macro library, use a LIBDEF to define the source library you want to search for macro members. See the *CA-Easytrieve/Plus Installation Guide*, or your systems administrator for details.
Introduction

CA-Easytrieve/Plus has features to permit you to access the system date in a four-digit year format and to have the full year specified on your CA-Easytrieve reports.

In addition, the product includes two CA-Easytrieve macros, DATECONV and DATEVAL, to assist in date conversion efforts. These macros can be used to convert your dates from a two-digit year format to a four-digit year format, as well as to determine which format your dates are in to begin with.

Obtaining a Four-Digit System Date

There is a system-defined field called SYSDATE-LONG. When used in your CA-Easytrieve program, SYSDATE-LONG provides the system date with a four-digit year. The format of the returned date is determined by the DATE option in your CA-Easytrieve/Plus Options Table.

Printing a Four-Digit Year on Reports

The options SHORTDATE and LONGDATE on the REPORT statement enable the printing of either a two-digit or a four-digit year when a date is printed on reports. The default date format is specified through the LONGDTE option in the Options Table.

The default is LONGDTE=NO which prints a two-digit year. To get four-digit years on report dates, either change the Options Table to LONGDTE=YES or specify LONGDATE on your REPORT statement.
DATECONV

The DATECONV routine converts a date-in-one format to any other date format. For example, you can convert month-day-year to year-month-day, Julian to Gregorian, and similar date conversions.

**Note:** Using non-numeric data or a zero for date fields results in a PAP299 error message which displays the field in error, along with its contents. Execution of the program stops, and a return code 32 is generated.

**Syntax**

```
DATECONV date1 format1 date2 format2 [THRESHOLD value]
```

**Parameters**

date1

Specify the name of the field containing the date to be converted. The date in this field must be in the format specified by format1. The name of any previously defined numeric field is valid.

format1

Specify the format of the date1 field. Format1 is a literal description of pairs of letters. The letters indicate positions, as follows:

- **MM** = month
- **DD** = day
- **YY** = year
- **CC** = century

The value of date1 is not checked for a valid date in conjunction with the specified format. However, CC always maintains the value specified in accordance with the THRESHOLD parameter. If you want date validation, use the DATEVAL routine before using DATECONV. The only valid Julian format is YYDDD.

The following are some, but not all, of the valid formats:

- DDMMYY
- MMDDYY
- MMDDCCYY
- YYYYMMDD
- YYDDD (Julian)
- CCYMMDD
- CCYYDDD
- DDMMCCYY
date2

Specify the name of the field to which the converted date is written. The date is written in the format specified by format2. A valid name is any previously defined field.

format2

Specify the format for the date2 field.

[THRESHOLD value]

This optional parameter establishes the upper end of a 100-year range in the 20th and 21st centuries, used in converting YY to CCYY.

For example, if THRESHOLD if 40, the lower boundary is 1941 and the upper boundary of the range is set to 2040. When converting YY to CCYY, each year is assigned a two-position century, based on the range established by THRESHOLD. The default value for THRESHOLD is 0. This causes all dates to have a range of 1901 through 2000.

If year is 52 when THRESHOLD is 40, century is 19; if year is 21, century is 20.

Valid values for THRESHOLD are 0 through 99.

Operation - INLINE

DATECONV generates no output and can be used alone or with other routines and/or CA-Easytrieve/Plus logic.

Operation - Database

No change in the specification of parameters is required to use DATECONV with database files.

Example

The following is an example of DATECONV:

INPUT

FILE ...
JULIAN-DATE  1  S  N
GREG-DATE    W  6  N
...
JOB ...
%DATECONV JULIAN-DATE YYDDD GREG-DATE YYMMDD
...
RESULTS

The Julian date in the field name JULIAN-DATE is converted to Gregorian format and the result is stored in the field named GREG-DATE.

DATEVAL

The DATEVAL routine examines the content of a specified date field for a valid date in accordance with a specified date format. If the date field contains a valid date, the field DATEVAL-FLAG is set to the value YES. If the date field is invalid, the DATEVAL-FLAG is set to the value NO.

Syntax

\%DATEVAL field format [THR\$HOLD value]

Parameters

field

Specify the name of the field that contains the date being validated. Valid names include any previously-defined numeric field.

format

The format for the comparison is a literal description of pairs of letters. The letters indicate positions, as follows:

MM = month
DD = day
YY = year
CC = century

You can specify the letter pairs in any order. YY must be specified whenever you specify CC. The only valid Julian format is YYDDD.

The following are valid formats:

MMDDYY
MMDDCCYY
YYMMDD
YYDDD (Julian)
Specify a value that establishes the upper end of a one-hundred-year range in the 20th and 21st centuries used to control the CC portion of generated dates.

For example, if THRESHOLD is 40, the upper boundary of the range is set to 2040 and the lower boundary is 1941. When converting YY to CCYY, each year is assigned a two-position century based on the range established by THRESHOLD. The default value for THRESHOLD is 0. This causes all dates to have range of 1901 through 2000.

In this example, if year is 52, century is 19; if year is 21, century is 20.

It is important that the THRESHOLD value be correct for the range of dates to be generated. For example, if DATEVAL is invoked to validate dates between the years 1949 and 1952 and THRESHOLD is 50, the years 1949 and 1950 become 2049 and 2050, while the years 1951 and 1952 remain 1951 and 1952. In this respect, the YY (year) portion of the date controls the CC (century) portion in accordance with the THRESHOLD value.

General rules for specifying THRESHOLD values are:
- If the dates to be generated do not exceed the year 2000, specify a THRESHOLD default to a value of 0.
- If the dates exceed the year 2000, choose a THRESHOLD high enough to generate correct dates in the 21st century, but not so high as to convert dates from the 20th century to the 21st century.
- When dates to be generated do not involve calculations for century, specify a THRESHOLD default to a value of 0.
- Valid values for THRESHOLD are 0 through 99.
- The THRESHOLD value is ignored if you provide a century value (CC).

Operation - INLINE

The date field is compared to the specified format. For comparison to be valid, the respective MM, DD, YY, and CC fields must contain valid values. For example:
- If the date field contains 043184, and the format field contains MMDDYY, comparison is invalid because the DD (day) portion of the date exceeds 30 for the month of April.
- If the date field contains 022979, comparison is invalid, because 1979 is not a leap year.
DATEVAL does not produce a report.

- If the date field contains a valid date, an internal field DATEVAL-FLAG is set to the value YES.
- If the date field is invalid, the DATEVAL-FLAG is set to the value NO.

To perform further processing activities, you must code CA-Easytrieve/Plus logic following the invocation of DATEVAL. You can code IF statements that test the DATEVAL-FLAG field.

For example, you can print a report of invalid dates, write all records with valid dates to an output file, and perform further processing of the invalid dates, or any combination of events. The example demonstrates coding IF, DISPLAY, and END-IF statements.

**Operation - Database**

No change in the specification of parameters is required to DATEVAL with database files.

