Advantage™ CA-Easytrieve® Plus Report Generator

Application Guide

6.4

Computer Associates®

Contents

Chapter 1: Overview

Chapter 2: Library

Chapter 3: Activity Definition

Chapter 4: Data Manipulation

Chapter 5: Decision and Branching Logic

Chapter 6: Input/Output Specification

Chapter 7: Report Processing

Chapter 8: File Processing

Chapter 9: Table Processing

Chapter 10: IMS/DLI Processing

Chapter 11: OS/390 and z/OS JCL

Chapter 12: VSE JCL

Chapter 13: Applications

Chapter 14: Basic Examples

Chapter 15: Advanced Techniques

Chapter 16: Bank System

Chapter 17: Project Management System

Appendix A: Table of Statements

Appendix B: Cross-References

Index

Chapter

1

Overview

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

This *Application Guide* is will help you generate reports and process files without extensive data processing training and experience. This guide covers a subset of CA-Easytrieve Plus statements. The statements are described briefly, along with the associated parameters.

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

This guide is written for the business-oriented professional. Using this guide enables you to manipulate files, and to design and print reports, without having to wait for available time from data processing personnel.

Topics

This guide contains information related to the following:

- Program data and task statements
- Assignment statement
- Decision and branching logic
- Automatic, controlled, and database input/output specifications
- Program-produced reports
- File processing to read, modify, delete or add new records
- Table processing with typical examples
- IMS/DLI facilities for information retrieval from databases
- OS/390, z/OS, and VSE JCL requirements

- Sample jobs that perform typical data processing functions
- Online and batch processing illustrations to demonstrate a variety of coding techniques
- Project management systems
- List and description of statements covered in this guide
- Cross references of statements to specific examples

Related Publications

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

- IBM IMS/DLI Applications Programming Manual

Program Capabilities

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

File Processing

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

Operations

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

Output

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

Application

CA-Easytrieve Plus is designed to make it easy for you to manipulate files and produce reports. It is suitable for beginners in data processing techniques because it is easy to learn.

The next exhibit presents a sample program that is used throughout the following chapters of this guide to demonstrate the use of CA-Easytrieve Plus statements. This sample program is contrived to exemplify a large selection of statements.

Sample Program

```
 1 PARM  DEBUG(FLOW FLDCHK)
 2 *
 3 FILE PERSNL FB(150 1800)
 4   NAME                     17  16   A
 5     LAST-NAME            NAME   8   A
 6   PAY-GROSS                94   4   P  2
 7   DEPT                     98   3   N
 8   DATE-OF-HIRE            136   6   N
 9     HIRE-MM   DATE-OF-HIRE      2   N
10     HIRE-DD   DATE-OF-HIRE +2   2   N
11     HIRE-YY   DATE-OF-HIRE +4   2   N
12   SALARY                    W   4   P  2
13   BONUS                     W   4   P  2
14   RAISE                     W   4   P  2
15   SERVICE                   W   2   N
16   CURR-DATE                 S   6   N
17     CURR-MM      CURR-DATE      2   N
18     CURR-DD      CURR-DATE +2   2   N
19     CURR-YY      CURR-DATE +4   2   N
20 *
21 FILE ERRPRINT PRINTER
22 *
23 JOB INPUT PERSNL
24   %GETDATE CURR-DATE
42   SALARY = PAY-GROSS * 52
43   PERFORM SERVICE-CALC
44   IF SERVICE LT 1
45     GO TO JOB
46   END-IF
47   PERFORM RAISE-CALC
48   BONUS = 0
49   IF SERVICE GT 14
50      PERFORM BONUS-CALC
51   END-IF
52   SALARY = SALARY + RAISE + BONUS
53   PRINT UPD-RPT
54 *
55   SERVICE-CALC. PROC
57     SERVICE = CURR-YY - HIRE-YY
58     IF CURR-MM < HIRE-MM
59       SERVICE = SERVICE - 1
60     END-IF
61     IF CURR-MM NE HIRE-MM
62       GOTO QUIT-SERV-CALC
63     END-IF
64     IF CURR-DD < HIRE-DD
65       SERVICE = SERVICE - 1
66     END-IF
67     QUIT-SERV-CALC
68   END-PROC
69 *
70   RAISE-CALC. PROC
72     IF DEPT LT 940
73       RAISE = SALARY * 0.1
74     ELSE
75       RAISE = SALARY * 0.15
76     END-IF
77   END-PROC
78 *
79   BONUS-CALC. PROC
81     IF SALARY GT 29999
82       DISPLAY ERRPRINT, LAST-NAME, +5, +
                    'INELIGIBLE FOR BONUS'
```

```
 83       GOTO QUIT-BONUS
 84     END-IF
 85     IF SERVICE GT 19
 86        BONUS = 2000
 87     ELSE
 88        BONUS = 1000
 89     END-IF
 90     PRINT BONUSRPT
 91     QUIT-BONUS
 92   END-PROC
 93 *
 94 REPORT UPD-RPT PAGESIZE 51  LINESIZE 63 NODATE NOPAGE
 95   SEQUENCE DEPT LAST-NAME
 96   CONTROL DEPT
 97   TITLE 1  'ANNUAL UPDATE REPORT - SALARIED EMPLOYEES'
 98   HEADING  LAST-NAME   'NAME'
 99   HEADING  SERVICE     'SERV'
100   LINE DEPT LAST-NAME SERVICE RAISE SALARY
101 *
102 REPORT BONUSRPT LINESIZE 60 NODATE NOPAGE
103   SEQUENCE DEPT LAST-NAME
105   TITLE 1  'ANNUAL BONUS REPORT - SENIOR EMPLOYEES'
106   LINE DEPT LAST-NAME SERVICE BONUS
107 *
```

The program illustrated in the above exhibit processes a Personnel Master File named PERSNL that contains the department numbers, names, salaries, and dates of hire of all employees in an imaginary company.

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

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

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

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

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

Sample Update Report

ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

DEPT	NAME	SERV	RAISE	SALARY
901	WALTERS	10	2,204.80	24,252.80
901			2,204.80	24,252.80
903	WIMN	30	1,942.72	23,369.92
903			1,942.72	23,369.92
911	ARNOLD	13	2,316.60	25,482.60
	GREEN	12	1,901.12	20,912.32
	HAFER	11	634.14	6,975.54
	ISAAC	16	1,630.72	18,937.92
	KRUSE	21	1,260.48	15,865.28
	LARSON	15	1,476.38	17,240.22
	POST	12	1,518.40	16,702.40
	POWELL	26	1,264.64	15,911.04
	REYNOLDS	20	905.58	11,961.38
	SMOTH	26	1,639.04	20,029.44
	STRIDE	13	2,009.28	22,102.08
	YOUNG	11	1,630.72	17,937.92
911			18,187.10	210,058.14
912	LOYAL	28	1,535.04	18,885.44
912			1,535.04	18,885.44
914	CROCI	17	1,955.20	22,507.20
	GRECO	18	5,220.80	57,428.80
	MANHART	16	1,792.96	20,722.56
	RYAN	11	2,075.84	22,834.24
	VETTER	31	1,452.67	17,979.39
914			12,497.47	141,472.19
915	CORNING	11	760.03	8,360.35
915			760.03	8,360.35
917	TALL	19	2,559.75	29,157.27
917			2,559.75	29,157.27

ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

DEPT	NAME	SERV	RAISE	SALARY
918	BRANDOW	09	4,184.12	46,025.40
	EPERT	11	1,614.08	17,754.88
918			5,798.20	63,780.28
919	DENNING	15	706.42	8,770.62
919			706.42	8,770.62
920	MILLER	07	1,630.72	17,937.92
920			1,630.72	17,937.92
921	HUSS	21	1,876.16	22,637.76
	PETRIK	21	1,148.16	14,629.76
	WARD	12	955.50	10,510.50
921			3,979.82	47,778.02
923	LACH	15	1,614.08	18,754.88
	THOMPSON	11	1,302.08	14,322.88
923			2,916.16	33,077.76
924	ROGERS	20	1,710.80	20,818.80
	ZOLTAN	13	650.00	7,150.00
924			2,360.80	27,968.80
931	FORREST	18	71.76	1,789.36
931			71.76	1,789.36
932	BYER	12	2,062.73	22,690.09
932			2,062.73	22,690.09
935	NAGLE	08	2,882.88	31,711.68
	OSMON	31	3,265.60	35,921.60
935			6,148.48	67,633.28
940	JONES	23	6,277.44	48,127.04
	KELLY	11	1,541.28	11,816.48
	PHILPS	08	1,975.42	15,144.94
	WEST	17	5,740.80	44,012.80
940			15,534.94	119,101.26

ANNUAL UPDATE REPORT - SALARIED EMPLOYEES

DEPT	NAME	SERV	RAISE	SALARY
942	JOHNSON	17	5,559.84	42,625.44
	MALLOW	22	2,202.72	18,887.52
942			7,762.56	61,512.96
943	BERG	26	5,921.76	45,400.16
	JUDAR	16	4,611.36	35,353.76
	MCMAHON	19	3,013.92	24,106.72
943			13,547.04	104,860.64
944	NORIDGE	13	2,527.20	19,375.20
	TALUS	15	3,594.24	28,555.84
944			6,121.44	47,931.04
			108,327.98	1080,388.14

Sample Bonus Report

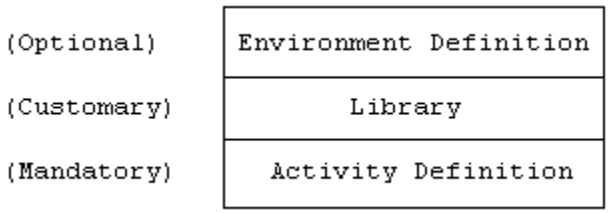
```
 ANNUAL BONUS REPORT - SENIOR EMPLOYEES


DEPT   LAST-NAME   SERVICE     BONUS

903    WIMN          30       2,000.00
911    ISAAC         16       1,000.00
911    KRUSE         21       2,000.00
911    LARSON        15       1,000.00
911    POWELL        26       2,000.00
911    REYNOLDS      20       2,000.00
911    SMOTH         26       2,000.00
912    LOYAL         28       2,000.00
914    CROCI         17       1,000.00
914    MANHART       16       1,000.00
914    VETTER        31       2,000.00
917    TALL          19       1,000.00
919    DENNING       15       1,000.00
921    HUSS          21       2,000.00
921    PETRIK        21       2,000.00
923    LACH          15       1,000.00
924    ROGERS        20       2,000.00
931    FORREST       18       1,000.00
942    MALLOW        22       2,000.00
943    MCMAHON       19       1,000.00
944    TALUS         15       1,000.00
```

Sample Error File Printout

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

Structure

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



Environment Definition Section

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

Library Section

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

Activity Definition Section

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

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

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

A JOB activity is composed of:

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

A procedure is composed of:

- A PROC statement
- One or more CA-Easytrieve Plus statements
- An END-PROC statement.

A report subactivity is composed of:

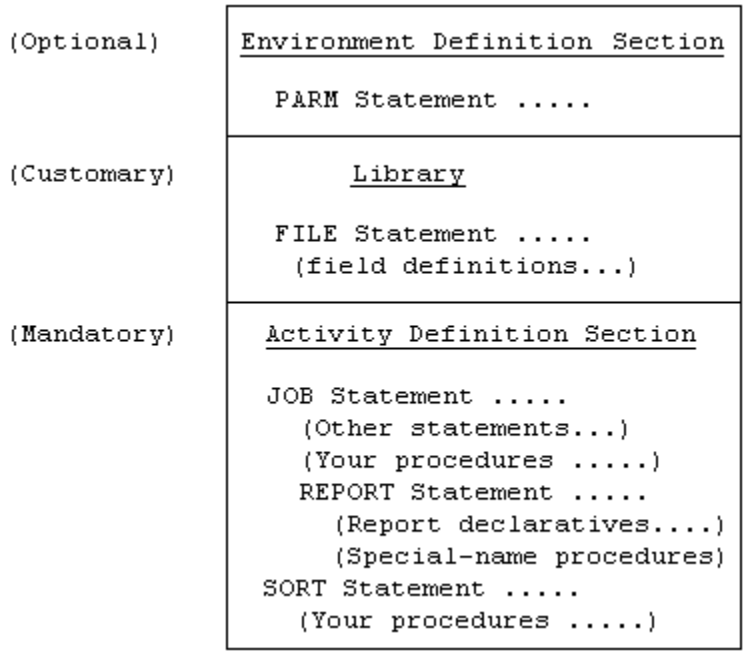
- A REPORT statement
- One or more report declaratives
- Report procedures (optional).

A SORT activity is composed of:

- A SORT statement
- Sort procedures (optional).

Procedures are discussed in the "Decision and Branching Logic" chapter. The REPORT statement and associated declaratives and procedures are described in the "Report Processing" chapter.

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



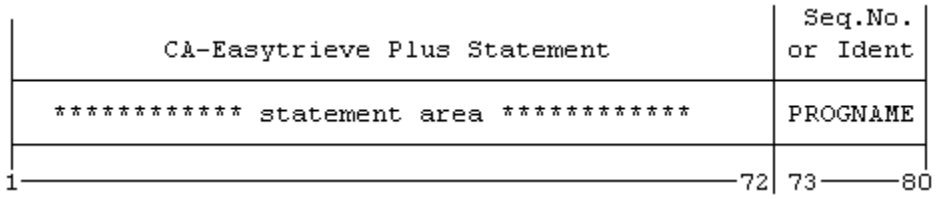
Rules of Syntax

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

Note: We recommend that you code your CA-Easytrieve Plus source programs in uppercase only. Lowercase keywords are not recognized by the compiler.

Statement Structure

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



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

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

```
A = 7. Y = 5. Z = X
```

Continued statements are discussed later in this section.

Words

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

Word Delimiter	Description
	Space
(	Left parenthesis
)	Right parenthesis
'	Apostrophe
.	Period

Word Delimiter	Description
,	Comma
:	Colon

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

Keywords

Keywords are words with specific meanings to CA-Easytrieve Plus. Some keywords are reserved for the use of CA-Easytrieve Plus only; the nonreserved words can be used as data names in the appropriate context. The "Cross-References" appendix lists all keywords and identifies those that are reserved.

Field Names

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

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

The first character of a field name must be an alphabetic character or a decimal digit. In addition, a field name must contain at least one alphabetic or special character to distinguish the field name from a number. All working storage field names and all field names within a single file must be unique. If you use the same field name in more than one file or working storage field, you must qualify the field name with the file name or the word WORK.

A qualified field name consists of the qualifying word followed by a colon and the field name. You can use any number of spaces, or no spaces, to separate the colon from either the qualifying word or the field name.

For example:

```
PERSNL: SALARY
WORK:SALARY
FILEX : SALARY
```

Valid Field Names

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

Invalid Field Names	Reason
SOCIAL SECURITY NUMBER	Embedded spaces
EMP'L-NO	Apostrophe not allowed

```
$AMOUNT                 Must begin with a letter
```

Literals

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

```
'Judge O''Connor'
```

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

```
'709 ENTERPRISE DR., OAK BROOK, ILL 60521'
```

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

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

```
1126
+112632
-11.2632
```

Comments

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

Continuations

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

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

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

Environment Definition

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

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

PARM Statement

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

The two most often used are:

- SYNTAX - Syntax check source statements
- COMPILE - Syntax check and compile source statements.

SYNTAX Parameter

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

COMPILE Parameter

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

If you do not use the PARM statement, the default is syntax check, compile, and execute.

Chapter 2

Library

The library section of your program describes the information that your program processes. This description is in terms of files, records, and fields.

File

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

Record

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

Field

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

The library section of your program provides:

- A general description of the groups of data (files) on which your program is to operate (through the FILE statement).
- A specific description of the individual items of data (fields) within each record of the files or within working storage (through the DEFINE statement).

The next exhibit illustrates the library section of the Sample Program depicted in the "Overview" chapter under the topic Application.

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

FILE Statement

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

Syntax

```
       FILE file-name  +

            [SYSxxx]    +
            [                                                    ]
            [IS                                                  ]
File        [VIRTUAL                                             ]
Type ==>    [DLI (dbdname [literal-1])       [              ]  ]    +
            [VS ([ES] [PASSWORD 'literal-2'] [CREATE [RESET]]) ]
            [                                [UPDATE        ]  ]
            [                                [              ]  ]

            [       ]
            [CARD   ]
Device      [PUNCH  ]
Type ==>    [PRINTER]   +
            [DISK   ]
            [TAPE   ]
            [       ]
```

```
            [                                ]
            [F   literal-3                   ]
Record      [V   literal-3                   ]
Format ==>  [U   literal-4                   ]
            [                { literal-4 } ]   +
            [FB  (literal-3 { FULLTRK   })]
            [                { literal-4 } ]
            [VB  (literal-3 {           })]
            [                { FULLTRK   } ]
            [VBS (literal-3 { literal-4 })]
            [                { FULLTRK   })]
            [                                ]

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

File-name Parameter

FILE file-name

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

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

```
FILE PERSNL
```

SYSxxx Parameter (VSE Only)

FILE file-name [SYSxxx]

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

- SYSLST
- SYSPCH
- SYSIPT
- SYS000 through SYS240.

Check with your data processing department to learn if you must supply this parameter.

File-type Parameters

```
         [                                                    ]
         [IS                                                  ]
File-    [VIRTUAL                                             ]
type ==> [DLI (dbdname  [literal-1])       [              ] ]
         [                                 [CREATE [RESET]])]
         [VS ([ES]  [PASSWORD 'literal-2'] [              ] ]
         [                                 [UPDATE        ] ]
         [                                 [              ] ]
```

This parameter specifies your file-type. If you do not supply it, the assumption is that your file is sequentially ordered. If it is not, you must specify this parameter to identify your file-type. This subject is covered in more detail in the "File Processing" chapter. The file-types are:

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

Device-type Parameters

```
                  [       ]
                  [CARD   ]
Device-           [PUNCH  ]
 type   =====>    [PRINTER]
                  [DISK   ]
                  [TAPE   ]
                  [       ]
```

This optional parameter specifies where to look for your file. TAPE or DISK is for VSE only.

[CARD]

This option retrieves your file data from the system input stream (SYSIN for OS/390 and z/OS, SYSIPT for VSE). If your operating mode is the default (syntax check, compile, and execute), your file data must follow an END statement within your program, as illustrated below.

Only one file in your program can use the CARD option; this file must contain 80-character unblocked records.

```
FILE PERSUPD CARD
JOB INPUT PERSUPD
      ...(Program) ...
REPORT NEW-RPT ...
 ...
END
 ... (Data Records) ...
```

[PUNCH]

The PUNCH option indicates punched card output. Files created with this option are 80-character unblocked records.

[PRINTER]

The PRINTER option indicates print output files, referenced by the DISPLAY and REPORT statements.

DISK/TAPE (VSE Only)

This option (required only for VSE) indicates the device on which your file resides. Specify this option only if your file is on a device other than the default established at installation.

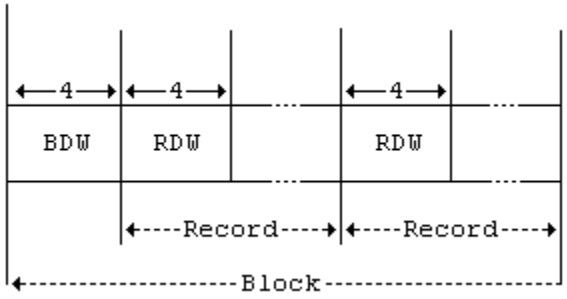
Record Format Parameters

```
                [                                ]
                [F   literal-3                   ]
                [V   literal-3                   ]
Record          [U   literal-4                   ]
Format ==>      [                {literal-4}    ]
                [FB  (literal-3  {          }  )]
                [                {FULLTRK   }    ]
                [                                ]
                [                {literal-4}    ]
                [VB  (literal-3  {          }  )]
                [                {FULLTRK   }    ]
                [                                ]
                [                {literal-4}    ]
                [VBS (literal-3  {          }  )]
                [                {FULLTRK   }    ]
                [                                ]
```

This parameter is required for VSE programs, but it is not necessarily required for OS/390 and z/OS programs. The record format is obtained from OS/390 or z/OS when the file is opened. It can be useful in OS/390 and z/OS, however, for output files. Record format codes are:

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

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

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



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

TABLE

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

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

INSTREAM Tables

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

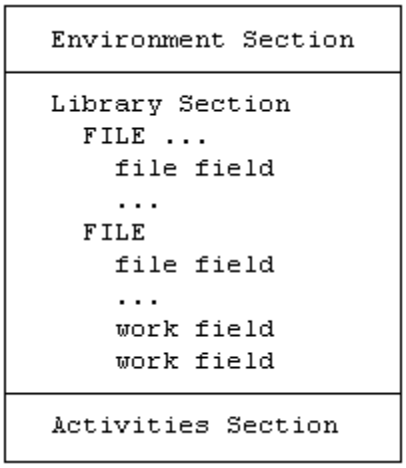
External Tables

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

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

DEFINE Statement

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



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

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

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

```
             [DEFINE]  field-name  +

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

                       {           {A}              }
                       {           {N}              }
Attributes ==>         {literal-2  {P}   [literal-3]}   +
                       {           {B}              }
                       {           {U}              }

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

                       [VALUE literal-5]
```

Field-name Parameter

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

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

The field names in the sample program are illustrated next.

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

Location Parameter

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

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

{literal-1}

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

{field-name-2}

Specifies the location of the leftmost byte of a file field as the relative displacement from the start of a previously defined field.

{W or S}

Establishes a working storage field. Fields coded as W are spooled to report (work) files; fields coded as S are not (see the "Report Processing" chapter).

In the field definitions sample program (shown earlier), the first designation to the right of the field-name is the location parameter. Four of the fields,

```
NAME, PAY-GROSS, DEPT, and DATE-OF-HIRE
```

are specified with a numeric value that indicates the starting position of each of these fields relative to the beginning of the record.

Four fields,

```
LAST-NAME, HIRE-MM, HIRE-DD, and HIRE-YY
```

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

Six fields,

```
VAC-HRS, SALARY, BONUS, RAISE, SERVICE, and CURR-DATE
```

are located in working storage. CURR-DATE also has three subfields:

```
CURR-MM, CURR-DD, and CURR-YY.
```

Attributes Parameter

```
                {           {A}             }
                {           {N}             }
Attributes ==>  {literal-2  {P}  [literal-3]}
                {           {B}             }
                {           {U}             }
```

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

Field Length (in bytes)

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

Data Format

Select one of the following codes:

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

N - zoned decimal. The field contains digits 0 through 9 in external decimal form (for example, 0 = X'FO').

P - packed decimal. The field contains numbers that meet IBM's definition of internal packed decimal. For example, a two-byte packed field containing the value 123 looks like X'123F'.

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

Length in Bytes	Digits
1	3
2	5
3	8
4	10

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

Number of Decimal Positions

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

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

Field Attribute Relationships

Data Format Code	Maximum Length (bytes)	Number of Decimal Positions
A	254	not valid
N	18	0 - 18
P	10	0 - 18
B	4	0 - 10
U	9	0 - 18

The field attribute specifications in the Sample Program Library Section, shown earlier, can be read as illustrated in the following table.

Field-name	Length	Format	Decimal Places
NAME	16 bytes	A	None
PAY-GROSS	4 bytes	P	2
DEPT	3 bytes	N	None
DATE-OF-HIRE	6 bytes	N	None
VAC-HRS	3 bytes	N	None
SALARY	4 bytes	P	2
BONUS	4 bytes	P	2
RAISE	4 bytes	P	2
SERVICE	2 bytes	N	None
CURR-DATE	6 bytes	N	None

Refer to the Sample Update Report in the "Overview" chapter, to see how the data fits into these field attribute specifications.

MASK Parameter

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

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

[letter]

Letter is an alphabetic identifier for a print mask that is:

- Specified in this DEFINE statement with literal-4
- Specified previously in the program
- Specified by your data center at installation. Check with them.

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

[BWZ]

BWZ (blank when zero) suppresses printing a numeric field when it contains all zeros.

[literal-4]

Literal-4 is the print edit mask to use. It is an alphabetic literal created with a combination of the following characters:

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

character

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

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

```
Number of
Decimals     ____________________ Mask

  none             ZZZZZZZZZZZZZZZZZZ        *
    0          ZZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ-
    1           ZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ.9-
    2            Z,ZZZ,ZZZ,ZZZ,ZZZ,ZZZ.99-
    3              ZZZ,ZZZ,ZZZ,ZZZ,ZZZ.999-
    4               ZZ,ZZZ,ZZZ,ZZZ,ZZZ.9999-
    5                Z,ZZZ,ZZZ,ZZZ,ZZZ.99999-
    6                  ZZZ,ZZZ,ZZZ,ZZZ.999999-
    7                   ZZ,ZZZ,ZZZ,ZZZ.9999999-
    8                    Z,ZZZ,ZZZ,ZZZ.99999999-
    9                      ZZZ,ZZZ,ZZZ.999999999-
   10                       ZZ,ZZZ,ZZZ.9999999999-
   11                        Z,ZZZ,ZZZ.99999999999-
   12                          ZZZ,ZZZ.999999999999-
   13                           ZZ,ZZZ.9999999999999-
   14                            Z,ZZZ.99999999999999-
   15                              ZZZ.999999999999999-
   16                               ZZ.9999999999999999-
   17                                Z.99999999999999999-
   18                                  .999999999999999999-
* For zoned decimal fields with no decimals, the default
  mask is '999999999999999999'.
```

The next exhibit illustrates some print masks and their purposes.