**Example**

The following is an example of DATEVAL:

```
INPUT

FILE ...
  DATE          1   6   N
  INVOICE-NUM   7   4   P
  ...
JOB ...
%DATEVAL DATE MMDDYY
IF DATEVAL-FLAG EQ 'NO'
  DISPLAY +5 INVOICE-NUM +5 DATE
END-IF
  ...

OUTPUT

This example prints the invoice number and date for every record with an invalid date, according to the format MMDDYY.

2983    083781
3953    023072
4263    063184
5337    131278
7654    000000
```
Index

$  
$ edit mask, 5-29, 5-30

&  
&, 16-11

*  
* edit mask, 5-29, 5-31  
* field location, 5-25  
* Field location, 5-18

+  
+/ - continuation characters, 3-11

<  
<, 8-4  
<=, 8-4

=  
=, 8-4

>  
>, 8-4  
>=, 8-4

9  
9 edit mask, 5-29

A  
ABEXIT, 4-1, 4-5  
NO, C-1  
SNAP, C-1  
Abnormal termination, 15-14  
ACCESS  
record, 16-3, 16-5  
statement, 16-5  
Access modes  
data locate, 12-2  
data move, 12-2  
macro library storage, 16-3  
Activity within programs  
data reference, 5-2, 5-33  
DEFINE, 5-15  
field definition, 5-15  
functions, 3-1, 6-1  
logic, 3-5  
PROC coding, 10-5, 10-6  
Addition, See arithmetic  
Address, record, 12-3
<table>
<thead>
<tr>
<th>Term</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF2</td>
<td>C-19</td>
</tr>
<tr>
<td>AFTER-BREAK procedure</td>
<td>11-7, 11-63, 11-69</td>
</tr>
<tr>
<td>AFTER-LINE procedure</td>
<td>11-7, 11-63, 11-66</td>
</tr>
<tr>
<td>Alphabetic conversion literals</td>
<td>3-15</td>
</tr>
<tr>
<td>ALPHABETIC test</td>
<td>8-9, 8-10</td>
</tr>
<tr>
<td>Alphanumeric subjects</td>
<td>8-5</td>
</tr>
<tr>
<td>ALTSEQ</td>
<td>4-5, 4-12</td>
</tr>
<tr>
<td>installation option, C-2</td>
<td></td>
</tr>
<tr>
<td>SORT activities, C-2</td>
<td></td>
</tr>
<tr>
<td>AND, combined conditions</td>
<td>8-4</td>
</tr>
<tr>
<td>Annotating summary lines</td>
<td>C-19</td>
</tr>
<tr>
<td>Apostrophe (')</td>
<td>use, 3-12, 3-21</td>
</tr>
<tr>
<td>Arithmetic</td>
<td></td>
</tr>
<tr>
<td>assignment statement, 7-1</td>
<td></td>
</tr>
<tr>
<td>averages, 11-68</td>
<td></td>
</tr>
<tr>
<td>decimals, 7-3</td>
<td></td>
</tr>
<tr>
<td>expressions, 7-1</td>
<td></td>
</tr>
<tr>
<td>leading zeros, 5-30</td>
<td></td>
</tr>
<tr>
<td>moving numbers, 7-1</td>
<td></td>
</tr>
<tr>
<td>negative numbers, 5-31</td>
<td></td>
</tr>
<tr>
<td>operators, 3-21</td>
<td></td>
</tr>
<tr>
<td>parentheses, 7-2</td>
<td></td>
</tr>
<tr>
<td>percentages, 11-68</td>
<td></td>
</tr>
<tr>
<td>Array, See tables and arrays</td>
<td></td>
</tr>
<tr>
<td>ASA (OS/390)</td>
<td>5-10</td>
</tr>
<tr>
<td>Assignment statement, 7-5</td>
<td></td>
</tr>
<tr>
<td>function, 7-1</td>
<td></td>
</tr>
<tr>
<td>syntax, 7-5</td>
<td></td>
</tr>
<tr>
<td>Automatic file input</td>
<td></td>
</tr>
<tr>
<td>controlled statement use, 12-2</td>
<td></td>
</tr>
<tr>
<td>file-name qualifier, 5-34</td>
<td></td>
</tr>
<tr>
<td>JOB, 6-2, 12-1</td>
<td></td>
</tr>
<tr>
<td>SORT, 12-1</td>
<td></td>
</tr>
<tr>
<td>synchronized input, 6-3</td>
<td></td>
</tr>
<tr>
<td>BIND (SQL parameter)</td>
<td>4-5, 4-10</td>
</tr>
<tr>
<td>Bit manipulation, 2-3, 7-1</td>
<td></td>
</tr>
<tr>
<td>Blank-when-zero displays, 2-3, 5-17, 5-23</td>
<td></td>
</tr>
<tr>
<td>Block-control-word, 12-3</td>
<td></td>
</tr>
<tr>
<td>BLOCKO, installation option, C-2</td>
<td></td>
</tr>
<tr>
<td>Branching logic, 8-1</td>
<td></td>
</tr>
<tr>
<td>BREAK, 2-6, 8-10, 11-45</td>
<td></td>
</tr>
<tr>
<td>BREAK-LEVEL, 8-10, 11-45</td>
<td></td>
</tr>
<tr>
<td>Buffers</td>
<td></td>
</tr>
<tr>
<td>allocation, 5-13, 12-5</td>
<td></td>
</tr>
<tr>
<td>BUFNO, 5-13</td>
<td></td>
</tr>
<tr>
<td>default, 5-13</td>
<td></td>
</tr>
<tr>
<td>I/O, C-3</td>
<td></td>
</tr>
<tr>
<td>BUFNO, 5-13</td>
<td></td>
</tr>
<tr>
<td>installation option, C-3</td>
<td></td>
</tr>
<tr>
<td>parameter of FILE statement, C-3</td>
<td></td>
</tr>
<tr>
<td>BWZ, See blank-when-zero displays</td>
<td></td>
</tr>
</tbody>
</table>

**C**

CA-Corporate Tie external security interface, 12-22
CA-Earl exit, D-55
CA-Easytrieve/Earl, D-55
CA-Easytrieve/Plus
- environment, See environment and PARM
- environment options table, C-1
- file-types processed, 5-5
- keywords, 3-12
- operating modes, 12-6
- operating modes, syntax check/compile, 4-10
- programs, See programs
- source statements, 3-1, 12-6
- statements, See statements
- subprogram CALL exits, 14-1
- syntax check/compile operating mode, 4-8
- top-to-bottom logic, 3-5, 6-1

CA-Easytrieve/Plus operating modes
- object module production, 3-1
- syntax check, 3-1
- syntax check/compile, 3-1, 4-6
- syntax check/compile/execute, 3-1

CA-IDMS IDD interface, processing MOVE LIKE, 7-18
CA-IDMS interface
- automatic file input, 6-2
- automatic path processing, 12-1

CA-IDMS update functions, C-20

CALCDUP, installation option, C-3

CA-Librarian, 16-4
- macro library, C-9
- maintenance, 16-4
- security, 16-4

CALL statement, 14-11
- passing parameters, 14-8, 14-12
- RETURNS parameter, 14-12
- syntax, 14-11