```
Mask             Use

'(999) 999-9999'          Telephone Number
'999-99-9999'             Social Security Number
'99/99/99'                Date
'$$$$$$$9.99-'            Money (with floating $)
'*******999.99-'          Protected check amount
```

VALUE Parameter

```
[VALUE literal-5]
```

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

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

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

where:

DEFINE—is the keyword that identifies your statement

MONTH—is the name of the field being defined

W—is the location parameter = working storage

10—is the field length parameter = 10 bytes

A—is the data format parameter = alphabetic

VALUE—specifies the initial contents of MONTH = JANUARY.

Initialization of a numeric field might read:

```
DEFINE YEAR W 4 N VALUE 1999
```

where:

DEFINE—is the keyword

YEAR—is your field-name

W—locates your field in working storage

4—indicates a field length of 4 bytes

N—specifies a zoned decimal data format

VALUE—initializes field YEAR to 1999.

Chapter

3

Activity Definition

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

JOB Activities

JOB activities, identified by the JOB statement, read data from input files described in the library section of your program. See the "Library" chapter. They examine and manipulate this data, and write data to output files and the appropriate report declaratives.

SORT Activities

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

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

This chapter discusses the JOB and SORT statements and their associated parameters. These statements provide the information required for automatic input and output of data. Data under your control is input with the GET and READ statements, and output with the PUT and WRITE statements.

The next exhibit illustrates the activity portion of the Sample Program including the REPORT declaratives. The Sample Program is depicted in the "Overview" chapter under the topic Application.

Sample Program Activity Section

```
 1 PARM  DEBUG(FLOW FLDCHK)
 2 *
 3 FILE PERSNL FB(150 1800)
 4   NAME                      17  16   A
 5     LAST-NAME             NAME   8   A
 6   PAY-GROSS                 94   4   P  2
 7   DEPT                      98   3   N
 8   DATE-OF-HIRE             136   6   N
 9     HIRE-MM   DATE-OF-HIRE       2   N
10     HIRE-DD   DATE-OF-HIRE +2    2   N
11     HIRE-YY   DATE-OF-HIRE +4    2   N
12   SALARY                     W   4   P  2
13   BONUS                      W   4   P  2
14   RAISE                      W   4   P  2
15   SERVICE                    W   2   N
16   CURR-DATE                  S   6   N
17     CURR-MM      CURR-DATE       2   N
18     CURR-DD      CURR-DATE +2    2   N
19     CURR-YY      CURR-DATE +4    2   N
20 *
21 FILE ERRPRINT PRINTER
22 *
23 JOB INPUT PERSNL
24   %GETDATE CURR-DATE
42   SALARY = PAY-GROSS * 52
43   PERFORM SERVICE-CALC
44   IF SERVICE LT 1
45     GO TO JOB
46   END-IF
47   PERFORM RAISE-CALC
48   BONUS = 0
49   IF SERVICE GT 14
50      PERFORM BONUS-CALC
51   END-IF
52   SALARY = SALARY + RAISE + BONUS
53   PRINT UPD-RPT
54 *
55   SERVICE-CALC. PROC
57     SERVICE = CURR-YY - HIRE-YY
58     IF CURR-MM < HIRE-MM
59       SERVICE = SERVICE - 1
60     END-IF
61     IF CURR-MM NE HIRE-MM
62       GOTO QUIT-SERV-CALC
63     END-IF
64     IF CURR-DD < HIRE-DD
65       SERVICE = SERVICE - 1
66     END-IF
67     QUIT-SERV-CALC
68   END-PROC
69 *
70   RAISE-CALC. PROC
72     IF DEPT LT 940
73       RAISE = SALARY * 0.1
74     ELSE
75       RAISE = SALARY * 0.15
76     END-IF
77   END-PROC
78 *
```

```
 79   BONUS-CALC. PROC
 81     IF SALARY GT 29999
 82       DISPLAY ERRPRINT, LAST-NAME, +5, +
                  'INELIGIBLE FOR BONUS'
 83       GOTO QUIT-BONUS
 84     END-IF
 85     IF SERVICE GT 19
 86        BONUS = 2000
 87     ELSE
 88        BONUS = 1000
 89     END-IF
 90     PRINT BONUSRPT
 91     QUIT-BONUS
 92   END-PROC
 93 *
 94 REPORT UPD-RPT PAGESIZE 51  LINESIZE 63 NODATE NOPAGE
 95   SEQUENCE DEPT LAST-NAME
 96   CONTROL DEPT
 97   TITLE 1  'ANNUAL UPDATE REPORT - SALARIED EMPLOYEES'
 98   HEADING  LAST-NAME   'NAME'
 99   HEADING  SERVICE     'SERV'
100   LINE DEPT LAST-NAME SERVICE RAISE SALARY
101 *
102 REPORT BONUSRPT LINESIZE 60 NODATE NOPAGE
103   SEQUENCE DEPT LAST-NAME
104   TITLE 1  'ANNUAL BONUS REPORT - SENIOR EMPLOYEES'
105   LINE DEPT LAST-NAME SERVICE BONUS
106 *
```

JOB Statement

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

```
    [INPUT  (file-name [KEY(field-name...)] ...) ]
JOB [                                            ][NAME job-name]
    [NULL                                        ]
```

[INPUT]

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

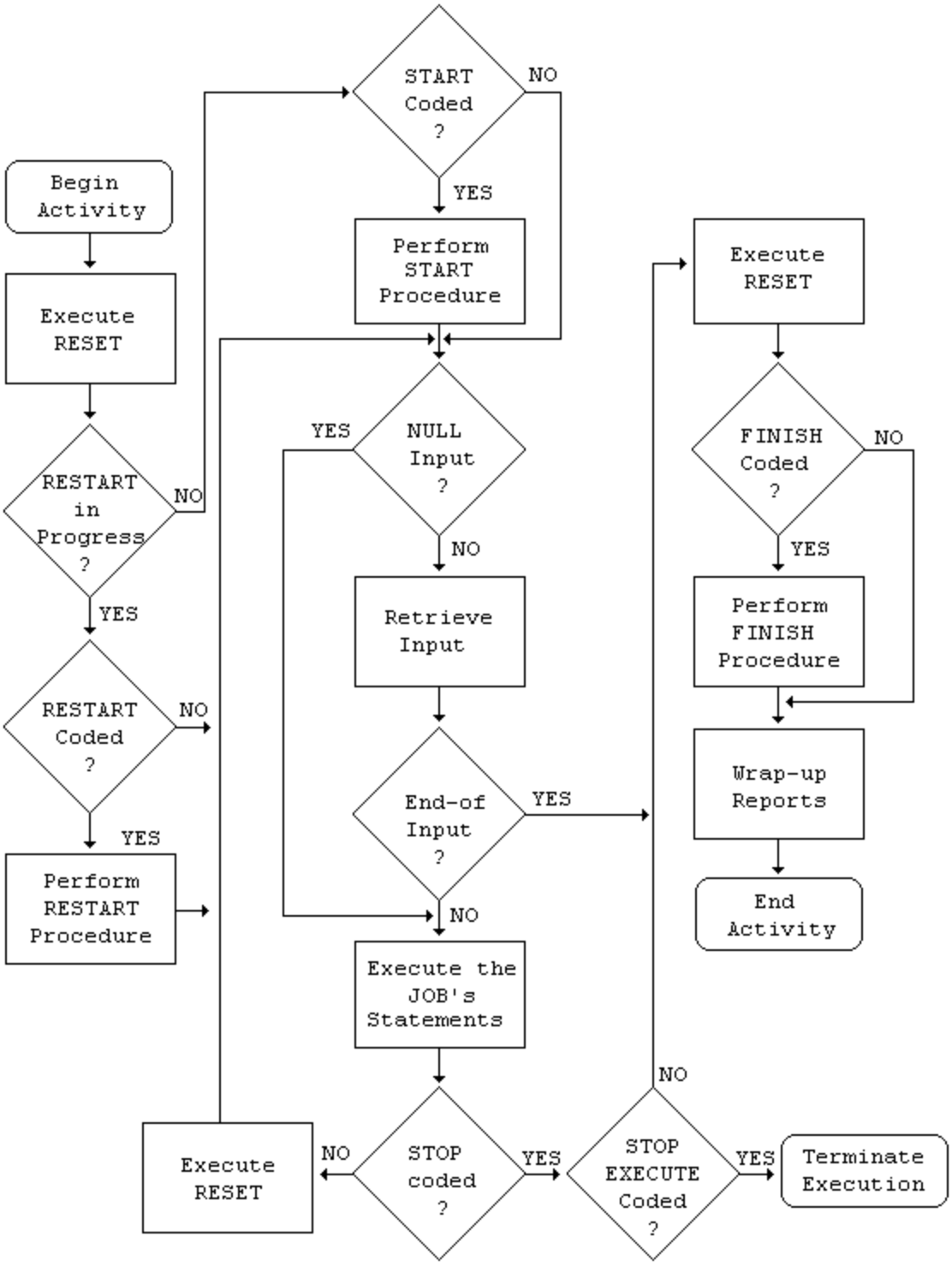
Parameter	Description
file-name	Provides the name of the file you want to have controlled automatically. This can be any name previously coded on a FILE statement.
KEY field-name	Use this subparameter to identify fields within the above-named file when it is used in synchronized file processing (see the "File Processing" chapter). The files are processed in the order in which these keys appear in the JOB statement. There is no limit on the number of fields that you can use as keys.

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

If you do not specify the INPUT parameter, an automatic input file is provided. The default input file is chosen as follows:

1. First choice is the file created by a SORT operation that immediately preceded this JOB activity. If there is no such SORT file, see number 2 below.
2. Second choice is the first file that you specified in the library section of this program.

The next exhibit illustrates the processing flow of a JOB activity:



SORT Statement

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

The next exhibit diagrams the SORT statement and associated parameters.

```
SORT  file-name-1  +

      TO file-name-2  +

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

      [BEFORE proc-name]      +

      [NAME sort-name]
```

file-name-1

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

TO file-name-2

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

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

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

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

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

[BEFORE proc-name]

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

SELECT Statement

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

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

[NAME sort-name]

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

Chapter

4

Data Manipulation

This chapter describes several ways you can manipulate data within your program. The primary vehicle is the Assignment statement.

Assignment Statement

The Assignment statement establishes the value of a field by one of two means:

- Equivalence by copying the data from another (named) field, or from a specified literal
- As the result of an arithmetic expression.

Equivalence

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

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

The value of field-name-1 is set equal to the value of field-name-2 or literal, whichever is specified. There are certain restrictions on this function, as follows:

- You can specify only one equivalent; that is, either field-name-2 or literal.
- If field-name-1 has been DEFINEd as alphabetic, literal must also be alphabetic. If literal is shorter than field-name-1, padding is on the right.
- If both field-name-1 and field-name-2 are alphabetic, but not the same size, padding or truncation, as appropriate, occurs on the right.
- If field-name-1 is alphabetic and field-name-2 is numeric, the resulting value in field-name-1 is zoned decimal, with padding or truncation on the left, as necessary.
- If field-name-1 is numeric, field-name-2 or literal must be numeric.

Arithmetic Expression

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

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

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

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

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

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

Another Assignment statement in the above exhibit is:

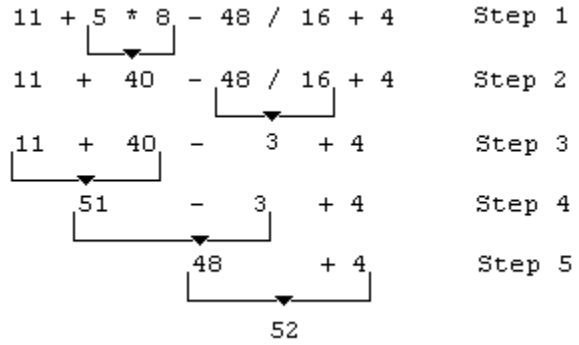
```
SALARY = SALARY + RAISE + BONUS
```

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

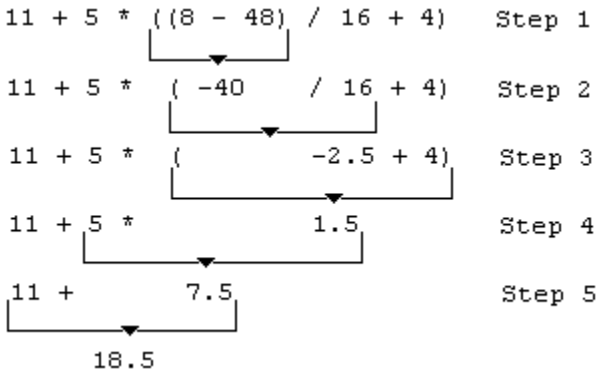
Arithmetic operations are normally performed in the following order:

```
* multiplication  or  /  division
+ addition        or  -  subtraction
```

This customary evaluation order is illustrated in the next exhibit.



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



Chapter

5

Decision and Branching Logic

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

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

IF and DO contain the conditional expressions on which the decisions are based.

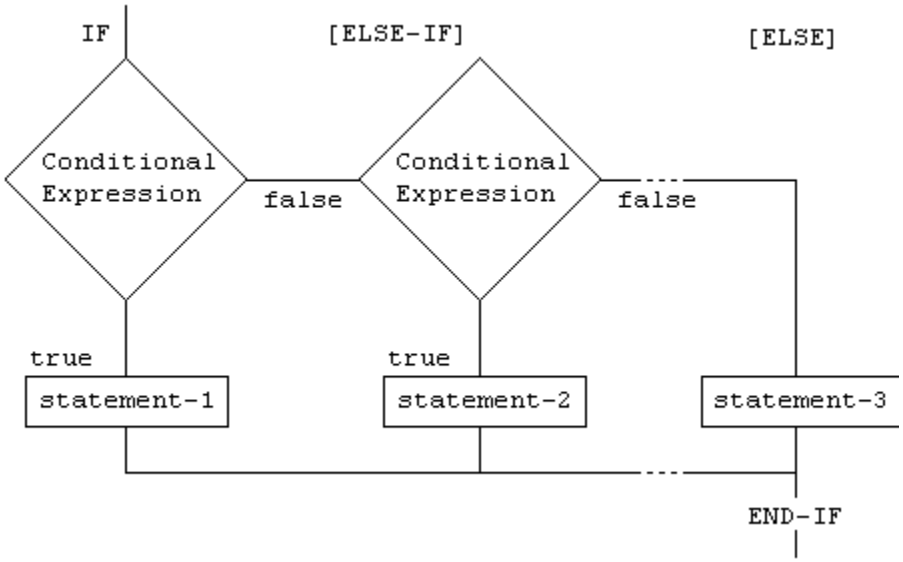
IF Statement Construction

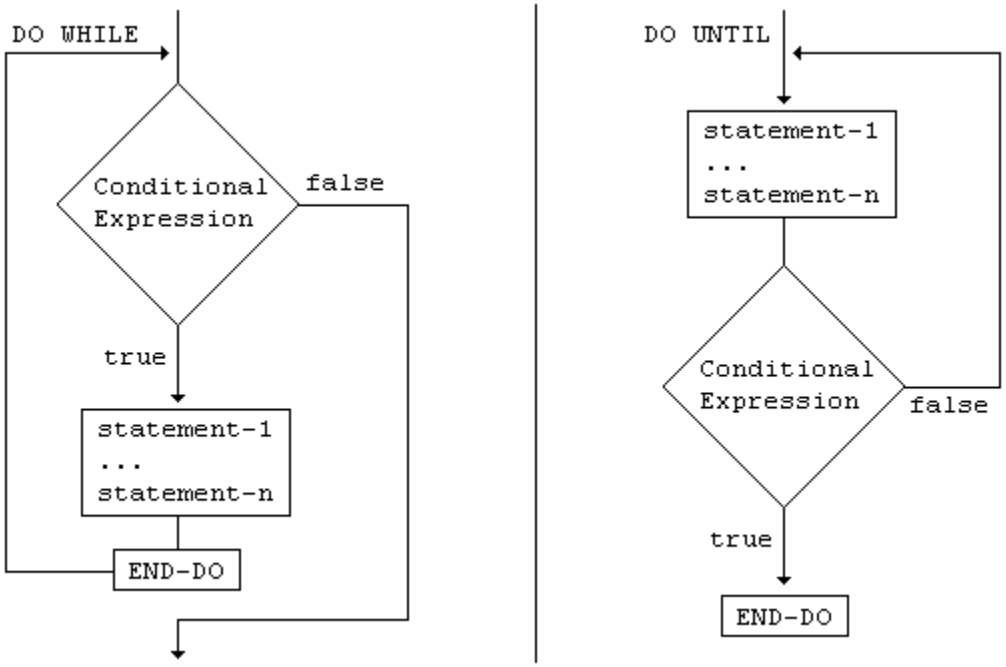
```
 IF  condition
   *Statements executed if condition is true*