Capabilities, 2-1

CA-Panvalet
- macro library, C-9
- macro storage, 16-3, 16-5

CARD, 5-3, 5-10, 5-12
- files, 12-3, 12-5, 12-6

CASE, 8-25
- statement, 8-25
- statement usage, 8-27

Character sets, 3-8
- DBCS, 3-8
- wrapping, 3-8

CHECKPOINT, JOB statement, 6-4

CHKP-STATUS, 12-4

CISIZE (VSE), 5-13

CLIST, 4-5, 4-8, 15-9
- installation option, C-3

CMS minidisk, file mode, C-3

COBOL
- environment, 4-5, 4-9
- links, 14-4

Coding program statements, 3-2

Collating sequence table, C-2

Column separation, SPREAD, C-17

Combined conditions
- AND, 8-4
- OR, 8-4

Comma (,), use of, 3-12, 3-21

Comment statements, 3-11

Comparison, varying length fields, 8-3

COMPILe, 4-1, 4-5, 4-6

Compile listing, 15-3

Compiler
- directives, 15-2
- length of output lines, C-8
- work file, C-13

Compiler listings
- CLIST, 4-8
- COMPILe, 4-6
- PMAP, 4-8

COMPNME, installation option, C-4

Conditional expressions, 8-1
- DBCS, 8-3
- IF, 8-2
- MIXED fields, 8-3

CONSOLE, C-14

Continuations, 3-11
- DBCS, 3-11

Control breaks, See reports

Control code, security, 16-3

CONTROL record, See control code

CONTROL reports, 11-42

CONTROL statement, 8-10, 11-24, 11-41, 11-42

CONTROL, print carriage, 9-3

Controlled file input
- I/O within JOB, 12-1, 12-2
- statements, 9-1, 12-1, 12-2

CONTROLSKIP, 11-15
- parameter, 2-7

COPY statement, 5-31
- DLI, 5-32
- field duplication, 2-3, 5-2, 5-31
- syntax, 5-31

CREATE option, 5-3, 5-6

Currency symbol, C-11

Current record count, 2-2
D

Data
  map, 15-10, C-6
  reference, 5-33
  strings, 13-15

Data access mode
  locate, 12-2
  move, 12-2

Data definition
  data reference rules, 5-33
  delimiters, 3-12, 3-21
  FDKLDK reference check, 4-8
  field-name qualifiers, 5-34
  fields, See field definitions
  files, See FILE statement
  format, 5-19
  indexing, 5-34
  library, See library names, 3-13

Database
  CA-IDMS interface, 12-1
  FILE statement, 5-3
  IDMS interface, 5-3, 5-6
  IMS/DLI interface, 5-4, 5-5, 5-32

Datacom/DB parameters, 4-11

DATE
  installation option, C-4
  report, See SHORTDATE. See LONGDATE system, See SYSDATE

DATECONV, E-2
DATEVAL, E-4

DB2 parameters, 4-10

DBCS
  capabilities, 2-2
  conditional expressions, 8-3
  format hexadecimal literals, 3-15
  format literals, 3-14
  options module, 15-3
  reporting restrictions, 11-5
  subjects, 8-6

DBCS CODE parameter
  FILE statement, 5-4, 5-14
  PARM statement, 4-2, 4-6

DCBEXCD1, 12-3

DDname, sort messages, C-15

DEBUG, 2-5, 4-1, 4-5, 4-8, 11-72
  CLIST, C-3
  DMAP, C-6
  FLDCHK, C-7
  FLOW, C-8
  FLOWSIZ, C-8
  NOCLIST, C-3
  NODMAP, C-6
  NOFLDCHK, C-7
  NOFLOW, C-8
  NOPMAP, C-13
  NOSTATE, C-18
  NOXREF, C-23
  parameter of PARM statement, C-3, C-23
  PMAP, C-13
  STATE, C-18
  XREF, C-23

DEBUG, XREF
  LONG, C-23
  SHORT, C-23

Debugging, See also error messages features, 2-5

Decimal numbers, 5-20, 5-23, 7-3

Decision and branching logic
determining field contents, 12-4
record availability, 12-4

Decision logic, 8-1

DEFER, 5-13, 12-5

DEFINE statement, 5-15, 5-23, 5-25
  activity statement, 5-15, 5-17
  edit masks, 5-29
  examples, 5-25
  field definitions, 5-16, 5-28
  HEADING, 5-22, 11-34
  INDEX, 5-22
  library statement, 5-15, 5-17, 5-25
  record layouts, 5-25
  syntax, 5-17

DEVICE, 4-2, 4-9, 4-12, 4-16, C-5
  3330, C-5
  3340, C-5
  3350, C-5
  3375, C-5
  3380, C-5
  3390, C-5
DISK, C-5
FBA, C-5
TAPE, C-5

Device type
default, C-5, C-10
for CA-Panvalet macro library, C-9
VFM overflow file, C-22

DEVICE, installation option, C-5

Diagnostic format, 15-8

Diagnostics, see error messages, A-1

DISK
  3330, C-5
  3340, C-5
  3350, C-5
  3375, C-5
  3380, C-5
  3390, C-5
  FBA, C-5
  installation option, C-5

DISPLAY statement, 9-1, 9-2
  function, 5-10, 9-10, 11-1
  report annotations, 11-61
  syntax, 9-2

Division, arithmetic, 3-21, 7-3

DLET, C-20

DLI
  parameter of FILE statement, 5-3, 5-5
  statement, 6-15
  update function codes, C-20

DLIV, installation option, C-6

DMAP, 2-5, 4-5, 4-8, 11-72, 15-10
  installation option, C-6

DO and END-DO statements, 8-22

DO loop nesting example, 8-24

DO UNTIL, 8-23

DO WHILE, 8-23
  program control, 2-3

Documentation conventions, 1-5

DOS/VSE environment, C-19
  output, 2-1
  PUNCH files, 12-8
  record format, 5-11
  storage, 14-2, 14-3
  subprogram LOADING, 14-2

Double Byte Character Set, See DBCS

DOWN, installation option, C-6

DQSCGEN messages, A-44

DTLCTL
  FIRST, C-7
  installation option, C-7
  NONE, C-7
  parameter of REPORT statement, C-7

DUPLICATE, 8-17

DUPS parameter of RETRIEVE statement, C-3

DYNAMIC (SQL parameter), 4-10

E

EBCDIC
  alphabetic literals, 3-13
  DBCS conversion, 2-6, 7-10
  format hexadecimal literals, 3-14

Edit characters, numeric, C-11

edit masks
  Z, 9, *, $, -, 5-30

Edit masks
  BWZ, 5-23
  currency, 5-29, 5-30
  decimal, 5-23
  default, 5-24
  DEFINE, 5-17, 5-29
  editing rules, 5-30
  function, 5-29
  HEX, 5-24
  installation defined, C-21
  MASK, 5-22
  negative numbers, 5-31
  social security number, 5-24
  user-defined, 5-22
  Z, 9, *, $, -, 5-29

ELSE, 8-18

ELSE-IF, 8-18

END delimiter, CARD option on FILE statement, 5-10

End of job, See FINISH

END record, 15-2
END-CASE, 8-25
END-DO, 8-23
END-IF, 8-18
ENDPAGE procedure, 11-7, 11-63, 11-70
END-PROC, 10-1
ENVIRON option, 14-6
Environment, See PARM statement customizing, 4-1
definition, 3-1, 3-2, 3-5, 4-1, C-1
types, 2-1
ENVIRONMENT COBOL parameter, 14-6
JOB statement, 6-2, 6-4
PARM statement, 4-2, 4-9
Environment, options table parameters, C-1
WKDSNPF, 11-72
EOF, See FINISH
EOF (end-of-file), POINT statement, 9-9
EQ, 8-4
ERASE, 4-5
Error analysis reports, 2-2, 2-5, 15-16
Error condition handling, 14-14
Error messages
DQSCGEN, A-44
error analysis reports, 2-2, 2-5, 4-5
execution, 15-15
file status, 9-1
format, A-1
IDD interface, A-46
operating system, A-2
source program syntax, A-9
statement number, 4-9
subprogram exits, 14-14
syntax, 15-15
VSAM open error code, 5-7
Evaluation rules
field class conditions, 8-10
field series conditions, 8-8
EXECUTE, 8-32
Execution listing, 15-14
EXIT, CA-Earl, D-55
Exits, user
CA-Earl, D-55
FILE statement EXIT parameter, 5-3, 5-9
file-types, 5-5
input/output, 5-9, 14-1
passing parameters, 5-9
VSAM, 5-14
EXITSTR, 4-2, 4-9
installation option, C-7
EXTENDED parameter, FILE statement, 5-3, 5-13
Extended reporting options module, 15-3
EZTPAQT
default modname, C-2
EZTPI
FILE statement, C-21
RECORD, C-21
EZTPOPT environment options table, C-1
EZTPX01, 5-35, 15-2
EZTPX03, 15-3
EZTPX04, 15-3
EZTPX05, 15-3
EZTVFM
optional
OS/390 file, D-58
VSE files, D-68