[ELSE                                      ]   Optional
[ *Statements executed if condition is false*]

 END-IF
```

DO Statement Construction

```
DO WHILE  condition
 *Statements executed repetitively if condition is true*
END-DO
```

The next two exhibits illustrate the processing that takes place when an IF or DO statement is executed.





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

Conditional Expressions

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

Single:

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

Combined:

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

Combinations can be between like conditions, such as two field relational conditions:

```
IF NAME EQ 'ANDERSON', AND EMPL# EQ 41552
```

Or unlike conditions, such as one field class condition and one field relational condition:

```
IF EMPL# NUMERIC, AND EMPL# GT 15555
```

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

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

The statement that reads:

```
IF SERVICE GT 19
```

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

The statement that reads:

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

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

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

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

File presence and record relational are useful only with synchronized file processing and are discussed in detail in the "File Processing" chapter. The extended conditions are:

- Field Series
- File Presence Series.

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

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

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

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

In the above exhibit, the first IF statement combines two field relational conditions to test for a specific name and a specific employee number. The second IF statement combines two field series conditions to test for a numeric and an alphabetic range. The third IF combines a field class and a field relational condition. The field class condition tests to see if field EMPL# is numeric. The field relational condition tests to see if the value of this field is greater than 15555.

The last IF statement also combines the field relational and field class conditions. The field class condition tests to see if field NET is zeros; the two field relational conditions test to see if the value of field NET is greater than the value of field GROSS, or if the value of field DEDUCTIONS does not equal the result of the arithmetic expression (GROSS - NET).

The most commonly used condition formats are:

- Field Relational
- Field Class
- Field Series.

These formats are described in detail next.

Field Relational Condition

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

```
                { EQ  = }
                { NE ¬= }
                { LT  < } { field-name-2          }
field-name-1    {       } { literal               }
                { LE <= } { arithmetic expression }
                { GT  > }
                { GE >= }
```

Valid operators for the field relational condition are:

```
EQ    =    - Equal
NE    ¬=   - Not equal
LT    <    - Less than
LE    <=   - Less than or equal to
GT    >    - Greater than
GE    >=   - Greater than or equal to
```

The following rules apply to the use of this condition:

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

The next exhibit presents some examples of field relational conditions:

```
*
FILE PAYFILE
  EMPL#         9   5  N
  NAME         17  20  A
  NET          90   4  P  2
  GROSS        94   4  P  2
  DEDUCTIONS    W   4  P  2
*
JOB INPUT PAYFILE
  IF  NAME  NE  'ANDERSON'
  IF  NET  LT  GROSS
  IF  EMPL#  GT  10555
  IF  DEDUCTIONS  EQ  (GROSS - NET)
*
```

Field Class Condition

This condition determines whether a named field does or does not contain a certain class of data, specifically, alphabetic, numeric, zero, or space characters, X'FFs, or X'00s. The format is diagrammed in the next exhibit.

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

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

Term	Description
ALPHABETIC	Each byte of the field is tested for either letters A through Z or a space character.
NUMERIC	The field is tested for digits 0 through 9 in the correct format for the field's defined data type. In the case of data types N and P, the low-order position of the field is tested for a valid sign.
SPACE, SPACES	Each byte of the field is tested for the space character.
ZERO, ZEROS, ZEROES	The field is tested for a zero value in the correct format for the field's defined data type.
HIGH-VALUES	Each byte of the field is tested for the X'FF' character.
LOW-VALUES	Each byte of the field is tested for the X'00' character.

The next exhibit illustrates the use of the field class condition.

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

Field Series Condition

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

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

Valid operators for the field series condition are:

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

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

- If field-name-1 is alphabetic, it can be compared to alphabetic or numeric fields, and/or alphabetic literals. Numeric fields are converted to zoned decimal before the comparison is made.
- If field-name-1 is numeric, it can be compared to numeric fields and/or literals. It cannot be compared to alphabetic fields or literals.
- Each value in the series of values to the right of the operator represents either a single value (for example, 10555) or a range of values (for example, 10555 through 15555).
- A field series conditional expression using the equal operator is tested by comparing field-name-1 to each value in the series. If the value is a single value, the test is for equality between field-name-1 and field-name-2 (or literal-1). If the value is a range of values, the test is for field-name-1 within the range defined by field-name-2 (or literal-1) and field-name-3 (or literal-2). The field series conditional expression is true if at least one test is true.
- A field series conditional expression using the not-equal operator is tested by comparing field-name-1 to each value in the series. If the value is a single value, the test is for inequality between field-name-1 and field-name-2 (or literal-1). If the value is a range of values, the test is for field-name-1 outside the range defined by field-name-2 (or literal-1) and field-name-3 (or literal-2). The field series conditional expression is true only if all tests are true.

The next exhibit presents some examples of field series conditions.

```
*
FILE PAYFILE
  EMPL#         9   5  N
  NAME         17  20  A
  NET          90   4  P  2
  GROSS        94   4  P  2
  DEPT#        98   3  N
  DEDUCTIONS    W   4  P  2
*
JOB INPUT PAYFILE
  IF NAME       EQ  'ANDERSON', 'BAKER', 'CARROLL'
  IF NET        NE  GROSS, EMPL#, DEDUCTIONS, 9999.9
  IF EMPL#      EQ  10555, 11555, 12550, 15550, 15555
  IF DEDUCTIONS NE  GROSS, NET, 999, 1111.34
*
JOB INPUT   PAYFILE
  IF NAME    EQ  'ANDERSON' THRU 'CARROLL'
  IF DEPT#   EQ   911 THRU 921
  IF EMPL#   NE  10555 THRU  15555
  IF NET     NE   DEDUCTIONS THRU GROSS
*
```

File Presence Condition

This condition determines if a record of the named input file is available for processing. It is discussed in detail in the "File Processing" chapter.

File Presence Series Condition

This condition is used in a JOB with synchronized file processing to determine whether or not the records from more than one file have the same key. This condition is discussed in detail in the "File Processing" chapter.

Record Relational Condition

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

IF, ELSE, and END-IF Statements

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

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

Note: We recommend that you code your CA-Easytrieve Plus source programs in uppercase only. Lowercase keywords are not recognized by the compiler.

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

```
 2 *
 3 FILE PERSNL FB(150 1800)
 4   NAME                    17  16   A
 5     LAST-NAME           NAME   8   A
 6   PAY-GROSS               94   4   P  2
 7   DEPT                    98   3   N
      ...
15   SERVICE                  W   2   N
      ...
22 *
23 JOB INPUT PERSNL
     ...
43   PERFORM SERVICE-CALC
44   IF SERVICE LT 1
45     GO TO JOB
46   END-IF
47   PERFORM RAISE-CALC
     ...
69 *
70   RAISE-CALC. PROC
72     IF DEPT LT 940
73       RAISE = SALARY * 0.1
74     ELSE
75       RAISE = SALARY * 0.15
76     END-IF
77   END-PROC
78 *
```

IF Statement

The IF statement controls the execution of subsequent statements that are associated with it. As a general rule, these associated statements should be indented below the IF statement so their relationship is immediately noticeable.

In the above exhibit, the first IF statement contains a field relational conditional expression that tests to see if the value of field SERVICE is less than 1.

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

The GO TO statement is discussed later in this chapter.

ELSE Statement

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

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

END-IF Statement

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

Nesting IF Statements

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

DO and END-DO Statements

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

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

DO Statement

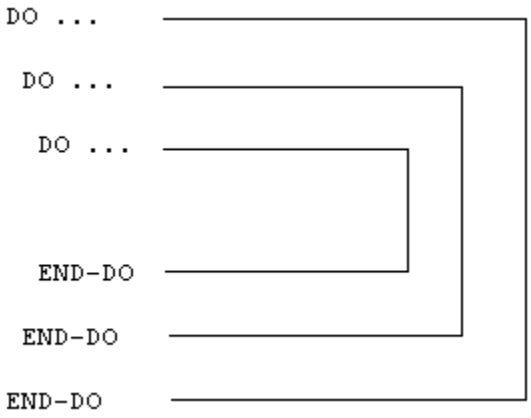
This statement identifies one or more statements that are to be executed WHILE the conditional expression tests true. When the condition tests false, the statements are bypassed. The conditional expression must have the possibility to be false eventually or DO loops forever. In the above exhibit, each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

END-DO Statement

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

Nesting DO Loops

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



GOTO (or GO TO) Statement

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

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

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

Note: GOTO and GO TO work the same way and may be used interchangeably.

Statement Labels

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

Note: We recommend that you code your CA-Easytrieve Plus source programs in uppercase only. Lowercase keywords are not recognized by the compiler.

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

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

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

```
22 *
23 JOB INPUT PERSNL
24   %GETDATE CURR-DATE
42   SALARY = PAY-GROSS * 52
43   PERFORM SERVICE-CALC
44   IF SERVICE LT 1
45     GO TO JOB
46   END-IF
     ...
54 *
55   SERVICE-CALC. PROC
57     SERVICE = CURR-YY - HIRE-YY
58     IF CURR-MM < HIRE-MM
59       SERVICE = SERVICE - 1
60     END-IF
61     IF CURR-MM NE HIRE-MM
62       GOTO QUIT-SERV-CALC
63     END-IF
64     IF CURR-DD < HIRE-DD
65       SERVICE = SERVICE - 1
66     END-IF
67     QUIT-SERV-CALC
68   END-PROC
69 *
```

Procedure Processing

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

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

PROC and END-PROC Statements

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

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

The procedure is constructed as follows:

Proc-name

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

Statement 1 through Statement n

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

END-PROC

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

Code any procedures immediately after their associated activity (JOB or SORT) or subactivity (REPORT). Procedures that you define are invoked by PERFORM statements. In addition, there are special-name report procedures that are used in report processing (see the "Report Processing" chapter).

PERFORM Statement

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

```
PERFORM  proc-name
```

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

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

STOP Statement

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

```
STOP  [EXECUTE]
```

EXECUTE

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

```
*
  BONUS-CALC. PROC
    IF SALARY GT 29999
      DISPLAY ERRPRINT, LAST-NAME, +5, +
              'INELIGIBLE FOR BONUS'
      STOP
    END-IF
    IF SERVICE GT 19
       BONUS = 2000
    ELSE
       BONUS = 1000
    END-IF
    PRINT BONUSRPT
  END-PROC
*
```

If you want to stop the current activity and bypass all subsequent activity, use the EXECUTE parameter. The next exhibit provides an example of complete termination.

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

Chapter

6

Input/Output Specification

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

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

- Automatic
- Controlled
- Database.

Automatic I/O

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

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

Controlled I/O

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

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

Database I/O

The most complex level of input/output involves the use of databases. Refer to the "IMS/DLI Processing" chapter.

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

DISPLAY Statement

Use the DISPLAY statement to output data to the system printer or a named file. This data is spaced according to the specified parameters.

Syntax

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

Parameters

[file-name]

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

```
DISPLAY ERRPRINT
```

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

```
[ SKIP literal-1    ]
[ CONTROL literal-2 ]
[ NEWPAGE           ]
```

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

```
DISPLAY ERRPRINT SKIP 10
```

Content and Spacing Parameters

```
[ literal-3  ]
[ field-name ]
```

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

```
DISPLAY ERRPRINT, DEDUCTIONS, GROSS
```

```
[ +literal-4 ]
[ -literal-4 ]
```

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

```
DISPLAY ERRPRINT, DEDUCTIONS, +5, GROSS
```

[COL literal-5]

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

```
DISPLAY ERRPRINT, DEDUCTIONS, COL 40, GROSS
```

Rules for Use

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

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

Some typical uses of the DISPLAY statement include:

- Printing specially formatted lines in a report, which is outside the capabilities of the REPORT declaratives. This should be done in a procedure coded at the end of the REPORT declaratives. The procedure is executed at the time the report data is formatted. The DISPLAYed data appears in the place you want.
- Printing error messages when abnormal conditions are encountered. You can avoid interspersing displayed data with your report data by specifying the file-name option on the DISPLAY statement, as illustrated within the BONUS-CALC procedure in the Sample Program under the Application topic in the "Overview" chapter.

Debugging

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

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

Refer to the *Reference Guide* for a detailed discussion of the ways to use this debugging aid.

PRINT Statement

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

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

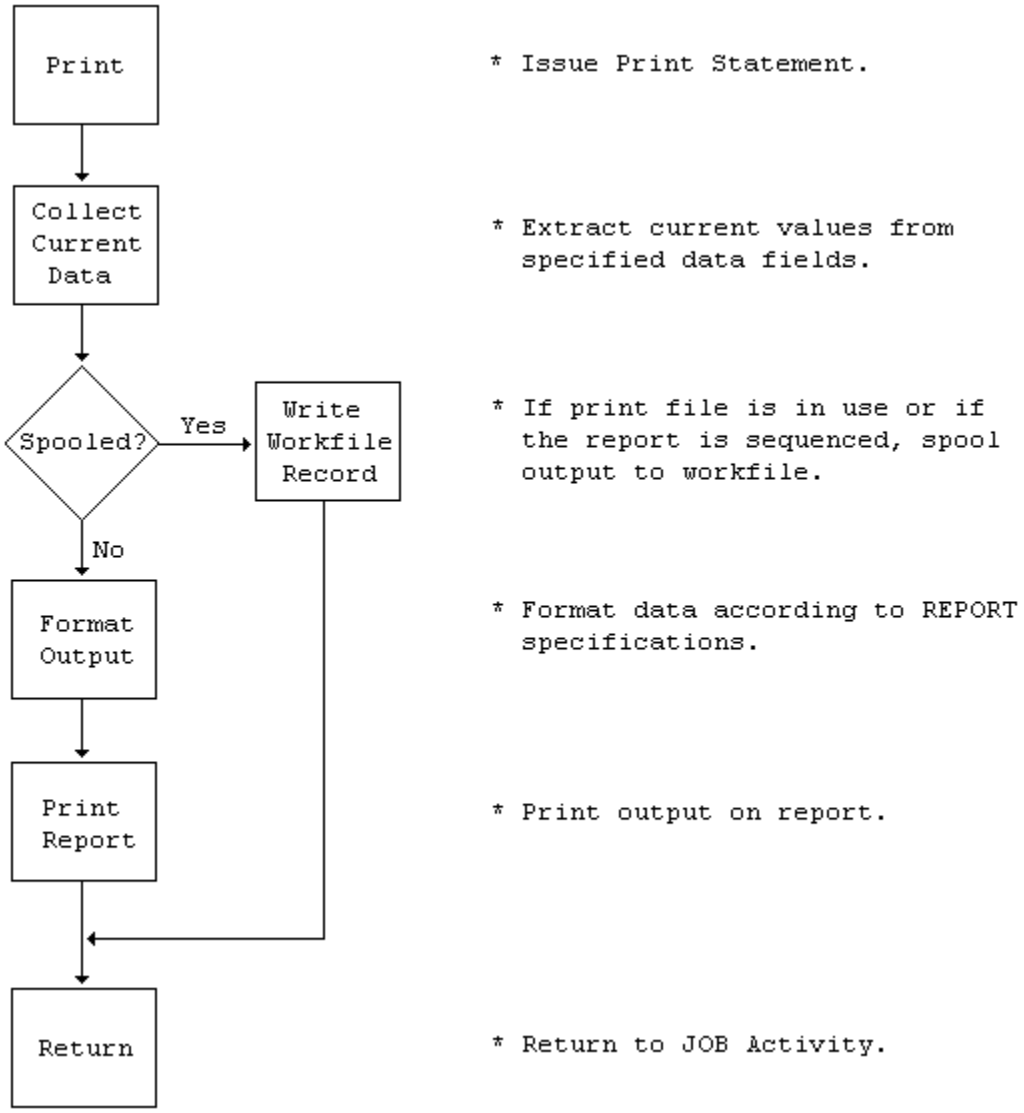
The next exhibit diagrams the format of the PRINT statement.

```
PRINT  report-name
```

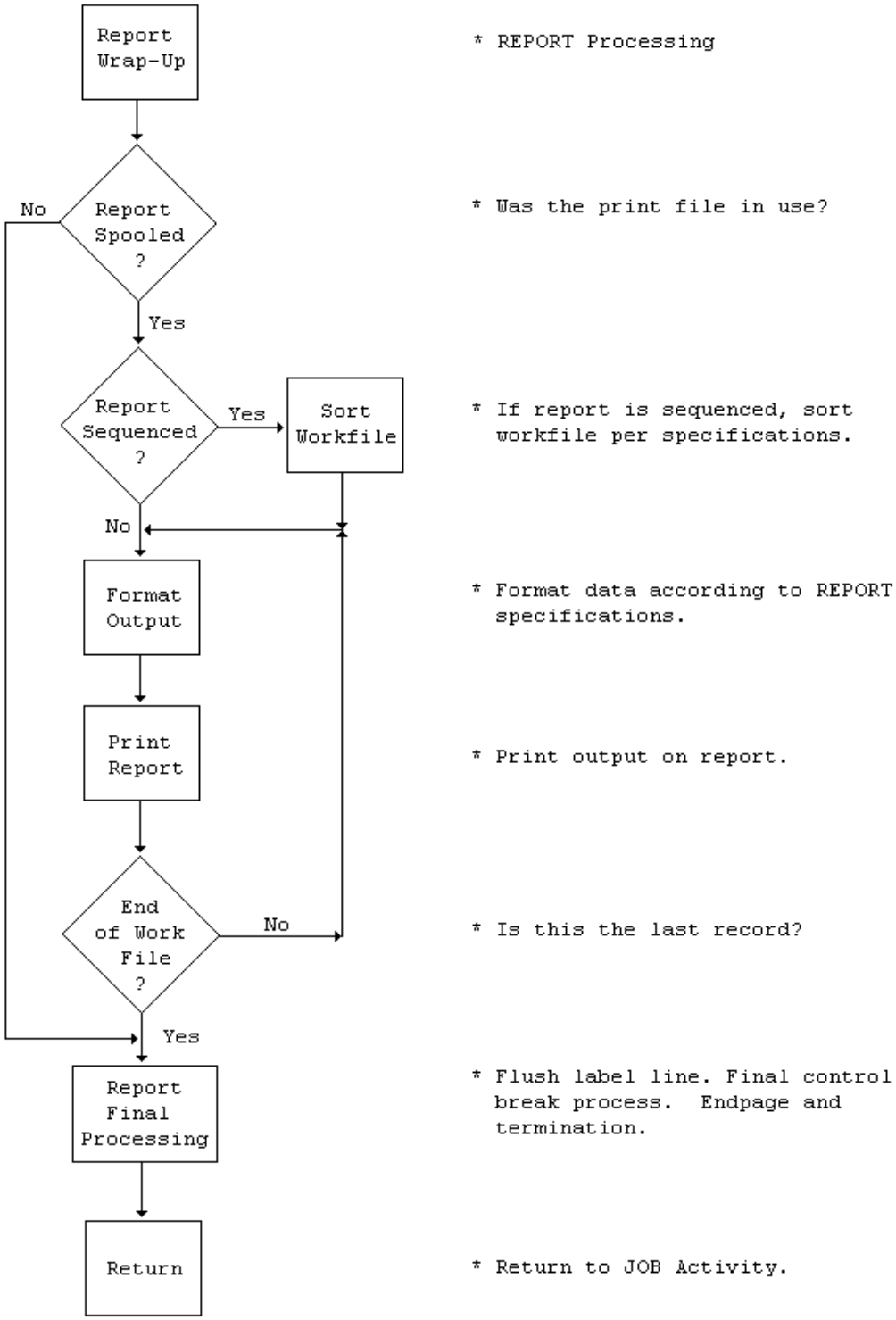
The report-name parameter is the name of the report that contains the data being output with the PRINT statement.

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

The next exhibit illustrates this process.



If the report is sequenced, the data is output to a work file that is sorted before the report is printed. The next exhibit illustrates this process.



GET Statement

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

```
GET  file-name
```

file-name

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

PUT Statement

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

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

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

file-name-1

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

[FROM file-name-2]

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

PUT Example

The next exhibit illustrates the use of the PUT statement.

```
JOB START POINTER INPUT PAYFILE
  IF REC-KEY GE 600
     STOP
  END-IF
  SALARY = SALARY * 1.1
  PUT SALUPD FROM PAYFILE
  PRINT UPD-RPT
*
  POINTER.  PROC
    POINT PAYFILE GE 500
  END-PROC
```

*

In the above exhibit, the statements retrieve those records with keys between 500 and 599 inclusive from file PAYFILE, increase the value in the SALARY field of each record by 10 percent, and output each updated record to file SALUPD and report UPD-RPT.

POINT Statement

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

Syntax

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

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

Parameters

file-name

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

Relational Operator

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

Search Value Parameters

{literal }
{ }
{field-name}

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

The next exhibit illustrates the use of the POINT statement.

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

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

READ Statement

The READ statement provides random access to keyed and relative-record VSAM and ISAM files.

Syntax

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

Parameters

file-name

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

KEY field-name

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

[STATUS]

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

The next exhibit illustrates the use of the READ statement.

```
FILE PAYFILE VS UPDATE
     EMPL#  W   5  N
     NAME   6  20  A
*
JOB INPUT NULL
  EMPL# = 44152
  READ PAYFILE, KEY EMPL#, STATUS
  IF FILE-STATUS NOT ZERO
    GOTO ERRTASK
  END-IF
  IF NAME EQ 'OLDNAME,M.'
    NAME EQ 'NEWNAME,M.'
    WRITE PAYFILE UPDATE
  ELSE
    GOTO ERRTASK
  END-IF
  STOP
*
```

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

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

WRITE Statement

Use the WRITE statement to maintain keyed and relative-record VSAM files (ISAM files are read/only). WRITE updates or deletes the current record of the named file, or adds new records.

Syntax

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

Parameters

file-name-1

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

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

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

[FROM file-name-2]

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

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

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

Chapter

7

Report Processing

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

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

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

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

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

Both of these statements are described in detail in the "Input/Output Specification" chapter. The discussion of automatic report processing in this chapter uses the PRINT statement exclusively.

The desired reports are defined by a set of statements at the end of the JOB activity. These statements specify the report type, format, sequence, and content, as follows:

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

You must code these statements in the order listed above.

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

Sample Program Report Declaratives

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

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

- The report name is UPD-RPT. Each page of the printed output is 51 lines long and 63 columns wide. Neither the date nor the page number is printed on the first title line of each page.
- The report is ordered (sequenced) by two levels: first, by department number and, within each department, in order by last name.
- The dollar values are subtotaled for each department, and the report is segmented by department.
- The title ANNUAL UPDATE REPORT - SALARIED EMPLOYEES is centered across the top of the report page.
- The column heading for field LAST-NAME reads NAME, and for field SERVICE reads SERV.
- There are five columns spaced three characters apart across the 63-character-wide report. The columns contain the data in the fields: DEPT, LAST-NAME, SERVICE, RAISE, and SALARY in that order from left to right.

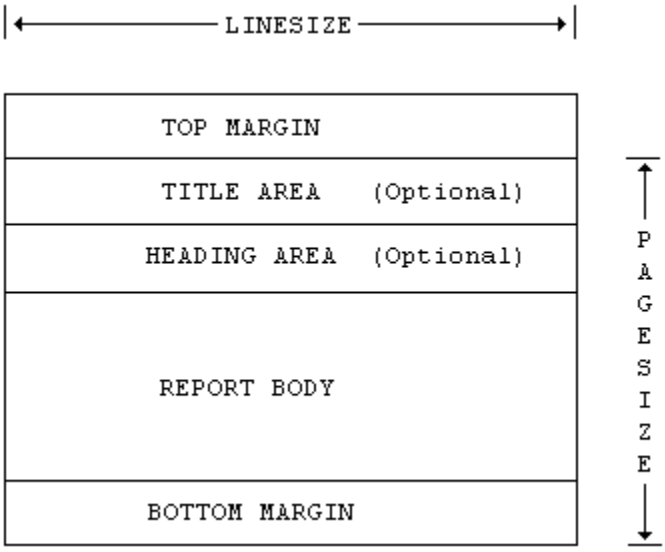
The result of this specification is the Sample Update Report that is illustrated in the "Overview" chapter.

Report Types

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

Standard Reports

The default is the standard format illustrated below.



TOP MARGIN

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

TITLE AREA

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

HEADING AREA

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

REPORT BODY

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

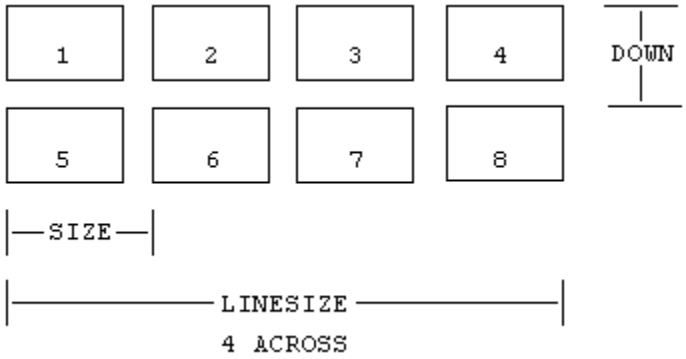
BOTTOM MARGIN

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

The default values for report spacing are sufficient for most applications.

Label Reports

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



Each individual label is one line group. Each PRINT statement in the JOB activity of your program produces one label, formatted on the lines of that label according to the report declaratives. The DOWN and SIZE parameters specify the label dimensions.

REPORT Statement

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

REPORT statement parameters fall into four categories:

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

The syntax of the REPORT statement is diagrammed below.