Field
name prefixes, 5-34
redefine, 5-18, 5-26
Field bits condition, 8-2, 8-13
syntax, 8-13
Field class condition, 8-2, 8-9
Field definitions
alphanumeric, 5-19
assignment statement, 7-5
binary, 2-3, 5-19, 5-28
copying, 5-31
data reference rules, 5-15, 5-33, 5-34
decimal, 5-28, 5-29, 7-3
DEFINE, See DEFINE statement
edit masks, 2-3

F
example data descriptions, 5-25
field-names, 5-15, 5-34
indexing, 5-34
moving characters, 7-15
moving numeric values, 7-1
numeric, 7-3
overlay redefinition, 5-18, 5-26
packed decimal, 5-19, 5-28
qualifier, optional, 5-34
redefine, 5-18, 5-26
S storage fields, 11-62, 11-68
signed fields, 5-28
system-defined fields, 5-34, 6-15, 12-3
unsigned packed, 2-3, 5-20, 5-29
varying length fields, 5-26, 11-18
working storage, See working storage
XREF listing, 4-9
zoned decimal, 5-19, 5-28, 7-6

Field definitions, location
  * value, 5-17, 5-18, 5-25
defining, 5-18
redefining, 5-18, 5-26
Field relational condition, 8-2, 8-4
  syntax, 8-4
Field series condition, 8-2, 8-7
  syntax, 8-8
Field-names, 5-15, 5-16
Fields
  non-quantitative, 5-19, 5-29
  quantitative, 5-19, 5-21, 5-28, 11-43
  signed, 5-29, 11-43
  unsigned, 5-29
File maintenance, 2-4
File mode, CMS minidisk, C-3
File presence condition, 8-2, 8-14
  syntax, 8-14, 8-16
File presence series condition, 8-2, 8-16
File processing
  automatic, See automatic file input
  control breaks, 2-4
  controlled, See controlled file input
data access modes, 12-2
data availability tests, 12-4
error analysis report, 2-2
error conditions, 12-4
FILE-STATUS, 9-1, 12-3
ISAM files, 12-8
keyed, 12-15
match/merge, 2-2
record address, 12-3
RECORD-COUNT, 12-3
RECORD-LENGTH, 12-3
SAM files, 12-5
synchronized, 2-2
VSAM files, 12-10
File records
  fixed length, 12-2
  undefined-length, 12-2
  variable length, 12-2
FILE statement, 12-16, C-3
coding, 5-3
creating files, 3-5
DEVICE, C-5
device types, 5-9
DLI, 5-3, 5-4
EXIT, 5-3, 5-9, 14-1, 14-8, 14-10
file-type, 5-3, 5-5
IDMS interface, 5-3, 5-6
modify, 5-14
MODIFY, 5-9, 14-9, 14-10
overlay field redefinition, 5-26
parameters, 5-4
PUNCH files (VSE), 12-8
summary file processing, 11-59
syntax, 5-3, 5-4
TABLE, 5-13, C-20
TAPE, C-13
VSAM, 12-10
work files, 11-72
WORKAREA, 5-12
FILE Statement, 5-3
FILE statement, CARD option
  END delimiter, 5-10
FILE statement, file-names
  rules for, 5-4, 5-34
  XREF listing, 4-9
FILE statistics, LIST, C-9
File status report, 2-2
Files
  summary, generating, 11-59
  working storage, see working storage, 11-72
FILE-STATUS, 9-1, 12-3
FILL, MOVE statement parameter, 7-15, 7-17
FINAL, REPORT control statement, 11-25
FINISH/START, 6-2, 6-3, 10-4
FIRST-DUP, 8-17
FLDCHK, 4-5, 4-8
installation option, C-7
FLOW, 4-5, 4-9
installation option, C-8
table, 2-5
FLOWSIZ, installation option, C-8
Font definition, 5-22, 11-28
LINE statement, 11-31
Format and conversion rules, 3-18
DBCS, 3-18
Format relationship rules, DBCS, 3-20
Format, data, 5-19
Forms, See output specification
FORTRAN, links, 14-3, 14-7
Free storage, C-16
FULLTRK, 5-12, 12-5

G
GE, 8-4
GET statement, 6-15, 9-1, 9-7, 9-8, 12-1
GETDATE macro, D-11
GETVIS space, C-7
GOTO JOB, 8-28
GOTO label, 8-27
GOTO statement, 8-27
GQ, 8-4
GR, 8-4
GT, 8-4

H
Header shift code, 3-8
HEADING
(DEFINE statement), 5-17, 5-22
statement, 11-29, 11-34
Hexadecimal
arithmetic fields, 7-8
DISPLAY HEX, 9-5, 11-60
edit mask, 2-3, 5-24
hex dump, 2-4
literals, use of, 3-14
padding fields, 7-3
HIGHEST-BREAK, 2-6, 8-10, 11-45
Hiragana
conversion literals, 3-17
Hirigana
characters, 3-17
Host disk files, 5-7, 5-8, 12-22
I
I/O buffer, C-3
IBM sort options, DBCS, 6-9, 11-20
ICCF, 2-1
IDD
IDMS, C-21
RECORD, C-21
IDD interface, C-8
messages, A-46
processing MOVE LIKE statement, 7-18
IDDEXIT, installation option, C-8
IDMS
IDD, C-21
RECORD, C-21
IDMS interface
RESET to zero option, 5-6
subschema-name, 5-3, 5-6
IF statement, 8-18
nested, 2-3
program control, 3-5
synchronized file processing, 12-19
syntax and logic, 8-19
usage, 8-20
IMS/DLI interface
- automatic file input, 6-2
- COPY statement, 5-32
- EXIT option, 5-9
- file-names, 5-4
- FILE-STATUS field, 12-3
- file-types, 5-5
- processing, 5-9, 12-1
- program control block, 5-5
- program specification block, 5-5
- RESET to zero option, 5-6

In-core binary search, 2-2
INDEX, 5-17, 5-22
Indexing, 5-34
Input specification
- automatic, See automatic file input
- controlled, See controlled file input
- DISPLAY, 9-2
- EXIT, 5-9
- FILE statement, 5-3
- file-types, 5-5
- GET, 9-7
- MODIFY, 5-9
- POINT, 9-8
- STATUS, 9-1