```
REPORT  report-name +

        [SUMMARY]   +

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

        [PRINTER file-name]      +         File Directing

        [PAGESIZE ({literal-6a} [literal-6b])] +
        [LINESIZE literal-5]                   +
        [SPREAD]                               +         Spacing
        [NOSPREAD]                             +         Control
        [NOADJUST]                             +
        [NODATE]                               +
        [NOPAGE]                               +

        [LIMIT literal-6]        +         Testing
        [EVERY literal-7]                  Aids
```

report-name

```
REPORT  report-name
```

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

[SUMMARY]

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

LABELS

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

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

ACROSS literal-1

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

DOWN literal-2

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

SIZE literal-3

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

NEWPAGE

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

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

```
LINESIZE >= (ACROSS - 1) * SIZE + (number of print positions on an
                                   individual label)
```

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

Mailing Label Program

```
 1 PARM  DEBUG(FLOW FLDCHK)
 2 *
 3 FILE PERSNL FB(150 1800)
 4   NAME                    17  16   A
 5     LAST-NAME             17   8   A
 6     FIRST-NAME            25   8   A
 7   ADDRESS                 37  39   A
 8     STREET                37  20   A
 9     CITY                  57  12   A
10     STATE                 69   2   A
11     ZIP                   71   5   N
12   DATE-OF-HIRE           136   6   N
13     HIRE-MM              136   2   N
14     HIRE-DD              138   2   N
15     HIRE-YY              140   2   N
16   SERVICE                  W   2   N
17   CURR-DATE                S   6   N
18     CURR-MM      CURR-DATE      2   N
19     CURR-DD      CURR-DATE +2   2   N
20     CURR-YY      CURR-DATE +4   2   N
21 *
22 JOB INPUT PERSNL
23   %GETDATE CURR-DATE
41   PERFORM SERVICE-CALC
42   IF SERVICE GT 19
43     PRINT MAILOUT
44   END-IF
45 *
46   SERVICE-CALC. PROC
48     SERVICE = CURR-YY - HIRE-YY
49     IF CURR-MM < HIRE-MM
50       SERVICE = SERVICE - 1
51     END-IF
52     IF CURR-MM NE HIRE-MM
53       GOTO QUIT-SERV-CALC
54     END-IF
55     IF CURR-DD < HIRE-DD
56       SERVICE = SERVICE - 1
57     END-IF
58     QUIT-SERV-CALC
59   END-PROC
60 *
61 REPORT MAILOUT LABELS (ACROSS 2  DOWN 4  SIZE 30)
62   SEQUENCE LAST-NAME
63   LINE 1 FIRST-NAME -3 LAST-NAME
64   LINE 2 STREET
65   LINE 3 CITY  -3 STATE  ZIP
66 *
```

Labels Produced by Mailing Label Programs

```
NANCY   BERG                PATTI   HUSS
3710 JENIFER ST N W         1355 TEWKESBURY PLAC
BALTIMORE   MD   21055      CLEARWATER  FL   33512

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

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

SAMUEL  OSMON               KATHY   PETRIK
4201 CATHEDRAL AVE N        5005 BENTON AVE
CHICAGO     IL   60618      WASHINGTON  DC   20032

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

PAT     ROGERS              CINDY   SMOTH
1625 FRANKLIN ST N E        4120 18TH STREET NE
CHICAGO     IL   60691      DALLAS      TX   75219

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

[PRINTER file-name]

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

[PAGESIZE]

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.

[LINESIZE literal-5]

This option specifies a value for the left-to-right width of each line of your report. The value of literal-5 is the number of print columns on each report line. The maximum you can specify is one character less than the physical length (record size) of the printer file receiving the report. The default is commonly 132 characters, which is one less than the actual size of the typical printer file record (133 characters). The first character is used for vertical form control (carriage control). Check with your data center to determine the default value for your installation.

LINESIZE must be able to accommodate the maximum size of all the fields listed across your report, including extra characters for totals when requested. A LINESIZE of 60 or 63 characters is specified for the reports in the sample program to enable them to fit on the pages of this guide.

[SPREAD]
[NOSPREAD]

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

[NOADJUST]

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

[NODATE]

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

[NOPAGE]

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

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

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

produces the SUMMARY Report illustrated below.

```
          ANNUAL UPDATE REPORT - SALARIED EMPLOYEES


       DEPT            RAISE              SALARY

       901              2,204.80           24,252.80
       903              1,942.72           23,369.92
       911             18,187.10          210,058.14
       912              1,535.04           18,885.44
       914             12,497.47          141,472.19
       915                760.03            8,360.35
       917              2,559.75           29,157.27
       918              5,798.20           63,780.28
       919                706.42            8,770.62
       920              1,630.72           17,937.92
       921              3,979.82           47,778.02
       923              2,916.16           33,077.76
       924              2,360.80           27,968.80
       931                 71.76            1,789.36
       932              2,062.73           22,690.09
       935              6,148.48           67,633.28
       940             15,534.94          119,101.26
       942              7,762.56           61,512.96
       943             13,547.04          104,860.64
       944              6,121.44           47,931.04
                      108,327.98         1080,388.14
```

[LIMIT literal-6]

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

[EVERY literal-7]

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

SEQUENCE Statement

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

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

field-name

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

```
SEQUENCE DEPT LAST-NAME
```

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

CONTROL Statement

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

```
          [field-name] [NEWPAGE]
CONTROL   [          ] [       ] [NOPRINT]  ...
          [FINAL     ] [RENUM  ]
```

[field-name]
[FINAL]

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

NEWPAGE

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

RENUM

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

NOPRINT

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

TITLE Statement

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

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

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

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

[literal-1]

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

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

field-name

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

'literal-2'

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

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

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

+literal-3 or -literal-3

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

The TITLE statement:

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

produces:

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

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

COL literal-4

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

HEADING Statement

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

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

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

field-name

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

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

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

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

produces:

```
EMPLOYEE
NAME
```

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

LINE Statement

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

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

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

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

that specifies to:

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

[literal-1]

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

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

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

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

field-name

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

literal-2

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

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

+literal-3 or -literal-3

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

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

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

COL literal-4

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

POS literal-5

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

```
LINE 1                          REGION +
                                SSN    +
                                NAME   +
                                DATE-OF-BIRTH

LINE 2      POS 2               PHONE +
            POS 3               STREET +
            POS 4               DATE-OF-HIRE

LINE 3      POS 3               CITY -3 STATE -2 ZIP
```

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

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

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

The appearance of one line group is:

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

Report Procedures

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

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

Coding Techniques

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

1. You cannot use the input/output generating statements listed below:

   DLI
   GET
   IDMS
   POINT
   PRINT
   PUT
   READ
   WRITE

2. You cannot use the STOP statement.

3. Use the DISPLAY statement to perform special report annotations. Use of DISPLAY requires the following extra considerations:

   You cannot code the DISPLAY statement's file-name-1 parameter. DISPLAY is only to the associated report.

   You cannot code the HEX option of DISPLAY.

   DISPLAY lines are counted and included in the end-of-page determination. However, the ENDPAGE procedure is not invoked by these lines.

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

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

LEVEL = 0 when processing detail lines
LEVEL = n for total line processing at each control level
LEVEL = n + 1 when processing FINAL totals.

Fields contained in S storage exhibit unique properties during report processing. S fields are stored in a static working storage area. Fields in this category are not copied onto report work files. All references to S fields occur at the time the report is actually formatted and printed. Remember, the format and print operation can occur at one of two different times. With this in mind, you should use S storage fields for:

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

Special-name Report Procedures

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

REPORT-INPUT

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

BEFORE-LINE

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

AFTER-LINE

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

BEFORE-BREAK

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

AFTER-BREAK

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

ENDPAGE

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

TERMINATION

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

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

                   5/18/84     PROCEDURE  USAGE         PAGE 1

                               CTLN   CTL1   AMT1

         (BEFORE-LINE)
detail                           NA    1A      1
         (AFTER-LINE)

         (REPORT-INPUT)<---(caused by the second PRINT statement)
         (BEFORE-LINE)
detail                           NA    1A      2
```

```
         (AFTER-LINE)

         (REPORT-INPUT)<---(caused by the third PRINT statement)
         (BEFORE-BREAK)
total                             NA    1A      3
         (AFTER-BREAK)

         (BEFORE-LINE)
detail                            NA    1B      1
         (AFTER-LINE)

         (REPORT-INPUT)<---(caused by the fourth PRINT statement)
         (BEFORE-BREAK)
total                             NA    1B      1
         (AFTER-BREAK)

         (BEFORE-BREAK)
total                             NA            4
         (AFTER-BREAK)
         ...
         ...
         (ENDPAGE)


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

                  5/18/84        PROCEDURE  USAGE       PAGE 99

                                 CTLN   CTL1   AMT1

                                  ...    ...    ...
         (BEFORE-BREAK)
total                              xx    yy     ...
         (AFTER-BREAK)

         (BEFORE-BREAK)
total                              xx           ...
         (AFTER-BREAK)

         (BEFORE-BREAK)
total                                           ...
         (AFTER-BREAK)
         ...
         ...
         (ENDPAGE)

         (TERMINATION)
```

REPORT-INPUT

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

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

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

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

BEFORE-LINE and AFTER-LINE

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

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

BEFORE-BREAK

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

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

```
 1 PARM  DEBUG(FLOW FLDCHK)
 2 *
 3 FILE PERSNL FB(150 1800)
 4   NAME                    17  16   A
 5     LAST-NAME           NAME   8   A
 6   PAY-GROSS               94   4   P  2
```

```
 7   DEPT                     98   3   N
 8   DATE-OF-HIRE            136   6   N
 9     HIRE-MM   DATE-OF-HIRE      2   N
10     HIRE-DD   DATE-OF-HIRE +2   2   N
11     HIRE-YY   DATE-OF-HIRE +4   2   N
12   SALARY                   W   4   P  2
13   SERVICE                  W   2   P  0
14   CURR-DATE                S   6   N
15     CURR-MM      CURR-DATE      2   N
16     CURR-DD      CURR-DATE +2   2   N
17     CURR-YY      CURR-DATE +4   2   N
18   SENIORS                  W   2   P  0
19 *
20 JOB INPUT PERSNL
21   %GETDATE CURR-DATE
39   PERFORM SERVICE-CALC
40   IF SERVICE GT 14
41     SENIORS = 1
42   ELSE
43     SENIORS = 0
44   END-IF
45   SALARY = PAY-GROSS * 52
46   PRINT SENR-RPT
47 *
48   SERVICE-CALC. PROC
50     SERVICE = CURR-YY - HIRE-YY
51     IF CURR-MM < HIRE-MM
52       SERVICE = SERVICE - 1
53     END-IF
54     IF CURR-MM NE HIRE-MM
55       GOTO QUIT-SERV-CALC
56     END-IF
57     IF CURR-DD < HIRE-DD
58       SERVICE = SERVICE - 1
59     END-IF
60     QUIT-SERV-CALC
61   END-PROC
62 *
63 REPORT SENR-RPT LINESIZE 62 SUMMARY SPREAD NODATE NOPAGE
64   SEQUENCE DEPT
65   CONTROL DEPT
66   TITLE 1 'SERVICE UPDATE REPORT - SALARIED EMPLOYEES'
67   HEADING TALLY  ('NUMBER OF' 'EMPLOYEES' 'IN DEPT')
68   HEADING SERVICE ('AVERAGE' 'SERVICE')
69   HEADING SENIORS ('PERCENT' 'SENIORS')
70   HEADING SALARY ('TOTAL' 'SALARY')
71   LINE DEPT TALLY SENIORS SERVICE SALARY
72 *
73   BEFORE-BREAK. PROC
75     SENIORS = SENIORS * 100 / TALLY + .5
76     SERVICE = SERVICE / TALLY
77   END-PROC
78 *
```

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

AFTER-BREAK

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

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

ENDPAGE

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

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

TERMINATION

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

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

Chapter

8

File Processing

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

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

File Operations

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

Control of Input/Output

As described in the "Input/Output Specification" chapter, the easiest way to control Input/Output (I/O) is to let the system do it.

- Automatic I/O (under system control) includes the files specified for input on the JOB and SORT statements, and the files specified for output on the SORT, PRINT, and DISPLAY statements.
- Controlled I/O (under your control) includes the GET, POINT, and READ statements for input, and the PUT and WRITE statements for output.

You can use I/O control statements within a JOB activity, with or without automatic I/O, by observing the following restrictions:

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

Record Formats

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

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

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

These assumptions are made about the record formats of CARD, PUNCH, and VSAM files:

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

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

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

System-Defined Fields

Three special data fields are provided for each file:

RECORD-LENGTH

Contains one of the following:

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

RECORD-COUNT

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

FILE-STATUS

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

Error Conditions

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

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

Data Availability Tests

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

Opening and Closing Files

All files are automatically OPENed and CLOSEed.

SAM Files

Sequential Access Method (SAM) files are processed according to the following rules:

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

Input

Both automatic and controlled I/O is permitted for SAM files. The sample program uses automatic I/O exclusively. The next two exhibits illustrate how to process a SAM file using each facility:

Automatic SAM Processing

```
*FILE PERSNL FB(150 1800)
  ...
*
JOB INPUT PERSNL
  ...
*
```

Controlled SAM Processing

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

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

Output

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

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

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

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

VFM Files

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

```
*
FILE PERSNL FB(150 1800)
     EMP#       9  5  N
FILE SORTPER F 150   VIRTUAL
     UPD-EMP#   9  5  N
*
SORT PERSNL TO SORTPER USING EMP#
*
JOB INPUT SORTPER
*
```

SORTPER is a virtual file. You do not have to define it in the JCL since it is actually stored and retrieved by VFM from storage.

ISAM Files

Indexed Sequential Access Method (ISAM) files are processed as input only. You can perform sequential, skip-sequential, or random processing on these files.

Sequential Processing

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

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

Skip-Sequential Processing

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

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

Random Processing

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

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

VSAM Files

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

VSAM Files	Description
ESDS	Entry-sequenced data set
KSDS	Key-sequenced data set
RRDS	Relative-record data set.

You must identify your VSAM file organization before coding your program.

File Loading

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

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

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

Input

VSAM input files are processed the same as ISAM files. You can perform sequential, skip-sequential, or random processing on VSAM input files.

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

```
FILE PAYMSTR VS
     SALARY  94  4  P  2
*
JOB INPUT PAYMSTR
  SALARY = SALARY * 1.1
  PRINT SAL-RPT
*
```

- Skip-sequential processing enables you to specify one or more records in the input file that are not processed. They are skipped and processing continues with the following records. The next exhibit illustrates skip-sequential processing.

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

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

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

Record Addition

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

VSAM Single Record Addition

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

VSAM Mass-Sequential Record Addition

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

Record Deletion

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

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

Record Update

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

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

Synchronized File Processing

CA-Easytrieve Plus simplifies combining the data from more than one file. It has the capacity to synchronize any number of files that can be processed sequentially. Synchronizing more than two files necessitates a high level of data processing expertise and a comprehensive understanding of file structures. If your application requires complex synchronized file processing, refer to the *Reference Guide* for detailed information.

This chapter of the *Application Guide* describes a match/merge operation using two input files with one key each. The code for a sample program to accomplish this task is illustrated in the next exhibit. Subsequent paragraphs present detailed discussions of the rules to be followed in specifying input to synchronized file processing and techniques for determining file relationships, using the code in the next exhibit, as illustrated.

```
 1 PARM  DEBUG(FLOW FLDCHK)
 2 *
 3 FILE PERSNL FB(150 1800)
 4      OLD-EMP#          9   5   N
 5 *
 6 FILE PERSUPD CARD
 7      EMP#              1   5   N
 8      RAISE-PERCENT     7   2   N
 9 *
10 FILE SORTPER F 150   VIRTUAL
11      UPD-EMP#          9   5   N
12      NAME             17   8   A
13      PAY-GROSS        94   4   P  2
14      NEWSAL            W   4   P  2
15 *
16 FILE NEWPERS FB(150 1800)
17 *
18 FILE ERRPRINT  PRINTER
19 *
20 *
21 SORT PERSNL TO SORTPER USING OLD-EMP#
22 *
23 JOB INPUT (SORTPER KEY(UPD-EMP#) +
              PERSUPD KEY(EMP#) )
24 *
25   IF MATCHED
26     NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)
27     PRINT NEW-RPT
28     PAY-GROSS = NEWSAL
29   END-IF
30   IF SORTPER
31     PUT NEWPERS FROM SORTPER
32   ELSE
33     DISPLAY ERRPRINT EMP# 'RECORD NOT MATCHED'
34   END-IF
35 *
36 REPORT  NEW-RPT  LINESIZE 80 NOPAGE NODATE
37   SEQUENCE  NAME
38   TITLE 1   'SALARY UPDATE REPORT'
39   TITLE 2   'EMPLOYEES WITH OVER 25 YEARS SERVICE'
40   HEADING   UPD-EMP#       ('EMPL' 'NUMBER')
41   HEADING   NAME           ('EMPL' 'NAME')
42   HEADING   PAY-GROSS      ('OLD' 'SALARY')
43   HEADING   NEWSAL         ('NEW' 'SALARY')
44   HEADING   RAISE-PERCENT  ('RAISE' '%')
45   LINE      UPD-EMP#  NAME  PAY-GROSS  NEWSAL  RAISE-PERCENT
46 *
47 END
   01730 08
   04225 09
   09481 09
   11473 11
   11710 10
   12267 12
```

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

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

```
                SALARY UPDATE REPORT
        EMPLOYEES WITH OVER 25 YEARS SERVICE


 EMPL       EMPL        OLD          NEW       RAISE
NUMBER      NAME       SALARY       SALARY       %

11473     BERG          759.20       842.71     11
04225     LOYAL         295.20       321.76     09
09481     OSMON         628.00       684.52     09
11710     POWELL        243.20       267.52     10
01730     SMOTH         315.20       340.41     08
12267     WIMN          373.60       418.43     12
```

Synchronized files are subject to the following rules:

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

  ```
  SORT PERSNL TO SORTPER USING OLD-EMP#
  ```

  SORTPER is then input to the JOB:

  ```
  JOB INPUT (SORTPER KEY(UPD-EMP#) +
                PERSUPD KEY(EMP#) )
  ```

  The key can be any defined field in the named file, while the file is in order by the value of the contents of the specified key field.

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

Input

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

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

Conditional Expressions

In synchronized file processing, you need to know:

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

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

File Presence Condition

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

```
IF  [NOT] [EOF]  file-name
```

The condition tests true if a record is available, such as:

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

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

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

File Presence Series Condition

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

```
IF  [NOT]  MATCHED
```

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

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

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

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

Record Relational Condition

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

```
                DUPLICATE
IF  [NOT]       FIRST-DUP    file-name
                LAST-DUP
```

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

DUPLICATE

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

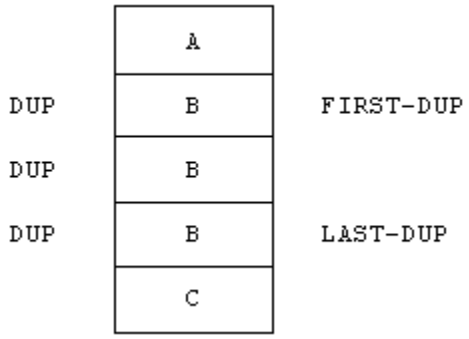
FIRST-DUP

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

LAST-DUP

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

The next exhibit illustrates how these condition tests work.



Chapter

9

Table Processing

This chapter describes table processing in CA-Easytrieve Plus.

Table Definition

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

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

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

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

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

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

Be careful when setting your table value. CA-Easytrieve Plus will allocate space for the table using the following formula:

```
LRECL * the number of table entries = The amount of allocated storage
```

For example, if you specify a table value of 600000, approximately 21 MB of real core storage will be allocated for the table, regardless of the actual number of table entries. If the table has 20000 entries, the amount needed is approximately 720,000 bytes. This is considerably smaller than the 21 MB reserved by CA-Easytrieve Plus.

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

Instream Tables

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

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

External Tables

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

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

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

SEARCH Statement

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

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

file-name

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

WITH field-name-1

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

GIVING field-name-2

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

The named TABLE file is searched for an ARGument whose value is the same as the value of field-name-1. If a match is found, the content of field-name-2 is set to the value of the DESCription associated with the ARGument. The content of field-name-2 is not changed if a match for field-name-1 is not found in the named TABLE file. An IF statement with a file presence condition (see the "File Processing" chapter) can be coded after the SEARCH statement to determine the success of the table search.

You can code SEARCH statements any place within a JOB activity, SORT procedure, or REPORT procedure. The next exhibit illustrates the retrieval of names of the days of the week based on numeric identification codes.

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

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

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

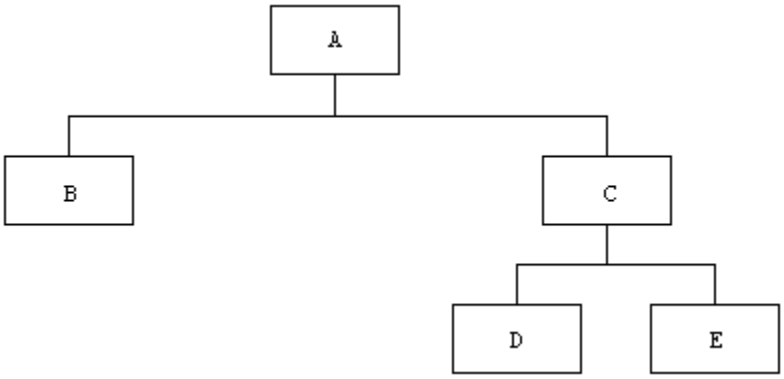
Chapter

10 IMS/DLI Processing

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

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

Hierarchical Database Structure



This chapter briefly describes the statements that define database processing:

Statement	Description
FILE	Identifies the database.
RECORD	Identifies the database segments available for processing.
RETRIEVE	Describes automatic database input.

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

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

FILE Statement

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

The FILE statement (see the "Library" chapter) identifies the database by specifying DLI as the file-type parameter and by identifying the PCB to be processed as follows:

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

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

RECORD Statement

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

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

segment-name-1

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

literal-1

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

segment-name-2

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

[KEY]

The KEY parameter is required for:

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

KEY is optional for:

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

field-name

This is the one- to eight- character name used to designate the key field to the IMS/DLI database. The name must correspond to a field named in the segment in the DBD.

literal-2

Literal-2 is a numeric integer that specifies the location of the key within the segment.

literal-3

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

RETRIEVE Statement

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

```
RETRIEVE  file-name-1 +

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

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

file-name-1

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

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

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

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

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

SELECT

The SELECT parameter identifies which segments (record-name) are retrieved.

record-name

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

[ID literal-1]

ID literal-1 is an optional two-byte alphabetic field that identifies retrieved paths. For example, in the Hierarchical Database Structure exhibit (shown earlier), one path might be designated AB; another as AD.

- The AB path includes two segments: A and B.
- The AD path includes three segments: A, C, and D.

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

[LIMIT literal-2]

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

[WHILE (condition)]

WHILE (condition) optionally pre-screens input segments. The syntax of the WHILE condition is exactly the same as the conditional expressions discussed in the "Decision and Branching Logic" chapter.

As the associated segment is returned by IMS/DLI, the WHILE condition is evaluated. Segments are accepted for input only if the WHILE condition is true.

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

Automatic Input with RETRIEVE

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

Sweep of a Database

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

Tickler File Control

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

Input Definition (Paths)

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

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

Chapter

11

OS/390 and z/OS JCL

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

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

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

Material in uppercase letters is required.

Sample Short Report Output Program

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

```
//jobname  JOB (acctng.info),your.name
//stepname EXEC  PGM=EZTPA00
//STEPLIB   DD DSN=your.load.library,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SORTWK01  DD UNIT=SYSDA,SPACE=(4096,500,,,ROUND)
//SYSOUT    DD SYSOUT=A
//PERSNL    DD DSN=your.input.filename,DISP=SHR
//PANDD1    DD DSN=your.macro.library,DISP=SHR
//SYSIN     DD *
PARM  DEBUG(FLOW)
*
FILE PERSNL FB(150 1800)
  NAME                    17  16   A
    LAST-NAME           NAME   8   A
  PAY-GROSS               94   4   P  2
  DEPT                    98   3   N
*
JOB INPUT PERSNL
  IF DEPT = 900 THRU 911
    PRINT SHORT-RPT
  END-IF
*
REPORT SHORT-RPT LINESIZE 60 SPREAD NODATE NOPAGE
  SEQUENCE DEPT
  CONTROL DEPT
  TITLE 1 'SALARY REPORT'
  TITLE 2 'DEPARTMENTS 900 - 911'
  HEADING PAY-GROSS ('TOTAL' 'SALARY')
  HEADING LAST-NAME ('EMPLOYEE' 'NAME')
  LINE DEPT LAST-NAME PAY-GROSS
/*
```

```
                SALARY REPORT
              DEPARTMENTS 900 - 911


               EMPLOYEE              TOTAL
DEPT             NAME               SALARY

901            WALTERS                 424.00
901                                    424.00

903            WIMN                    373.60
903                                    373.60

911            HAFER                   121.95
               REYNOLDS                174.15
               KRUSE                   242.40
               POWELL                  243.20
               LARSON                  283.92
               POST                    292.00
               ISAAC                   313.60
               YOUNG                   313.60
               SMOTH                   315.20
               GREEN                   365.60
               STRIDE                  386.40
               ARNOLD                  445.50
911                                  3,497.52

                                     4,295.12
```

Mailing Label Output Program

The label-generating program discussed in the "Report Processing" chapter reads one input file (PERSNL) and outputs a set of mailing labels. The JCL and code for this program are illustrated in the next exhibit.

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

```
  CURR-DATE                S   6   N
    CURR-MM      CURR-DATE      2   N
    CURR-DD      CURR-DATE +2   2   N
    CURR-YY      CURR-DATE +4   2   N
*
JOB INPUT PERSNL
  %GETDATE CURR-DATE
  PERFORM SERVICE-CALC
  IF SERVICE GT 19
    PRINT MAILOUT
  END-IF
*
  SERVICE-CALC. PROC
    SERVICE = CURR-YY - HIRE-YY
    IF CURR-MM < HIRE-MM
      SERVICE = SERVICE - 1
    END-IF
    IF CURR-MM NE HIRE-MM
      GOTO QUIT-SERV-CALC
    END-IF
    IF CURR-DD < HIRE-DD
      SERVICE = SERVICE - 1
    END-IF
    QUIT-SERV-CALC
  END-PROC
*
REPORT MAILOUT LABELS (ACROSS 2  DOWN 4  SIZE 30)
  SEQUENCE LAST-NAME
  LINE 1 FIRST-NAME -3 LAST-NAME
  LINE 2 STREET
  LINE 3 CITY  -3 STATE  ZIP
/*
```

Synchronized File Processing Program

The synchronized file processing program discussed in the "File Processing" chapter reads two input files (PERSNL and PERSUPD), sorts one file (PERSNL) to another file (SORTPER), outputs one printed report, displays messages to an error file (ERRPRINT), and creates a new master file (NEWPERS). The JCL and code for this program are illustrated in the next exhibit.