INSTREAM, See tables and arrays
Instream macros, 16-8
INTEGER (assignment statement), 7-7
Integrated data dictionary, See IDD interface
Internal work file, prefix, C-22
ISAM files, 12-8
- automatic processing, 12-8
- controlled processing, 12-8
- EXIT option, 5-9
- FILE-STATUS, 9-1
- FILE-STATUS field, 12-3
- file-types, 5-5
- processing, 12-1
- random access, 9-12
- random processing, 12-8
- READ, 9-12
- record format, 5-11
- retrieval with POINT, 9-8
- skip-sequential processing, 12-8
ISRT, C-20

J

JCL
- examples, D-63
- file OPEN errors, 12-4
- JOB processing, 6-1
- PARM, 5-35, 15-2
- PARM statement interrogation, 15-2
- VSAM file processing, 12-10

JEF sort options, DBCS, 6-12

JOB activities, 12-18
- automatic input, 6-2
- coding, 6-1
- DISPLAY, 9-6
- execution, 6-1
- function, 3-2, 3-5, 6-1
- INPUT, 6-2
- JOB procedures, 3-2, 6-2
- labels, 3-13
- NAME, 6-4
- opening files, 12-5
- processing logic, 6-5, 6-6
- report input modification, 11-65
- STOP statement, 6-7
- summary file processing, 11-59

JOB INPUT statement, 12-21

JOB statement, 6-1, 12-17
- coding, 3-5
- FINISH, 10-3, 10-4
- function, 6-1
- KEY, 6-3
- printing files, 11-3
- sorting, effect on input, 6-2
- START, 10-3, 10-4
- START/FINISH, 6-3
- syntax, 6-2

K

KANJI parameter, 5-23
KANJI-DATE, 5-35
KANJI-TIME, 5-35
Katakana
- characters, 3-16
- conversion literals, 3-16
Keyed file processing, 12-15
Keywords, 3-12, B-1

L

Label reports, 11-40
Labels, C-2
   mailing, 2-4, 11-5, 11-11, 11-40, D-19
   statement, 3-13
LABELS
   ACROSS, C-2
   DOWN, C-6
   NEWPAGE, C-11
LABELS parameter of REPORT statement, C-6, C-8
LABLSIZ, installation option, C-8
LAST-DUP, 8-17
LE, 8-4
Length
   compiler output lines, C-8
   report line, C-8
LEVEL, control break, 11-44, 11-61
Library
   DEFINE, 5-1
   fields, See field definitions
   files, See FILE statement
   function, 3-2, 5-1
   password, 16-5
   program placement, 3-1, 3-5
   secured, 16-5
Line separation, SPREAD, C-17
LINE statement, 11-30
   spacing, C-17
LINE-COUNT, 5-36
LINESIZ, 5-10
   installation option, C-8
LINESIZE, 11-12, 11-14, 11-30, 11-36, C-8
LINK, 4-1, 4-2, 4-10
Linkage conventions, 14-3
LIST, 4-1, 4-2, 4-10
   FILE, C-9
   installation option, C-9
   parameter of PARM statement, C-9
   PARM, C-9
   statement, 15-6
Literals, use of, 3-12, 3-14
   alphabetic conversion, 3-15
   DBCS, 3-14
   DBCS HEX, 3-15
   EBCDIC, 3-13
   Hiragana Conversion, 3-17
   katakana conversion, 3-16
   mixed, 3-15
   numeric, 3-13
   user converted, 3-17
LOADing subprograms, 14-2
LONG option of XREF, C-23
LONGDATE, 2-10, 11-16, 11-26
LONGDTE value, options table, 2-10
LQ, 8-4
LS, 8-4
LT, 8-4

M

MAC#LIB, installation option, C-9
MACDDN, installation option, C-9
MACDEV, C-5
   3330, C-10
   3340, C-10
   3350, C-10
   3375, C-10
   3380, C-10
   3390, C-10
   DISK, C-10
   FBA, C-10
   installation option, C-9
MACDEV, installation option, C-10
Macro
   definition, 16-5
   invocation statement, 16-1
   library, 16-3, C-9
   library support, C-10
   processing, 16-10
MACRO
  LIBR, C-10
  NO, C-10
  PAN, C-10
  PDS, C-10
  USER, C-10
  VSAM, C-10
MACRO, installation option, C-10
macros
  & prefix, 16-11
Macros
  examples, D-2, D-3
  GETDATE macro, D-11
  instream, 16-9
  invoking, 16-2, 16-11
  libraries, 16-3, D-66, D-75
  names, 16-5
  options table, 16-3
  parameter substitution, 16-11
  protection, 16-3, 16-5
  syntax, 16-5, 16-6
  user supplied, 16-3, 16-5
MACSYS#, installation option, C-11
Mailing labels, C-2
Mapping, 2-6
MASK, See edit masks
MASTER
  optional
    OS/390 file, D-58
    VSE files, D-68
Match/merge
  operations, 12-16
  processing, 2-2
MATCHED test, 12-19
MEMORY, 4-5
MEND macro statement, 16-8, 16-9
MIXED fields, conditional expressions, 8-3
Mixed subjects, 8-5
Modes of operation, 3-1
MODIFY, 5-3, 5-9
MONEY, installation option, C-11
MOVE LIKE statement, 7-17
  function, 7-1
  syntax, 7-17
MOVE statement, 7-15
  function, 7-1, 9-12
  syntax, 7-15
Moving numbers, See assignment statement
MSG, 4-5, 4-12
MSTART macro statement, 16-9
Multiple dimension arrays, 13-7
Multiplication, arithmetic, 3-21, 7-3
N
NE, 8-4
Nesting DO loops, 8-23
NEWPAGE
  installation option, C-11
  statement, 15-6
NOCLIST, installation option, C-3
NODMAP, installation option, C-6
NODUPS parameter of RETRIEVE statement, C-3
NOFLDCHK, installation option, C-7
NOFLOW, installation option, C-8
Non-quantitative fields, 5-19, 5-28
NOPMAP, installation option, C-13
NOPPRINT, 11-25
NORWD, 5-3
NOT, 8-17
NOVERIFY, 5-3
NOXREF, installation option, C-23
NQ, 8-4
Numeric
  edit characters, C-11
  subjects, 8-5
NUMERIC
   installation option, C-11, C-12
   test, 2-3, 5-29, 8-9, 8-11
   test range, 8-8
Numeric data
   decimals, 5-20, 5-23, 5-28, 7-3
   literals, 3-14
   negative numbers, 5-31
Numeric literals, 3-14
   HEX, 3-14
NUMWORK, 6-14, C-12

O

Object of comparison, 8-5
OCCURS, 5-17, 5-24, 5-34
One-dimension arrays, 13-5
Operating environments
   DOS/VSE, 2-1
   modifying (PARM statement), 4-1
   OS/VS, 2-1
   VM/CMS, 2-1
Operating system, host, C-19
Operational diagnostic messages, A-2
Options module
   DBCS, 15-3
   extended reporting, 15-3
Options table
   modifying with PARM statement, 4-1
   parameters, C-1
   regeneration, 15-3
OR, combined conditions, 8-4
OS/390, C-19
OS/VS environment
   macro storage, 16-3, 16-5
   output, 2-1
   subprogram LOADING, 14-2
OTHERWISE, 8-26
Output specification
   blocksize, 5-12
   default, 11-73
   DISPLAY, 5-10, 9-2
   EXIT, 5-9, 14-8
   file-types, 5-5
   JOB, 6-1
   PRINT, 9-10
   PRINTER, 5-10
   PUNCH, 5-10
   PUT, 9-11
   REPORT statement, 5-10, 11-73
   sequential file output, 9-11
   sorted records, 6-8, 6-15
   statements, 9-1
   STATUS, 9-1
   subprogram exits, 14-1
   terminal, 2-1
Overlay field redefinition, 5-18, 5-26

P

P - packed decimal data, 5-28
   even number of digits, 5-21
PAGE, spelling, C-12
PAGE-COUNT, 5-36
PAGESIZ installation option, C-12
PAGESIZE, C-12
PAGEWRD, installation option, C-12
PAN, PANMODI, C-10
PANDD, D-66, D-75
PANDDI, D-66, D-75
   optional VSE files, D-68
   OS/390 file, D-58
PANMODI, C-10
Parameter lists, 14-8
Parenthesis (), use of, 3-12, 3-21
PARM
   JCL, 5-35, 15-2
   LIST statistics, C-9
PARM statement, 4-1
   ABEXIT, C-1
   coding, 3-2, 4-1
   DB2SSID, C-18
   DEBUG, C-3, C-6, C-8, C-13, C-18, C-23
   environment options table, C-1
   examples, 4-17
   EXITSTR, C-7
function, 3-1, 4-1, C-1
LINK, 6-4
LIST, C-9
parameters, 4-5
PREPNAME, 4-11, C-13
PRESIZE, C-13
SORT, C-14, C-15, C-16
SORTRLS, C-16
SSID, 4-15
syntax, 4-2
USERID, 4-16
VFM, C-22
WORK, C-12
PARM, LIST, 15-8
PARM-REGISTER, 5-35
Passed parameter list, 5-35
Password (VSAM files), 16-5
PDS macro library, C-9
Percent, calculating, 11-62, 11-68
PERFORM statement, 2-3, 8-31, 10-3
syntax, 8-31
Period (.), use of, 3-12
PL/I, links, 14-3, 14-7
PLAN (SQL parameter), 4-10
PLANOPTS, 4-10
parameter, 2-8, 4-10
PMAP, 2-5, 4-5, 4-8, 15-13
installation option, C-13
POINT statement, 6-15, 9-1, 9-8, 12-1, 12-2, 12-12, 12-16
POP
facility, 2-6
statement, 15-7
Prefix, internal work file, C-22
PREPNAME (PARM statement parameter), 4-11
PREPNME, installation option, C-13
PRESIZE, 4-2, 4-5, 4-11
installation option, C-13
parameter of PARM statement, C-13
PRIMARY, 8-17
Print positions, C-8
additional, C-19
PRINT statement, 9-1
function, 11-1, 11-2
labels, 11-5
program placement, 3-5
report input modification, 11-65
syntax, 9-10
work files, 11-2, 11-3, 11-7, 11-72
PRINT Statement, 9-10
PRINTER, C-14
option, 5-3, 5-10
Printer, extended reporting, 5-3, 5-13
Procedure, See PROCs
placement, 10-5
START, 12-16
syntax, 10-1
Proc-name, 8-31
PROCs (procedures)
coding rules, 3-2, 10-5, 10-6
function, 3-2
invoking, 10-3
labels, 3-13
loading user exit programs, 12-5
prescreening, 6-14
special-name procedures, 11-7, 11-63
START, JOB, 6-3
syntax, 10-1
XREF listing, 4-9
PROCs (procedures)
BEFORE, SORT, 6-7, 6-14, 6-18
Program compilation statistics, update dictionary, C-20
Program structure, 3-1
Programming languages, 14-1
Programs
activity section, 3-1, 3-5
data reference rules, 5-33
environment definition, 3-1, 3-5, 4-1
invoking PROCs, 10-3
invoking user-written, 10-3
library definition, 3-1, 3-5
mapping, 2-6
Programs, statements
labels, 3-13
order of coding, 3-2, 3-5
syntax rules, 3-7
Publications
CA-Easytrieve/Plus, 1-3
related, 1-4

PUNCH files, 5-3, 5-10, 12-3, 12-8

PUSH
facility, 2-6
statement, 15-7

PUT statement, 3-5, 6-15, 9-1, 9-11, 12-1, 12-2, 12-5, 12-7, 12-10, 12-13

Q

Qualification of fields, 5-34
Quantitative fields, 5-19, 5-21, 5-28, 11-43

R

Range, numeric test, 8-8
READ statement, 6-15, 9-1, 9-12, 12-1, 12-12
REAL, 4-5, 4-13
Record key, 12-21
Record length compiler work file, C-13
Record relational condition, 8-2, 8-17
syntax, 8-17
Record-control-word, 12-3
RECORD-COUNT, 12-3, 15-14
RECORD-LENGTH, 6-15, 12-3
Records
duplicate keys, 2-2
formats, 5-3, 5-11
layout examples, 5-25
matching, 2-2
modification at exits, 5-9
multiple statements, 3-10
output, 5-10
SCANCOL option, 3-7
sorting, 6-13, 6-14, 6-18
Recreating the options table, 15-3
Redefine fields, 5-18, 5-26
Relational operators, 8-5
RELEASE, 4-5, 4-13
REPL, C-20
Report
definition statements, 11-18
line length, C-8
procedures, 11-60
work files, 11-72
Report activities
report procedures, 9-6, 11-63
REPORT activities
JOB, 6-1
PROC coding, 3-2, 10-5
report procedures, 11-60
REPORT procedures, 2-4, 3-2, 12-1
REPORT-INPUT procedure, 11-65
special-name procedures, 10-3, 10-5
REPORT option
LONGDATE=xxx, C-9
REPORT statement, 11-7
coding, 3-5
CONTROL, 11-7, 11-18, 11-24, 11-41, 11-42
cross-reference listing of report-names, 4-9
DLTCOPY, 11-52
DTLCTL, 11-10, 11-42, 11-47, C-7
FILE, 11-72
HEADING, 11-7, 11-18, 11-29, 11-34
LABELS, C-2, C-6, C-11
LEVEL, 11-44, 11-61, 11-69
LINE, 11-7, 11-18, 11-30
LINESIZ, C-8
LINESIZE, 11-12, 11-14, 11-30, 11-32, 11-36, C-8
LONGDATE, 11-26
LONGDATE option displays SYSDATE-LONG, 11-16
PAGESIZE, C-12
PRINT, 9-10, 11-72
printer files, 5-10
PRINTER output, 11-8, 11-12
SEQUENCE, 11-7, 11-18, 11-41, 11-72
SHORTDATE, 11-26
SHORTDATE option displays SYSDATE, 11-16
SKIP, C-14
SPACE, C-16
special-name procedures, 11-7, 11-63
SPREAD, 11-39, C-17
SUM, 11-7, 11-18, 11-25
SUMCTL, 11-50, C-19
SUMFILE, 11-59, 11-61
SUMMARY, 11-42, 11-47
SUMSPACE, 11-58, C-19
syntax, 11-8
TAG, 11-51
TALLY, 11-10, 11-43, 11-59, 11-68
TALLYSIZE, C-19
testing aid parameters, 11-8
TITLE, 11-7, 11-18, 11-26, 11-32, 11-54
TITLESKIP, C-20
work files, 11-72
REPORT statement parameters, 11-9
REPORT-INPUT procedure, 11-7, 11-63, 11-65
Reports
alternate heading, 5-28
annotations, 11-61, 11-66, 11-69, 11-70, 11-71
column headings, 11-29, 11-34
content, 11-18
control breaks, 2-4, 11-24, 11-42, 11-59, 11-61, 11-62
control field titles, 11-54
control field totals, 11-42, 11-56
control report contents, 11-46
DBCS restrictions, 11-6
decimal positioning, 11-37
elements, D-1
file status report, 2-2
footers, 11-4, 11-70, 11-71
formatting, 2-4, 11-4, 11-7, 11-8, 11-18, 11-32
generating, 11-1
IRS forms, 11-38
line item positioning, 11-35
linesize, 11-4
mailing labels, 11-40
modifying data, 11-65
multiple, 2-5
output, 2-4
pagesize, 11-4
pre-printed forms, 11-38, 11-40
printing control field, 11-10
printing zeros, 5-30
renumbering pages, 11-24
sequenced, 2-5
subtotals, 2-4
summary files, 2-4, 11-9, 11-47, 11-59
TALLY, 11-10, 11-43, 11-59, 11-68, D-21, D-27
testing aids, 11-8, 11-17
titles, 11-4, 11-26, 11-32, 11-54
totals, quantitative field, 11-25
work files, 11-72
Reserved words, B-3
RESET, 5-3, 12-10
(DEFINE statement), 5-24
RESTART, JOB statement, 6-4
RESTARTABLE, PARM statement, 4-11
RETAIN, 5-5, 5-14
Retrieve a record, version number, C-21
RETRIEVE statement
DUPS, C-3
NODUPS, C-3
RETURN-CODE, 5-35
RETURNS parameter, CALL statement, 14-12
Revisions, 2-6
REWIND, 5-3
installation option, C-13
Rounded, 7-7
Routing
printed output, 11-73
sort messages, C-14
S
S working storage fields, 5-18, 11-62, 11-68
SAM files, 12-5
automatic processing, 12-5
controlled processing, 12-5
defining, 5-14
device types, 5-9
file-types, 5-5
processing, 12-1
PUNCH files, 12-8
PUT records, 9-11, 12-7
record format, 5-11
work file contents, saving, 11-73
SAMV, C-22
SCANCOL, 3-7
installation option, C-13
Schema, VERSCHM, C-22
SD type work data sets, C-16
SEARCH statement, 13-3
syntax, 13-3
table access, 13-3
SECONDARY, 8-17
Secured libraries, 16-5
Security, 2-6, 16-3, 16-5
Segmented data, 13-13
SELECT statement, 6-18, 11-65, 12-16
  sorting, BEFORE, 6-18
Selectable options, C-1
SEQUENCE statement, 11-18, 11-41
Sequenced reports, 11-41
Shift code systems, 3-8
  header, 3-8
  wrapping, 3-8
Shift in, 3-8
Shift out, 3-8
SHORT option of XREF, C-23
SHORTDATE, 2-10, 11-16, 11-26
Signed fields, 5-29, 11-43
Single dimension arrays, 13-5
Single file keyed processing, 12-21
SINXIT, installation option, C-14
SKIP
  installation option, C-14
  parameter of REPORT statement, C-14
  statement, 15-6
Skip to channel 1, C-11
Slashes, removing from SYSDATE, D-11
Sort
  work areas used, C-12
  work data set logical assignments, C-16
SORT
  ALL, C-14
  CONSOLE, C-14
  CRITICAL, C-14
  DDname messages, C-15
  DEFAULT, C-14
  DIAG, C-15
  ERASE, C-15
  NO, C-14
  NODIAG, C-15
  NOERASE, C-15
  NOTP, C-15
  parameter of PARM statement, C-2, C-14, C-15, C-16
  PRINTER, C-14
  REAL, C-15
  routing messages, C-14
  SORTSIZE, C-16
  SORTWK#, C-16
  SORTWRK, C-16
  TP, C-15
  VIRTUAL, C-15
SORT activities
  function, 3-2, 3-5, 6-1
  KEY sequencing, 6-14
  opening files, 12-5
  prescreening, 6-14
  record length (F) parameter, 5-6
  selective sorting, 6-18
  sort procedures, 3-2, 6-7, 6-15, 12-1
  subprogram loading, 14-2
SORT interface, 3-2, 4-2, 4-5, 4-11, 4-13, 6-7
Sort program
  amount of storage, C-16
  name, C-15
  setting of options, C-15
SORT statement, 6-7
  BEFORE, 10-3, 10-4
  coding, 3-5
  function, 6-7
  messages, printing, 4-12
  syntax, 6-7
  USING, 6-7, 6-8
  WORK, 6-7, 6-14, C-12
  work data sets, 6-14
SORTMSG, installation option, C-14
SORTMSR, installation option, C-14
SORTNAM, installation option, C-15
SORTOPT, installation option, C-15
SORTPRT, installation option, C-15
SORTRLS, installation option, C-16
SORSIZE, installation option, C-16
SORSZ, MAX, C-16
SORTWK#, installation option, C-16
SORTWKn, VSE file, D-69
SORTWKnn, OS/390 file, D-59
Sortwork data set, C-16
SORTWRK, installation option, C-16
Source statement libraries, 16-3
SPACE parameter of REPORT statement, C-16
SPACE test, 8-9, 8-11
SPACE, installation option, C-16
Special tests, 2-3
Special-name report procedures, 10-5
SPREAD
   installation option, C-17
   parameter of REPORT statement, C-17
SPRTXIT, installation option, C-17
SQL
   access module, C-13
   DB2, C-18
   parameter of SSID, 4-15
   parameters, 4-10
   subsystem, C-18
   supplemental diagnostic messages, A-48
SQLSYNTAX parameter, 2-8, 4-14
SSID
   (PARAM statement parameter), 4-15
   installation option, C-18
   SQL parameter, 4-15
Standard reports, 11-32
START procedure, 12-16
START/FINISH
   JOB, 6-2
   parameters (JOB), 10-4
Starting SYS number, C-11
STATE, installation parameter, C-18
Statement label, 8-28
   GOTO statement, 8-30
Statement label syntax, 8-29
Statement number, C-18
statements
   labels, 4-9
Statements
   comment, 3-11
   continuation (- or +), 3-11
   delimiters, 3-12, 3-21
   labels, 3-13
   multiple, coding, 3-10
   source, 3-7
   syntax, 3-7
Static working storage, 11-62
STATUS, 9-1
STEPLIB, OS/390 file, D-60
STOP in a SORT activity, 8-32
STOP statement, 8-32
   function, 6-7
   report proc warning, 11-60
   syntax, 8-32
Storage
   maximum amount, C-19
   sort program, C-16
   VFM buffer pool, C-22
Storage management, 14-3
STORMAX, installation option, C-19
Subject element, 3-18, 3-20
Subject of comparison, 8-4
Subprograms
   EZTPX01, 5-35, 15-2
   linking, 14-3
   loading, 14-1
Subschema, VERSCHM, C-22
Subscripts, 13-10
Subtraction, 3-21, 7-3
SUM statement, 11-25
SUMCTL
   ALL, C-19
   HIAR, C-19
   installation option, C-19
   NONE, C-19
   parameter of REPORT statement, C-19
   TAG, C-19
Summary files, 11-59
Summary lines, annotations of, C-19
SUMSPACE
   installation option, C-19
   parameter of REPORT statement, C-19
Symbol references, B-3
Synchronized file input process, 12-18
Synchronized file processing, 2-2, 12-15
   record availability, 12-17
   special IF statements, 12-19
   updating a master file, 12-20
SYNTAX, 4-1, 4-5, 4-16
Syntax diagnostic messages, A-9
Syntax rules, 3-7
SYS, 4-5, 4-13
SYS number, starting, C-11
SYSDATE, 2-10, 5-35, 6-15, 11-16, 11-33
   GETDATE macro, D-11
   LONGDATE=xxx, C-9
   removing slashes, D-11
SYSDATE-LONG
   four-digit year, E-1
   LONGDATE=xxx, C-9
SYSIN, 5-10, 12-6
   OS/390 file, D-57
SYSIN/SYSIPT exit, C-14
SYSIPT (VSE), 5-5, 5-10, 12-6
   files, D-67
SYSLIN, OS/390 file, D-59
SYSLIST/SYSPRINT exit routine, C-17
SYSLNK, VSE file, D-69
SYSLST (VSE), 5-5, 11-73
   file, D-67
SYSOUT, C-15
   OS/390 file, D-59
SYSPCH (VSE), 5-5
SYSPRINT, 9-2, 11-73
   OS/390 file, D-57
SYSPRINT/SYSLIST exit routine, C-17
SYSSNAP, OS/390 file, D-61
System defined file fields, 5-35
System Management Storage (SMS), 5-12
SYSTEM, installation option, C-19
System-defined
   fields, 5-34
   file fields, 12-3
SYSTIME, 5-35
SYSUDUMP, OS/390 file, D-61

Table
   definition, 13-1
   maximum number of entries, C-20
TABLE, See tables and arrays
Tables and arrays, 13-1
   data string evaluation, 13-15
   edit masks, 5-24
   external tables, 13-2
   fields, 13-2
   file SEARCH method, 13-2
   INDEX method, 5-22, 13-5
   instream tables, 13-2
   multiple dimension arrays, 13-7
   one-dimension arrays, 13-5
   SEARCH statement, 13-3
   sorting tables, 13-2
   TABLE option of FILE, 13-4
Tables and arrays, file SEARCH method
   instream tables, 5-13
   TABLE option of FILE, 5-13
Tables and arrays, INDEX method
   OCCURS option of DEFINE, 5-24
TALLY, 11-10, 11-43, 11-59, 11-68, D-21, D-27
   field size, C-19
TALLYSIZE parameter of REPORT statement, C-19
TALYSIZ, installation option, C-19
TAPE
   NORWD, C-13
   UNLOAD, C-13
   YES, C-13
TAPE (VSE) option, 5-3, 5-10
TAPE parameter of FILE statement, C-13
TBLMAX, installation option, C-20
TERMINATION procedure, 11-7, 11-63, 11-71
Test
   alphabetic characters, 8-10
   CONTROL break, 8-10
   numeric characters, 8-11
   spaces, 8-11
   zero characters, 8-11
Test under mask, 2-3
Tests, special
  field content, 2-3
  test under mask, 2-3
  zero, 2-3
Test valid DBCS characters, 8-11
TIME, See SYSTIME
TITLE statement, 11-26
  LONGDATE, 11-16, 11-26
  SHORTDATE, 11-16, 11-26
  spacing of, C-16
TITLESKIP
  installation option, C-20
  parameter of REPORT statement, C-20
Top of form, C-11
TP, 4-5, 4-13
Trace entries available, C-8
TRUNCATED (assignment statement), 7-7
TSO, 2-1

U
UNLOAD, 5-3
Unsigned fields, 5-29
UNTIL, 8-22
UPDATE, 5-3, 12-13
Update functions, C-20
Updating
  files, 5-6
    master file, 12-20
UPDTDLI, installation option, C-20
UPDTI0D, installation option, C-20
UPDTIDM, installation option, C-20
UPDTIVS, installation option, C-21
User converted literals, 3-17
userfiles
  OS/390 files, D-61
  VSE files, D-69
USERID (PARAM statement parameter), 4-16
USERSM, installation option, C-21
User-written programs, See PROCs (procedures)
USING parameter, 5-3
Utility programs, 15-2
  EZTPX01, 15-2
  EZTPX03, 15-3
  EZTPX04, 15-3
  EZTPX05, 15-3

V
V command codes, C-6
VALUE, 5-17, 5-24
Variable length records, 12-2, 12-7
VARYING (DEFINE statement), 5-26, 11-18
Varying length field, 5-26, 11-18
  condition, 8-3
VARYING, field length, 5-21
VERFILE, installation option, C-21
VERPGM, installation option, C-21
VERREC, installation option, C-21
VERSCHM, installation option, C-22
Version number
  non-data base file, C-21
  program accessing dictionary, C-21
  record, C-21
VFM, See virtual file manager
  3330, C-22
  3340, C-22
  3350, C-22
  3375, C-22
  3380, C-22
  3390, C-22
  buffer pool, C-22
  DISK, C-22
  FBA, C-22
  MEMORY, C-22
  overflow file, C-22
  parameter of PARM statement, C-22
VFMDEV, C-5
  installation option, C-22
VFMSPAC, C-22
installation option, C-22

VIRTUAL, 4-5, 4-13

Virtual file manager, 12-15
access method, 12-15
file-types, 5-5
function, 2-5
PUT records, 9-11
RETAIN, 5-14, 12-15
SAM files, 12-15
space management, 2-5
spill area, 2-5
VFM files, 12-15
VFM PARM parameter, 4-2, 4-5, 4-16
work files, 2-5, 4-16

VSAM
adding records to files, 12-13
automatic input, 12-2
automatic processing, 12-11
controlled processing, 12-12
CREATE, 5-6
deleting records, 12-14
exit processing, 5-14
FILE-STATUS field, 12-3
I/O exits, C-7
macro library, 16-3, 16-5, C-9
mass-sequential-insertion, 12-13
open error code, 5-7
PUT statement, 12-10
random access, 9-12
retrieval with POINT, 9-8
skip-sequential processing, 12-12
subprogram file exit loading, 14-2
update functions, C-21
updating records, 12-14

VSAM files, 12-10
adding records, 12-13
creating, 12-10
defining, 5-14
ESDS, 12-10
EXIT option, 5-9
file-types, 5-5
KSDS, 5-6, 9-8, 9-12, 9-14, 12-10, 12-12
loading, 5-3, 5-6
maintenance, 9-14
mass-sequential-insertion, 9-11
password, 5-3, 5-6
processing, 12-1, 12-3
PUT records to, 9-11
READ, 9-12
reloading, 5-6
static, 11-62
system work file, 11-72
VALUE, 5-24, 5-26
W fields, 5-18, 5-24
WKDSNPF, 5-4, 11-72
WORKAREA, 5-9, 5-12
zeros/blanks, 5-26

Working storage, temporary
creating, 2-2
virtual file manager, 2-5, 12-15
virtual files, 5-5

Wrapping shift code, 3-8
WRITE statement, 6-15, 9-1, 9-14, 12-1, 12-2, 12-13

X

X edit mask, 5-29
XREF, 4-9, 15-13
   installation option, C-23

Y

Year 2000 support, 2-10, E-1
   DATECONV, E-2
   DATEVAL, E-4

Z

Z edit mask, 5-29, 5-30, 5-31
Zero
   blank-when-zero, 5-17, 5-23
   leading zeros, print/replace/suppress, 5-30
   testing, 2-3, 9-1
   work data set indicator, 6-14
ZERO test, 8-9, 8-11