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

```
FILE PERSUPD CARD
     EMP#              1   5   N
     RAISE-PERCENT     7   2   N
*
FILE SORTPER F 150   VIRTUAL
     UPD-EMP#          9   5   N
     NAME             17   8   A
     PAY-GROSS        94   4   P  2
     NEWSAL            W   4   P  2
*
 FILE NEWPERS FB(150 1800)
 *
 FILE ERRPRINT  PRINTER
 *
 SORT PERSNL TO SORTPER USING OLD-EMP#
 *
 *
 JOB INPUT (SORTPER KEY(UPD-EMP#) +
            PERSUPD KEY(EMP#) )
 *
  IF MATCHED
    NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)
    PRINT NEW-RPT
    PAY-GROSS = NEWSAL
  END-IF
  IF SORTPER
    PUT NEWPERS FROM SORTPER
  ELSE
    DISPLAY ERRPRINT EMP# ' RECORD NOT FOUND'
  END-IF
*
REPORT  NEW-RPT  LINESIZE 80 NOPAGE NODATE
  SEQUENCE  NAME
  TITLE 1   'SALARY UPDATE REPORT'
  TITLE 2   'EMPLOYEES WITH OVER 25 YEARS SERVICE'
  HEADING   UPD-EMP#       ('EMPL' 'NUMBER')
  HEADING   NAME           ('EMPL' 'NAME')
  HEADING   PAY-GROSS      ('OLD' 'SALARY')
  HEADING   NEWSAL         ('NEW' 'SALARY')
  HEADING   RAISE-PERCENT  ('RAISE' '%')
  LINE      UPD-EMP# NAME PAY-GROSS NEWSAL RAISE-PERCENT
*
END
01730 08
04225 09
09481 09
11473 11
11710 10
12267 12
/*
```

SORTPER, the sort output file, is not defined in the JCL because it is a temporary VIRTUAL file. PERSUPD, the input employee number file, is also not defined in the JCL, since it is a CARD file whose data is obtained from the records after the END statement following the program.

Compile and Link-Edit Load Module

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

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

Previously Compiled and Link-Edited Programs

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

```
//jobname  JOB (acctng.info),your.name
//stepname EXEC  PGM=TESTPGM
//STEPLIB   DD DSN=your.load.library,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//EZTVFM    DD UNIT=SYSDA,SPACE=(4096,(100,200),,,ROUND)
//SYSOUT    DD SYSOUT=A
//SORTWK01  DD UNIT=SYSDA,SPACE=(4096,500,,,ROUND)
//filename  DD DSN=your.input.file,DISP=SHR
//SYSIN     DD * (optional card input)
```

Chapter

12 VSE JCL

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

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

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

Material in uppercase letters is required.

Sample Short Report Output Program

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

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

```
                    SALARY REPORT
                 DEPARTMENTS 900 - 911


                 EMPLOYEE              TOTAL
DEPT               NAME               SALARY

901              WALTERS                424.00
901                                     424.00

903              WIMN                   373.60
903                                     373.60

911              HAFER                  121.95
                 REYNOLDS               174.15
                 KRUSE                  242.40
                 POWELL                 243.20
                 LARSON                 283.92
                 POST                   292.00
                 ISAAC                  313.60
                 YOUNG                  313.60
                 SMOTH                  315.20
                 GREEN                  365.60
                 STRIDE                 386.40
                 ARNOLD                 445.50
911                                   3,497.52

                                      4,295.12
```

Mailing Label Output Program

The label-generating program discussed in the "Report Processing" chapter reads one input file (PERSNL) and outputs a set of mailing labels. The JCL and code for this program are illustrated in the next exhibit.

```
*  $$ JOB JNM=jobname
//  JOB    jobname
//  DLBL   EZTP,'your.eztp.sysclb',0,SD
//  EXTENT SYS003,volser,1,0,start,lgth
//  ASSGN  SYS003,nnn
//  LIBDEF CL,SEARCH=EZTP,TEMP
//  ASSGN  SYS001,nnn
//  ASSGN  SYS006,nnn
//  ASSGN  SYS008,nnn
//  ASSGN  SYS010,nnn
//  DLBL   SORTWK1,,0,DA
//  EXTENT SYS001,volser,,,start,lgth
//  DLBL   PANDD1,'your.macro.library',0,SD
//  EXTENT SYS006,volser,,,start,lgth
//  DLBL   PERSNL,'your.input.filename',0,SD
//  EXTENT SYS008,volser,,,start,lgth
//  DLBL   EZTVFM,,0,SD
//  EXTENT SYS010,volser,,,start,lgth
//  EXEC   EZTPA00,SIZE=200K
PARM  DEBUG(FLOW)
*
```

```
FILE PERSNL FB(150 1800)
  NAME                     17  16   A
    LAST-NAME              17   8   A
    FIRST-NAME             25   8   A
  ADDRESS                  37  39   A
    STREET                 37  20   A
    CITY                   57  12   A
    STATE                  69   2   A
    ZIP                    71   5   N
  DATE-OF-HIRE            136   6   N
    HIRE-MM               136   2   N
    HIRE-DD               138   2   N
    HIRE-YY               140   2   N
  SERVICE                   W   2   N
  CURR-DATE                 S   6   N
    CURR-MM      CURR-DATE      2   N
    CURR-DD      CURR-DATE +2   2   N
    CURR-YY      CURR-DATE +4   2   N
*
JOB INPUT PERSNL
  %GETDATE CURR-DATE
  PERFORM SERVICE-CALC
  IF SERVICE GT 19
    PRINT MAILOUT
  END-IF
*
  SERVICE-CALC. EZTC
    SERVICE = CURR-YY - HIRE-YY
    IF CURR-MM < HIRE-MM
      SERVICE = SERVICE - 1
    END-IF
    IF CURR-MM NE HIRE-MM
      GOTO QUIT-SERV-CALC
    END-IF
    IF CURR-DD < HIRE-DD
      SERVICE = SERVICE - 1
    END-IF
    QUIT-SERV-CALC
  END-EZTC
*
REPORT MAILOUT LABELS (ACROSS 2  DOWN 4  SIZE 30)
  SEQUENCE LAST-NAME
  LINE 1 FIRST-NAME -3 LAST-NAME
  LINE 2 STREET
  LINE 3 CITY  -3 STATE  ZIP
/*
/&
* $$ EOJ
```

Synchronized File Processing Program

The synchronized file processing program discussed in the "File Processing" chapter reads two input files (PERSNL and PERSUPD), sorts one file (PERSNL) to another file (SORTPER), outputs one printed report, displays messages to an error file (ERRPRINT), and creates a new master file (NEWPERS). The JCL and code for this program are illustrated in the next exhibit.

```
*  $$ JOB JNM=jobname
//  JOB    jobname
//  DLBL   EZTP,'your.eztp.sysclb',0,SD
//  EXTENT SYS003,volser,1,0,start,lgth
//  ASSGN  SYS003,nnn
//  LIBDEF CL,SEARCH=EZTP,TEMP
//  ASSGN  SYS001,nnn
//  ASSGN  SYS006,nnn
//  ASSGN  SYS008,nnn
//  ASSGN  SYS010,nnn
//  ASSGN  SYS011,SYSLST
//  DLBL   SORTWK1,,0,DA
//  EXTENT SYS001,volser,,,start,lgth
//  DLBL   PANDD1,'your.macro.library',0,SD
//  EXTENT SYS006,volser,,,start,lgth
//  DLBL   PERSNL,'your.input.filename',0,SD
//  EXTENT SYS006,volser,,,start,lgth
//  DLBL   NEWPERS,'your.new.filename',0,SD
//  EXTENT SYS009,volser,,,start,lgth
//  DLBL   EZTVFM,,0,SD
//  EXTENT SYS010,volser,,,start,lgth
//  EXEC   EZTPA00,SIZE=200K
PARM  DEBUG(FLOW)
*
FILE PERSNL FB(150 1800)
     OLD-EMP#          9   5   N
*
FILE PERSUPD CARD
     EMP#              1   5   N
     RAISE-PERCENT     7   2   N
*
FILE SORTPER F 150   VIRTUAL
     UPD-EMP#          9   5   N
     NAME             17   8   A
     PAY-GROSS        94   4   P  2
     NEWSAL            W   4   P  2
*
 FILE NEWPERS FB(150 1800)
 *
FILE ERRPRINT  PRINTER
*
SORT PERSNL TO SORTPER USING OLD-EMP#
*
*
```

```
JOB INPUT (SORTPER KEY(UPD-EMP#) +
           PERSUPD KEY(EMP#) )
*
  IF MATCHED
    NEWSAL = PAY-GROSS * (1 + RAISE-PERCENT / 100)
    PRINT NEW-RPT
    PAY-GROSS = NEWSAL
  END-IF
  IF SORTPER
    PUT NEWPERS FROM SORTPER
  ELSE
    DISPLAY ERRPRINT EMP# ' RECORD NOT FOUND'
  END-IF
*
REPORT  NEW-RPT  LINESIZE 80 NOPAGE NODATE
  SEQUENCE  NAME
  TITLE 1   'SALARY UPDATE REPORT'
  TITLE 2   'EMPLOYEES WITH OVER 25 YEARS SERVICE'
  HEADING   UPD-EMP#       ('EMPL' 'NUMBER')
  HEADING   NAME           ('EMPL' 'NAME')
  HEADING   PAY-GROSS      ('OLD' 'SALARY')
  HEADING   NEWSAL         ('NEW' 'SALARY')
  HEADING   RAISE-PERCENT  ('RAISE' '%')
  LINE      UPD-EMP# NAME PAY-GROSS NEWSAL RAISE-PERCENT
*
END
01730 08
04225 09
09481 09
11473 11
11710 10
12267 12
/*
/&
* $$ EOJ
```

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

Compile and Link-Edit Load Module

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

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

Previously Compiled and Link-Edited Programs

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

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

Chapter

13 Applications

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

- The first form processes pre-existing data files. The "Basic Examples" and "Advanced Techniques" chapters present examples of reports generated from two sample files: a Personnel Master File and an Inventory Master File.
- The second form implements entire application systems. The "Bank System" and "Project Management System" chapter contain mini-applications that demonstrate the wide scope of capabilities.

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

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

Application Overview

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

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

Each of these applications has its own master and auxiliary files. Each file structure is defined in the description of the particular system.

In general, the applications presented are not intended to be usable in a real-world environment (except possibly the Project Management System). They are intended to provide you with ideas for developing your own programs and systems.

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

Program Formatting Standards

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

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

Program Output Standards

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

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

It is a good practice to TITLE your reports with something meaningful and to include the current date in the title. Most installations have report format standards; our reports should be compatible with your standards.

Inventory Sample File

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

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

*
  VENDOR-INFO           104  17 A
    VENDOR-NUMBER                 104  8 N   MASK '99-99-9-999' -
                                              HEADING('VENDOR' 'NUMBER')
    VENDOR-LOCATION-CITY          112  7 A   HEADING('VENDOR' 'CITY')
    VENDOR-LOCATION-STATE         119  2 A   HEADING('VENDOR' 'STATE')
*
  SHIPPING-INFO         121   6 A
    SHIPPING-FOB-CODE             121  2 P   HEADING('FOB' 'CODE')
    SHIPPING-CARRIER-ALPHA-CODE   123  4 A   HEADING('CARRIER' 'CODE')
*
```

Personnel Sample File

The Personnel sample file is the basis for many of the examples in the following chapters. The following JOB lists the contents of the file. The field definitions are imbedded as macro %PERSNL. These field definitions are not repeated in the examples. Refer to this chapter when studying an example that 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   16 A     HEADING 'EMPLOYEE NAME'
14     NAME-LAST  NAME     8 A     HEADING('LAST' 'NAME')
15     NAME-FIRST NAME +8  8 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 *
53   LINE 1            REGION          -
                       BRANCH          -
                       SSN             -
                       EMP#            -
                       NAME            -
                   +2  PAY-GROSS       -
                       DEPT            -
                       DATE-OF-BIRTH   -
                       SEX
54   LINE 2  POS 3 -1  TELEPHONE       -
             POS 5     ADDR-STREET     -
             POS 6     PAY-NET         -
             POS 7 +1  JOB-CATEGORY    -
             POS 8     DATE-OF-HIRE    -
             POS 9 +1  MARITAL-STAT
55   LINE 3  POS 5     ADDR-CITY       -
                   -1  ADDR-STATE      -
                       ADDR-ZIP        -
             POS 7 +1  SALARY-CODE


12/03/83               NEW PERSONNEL SAMPLE FILE LISTING          PAGE      1

 R      SOCIAL SECURITY                                 PAY -    DPT/ DATE OF
 G          NUMBER/      EMPL                           GROSS/   J*C/  BIRTH/  SEX/
 N BRCH    TELEPHONE    NUMBER  EMPLOYEE NAME            NET     S*C    HIRE   M/S

 1  01    025-30-5228   12267  WIMN    GLORIA          373.60  903   5/22/30  1
       (617) 332-2762          430 M ST SW 107         251.65   10   7/12/51  S
                               BOSTON      MA 02005             01

 1  02    121-16-6413   11473  BERG    NANCY           759.20  943   8/15/31  1
       (301) 636-8995          3710 JENIFER ST N W     547.88   25   6/17/55  M
                               BALTIMORE   MD 21055             03

 1  03    228-58-8307   02688  CORNING GEORGE          146.16  915  10/12/52  2
       (609) 444-7688          3208 S 5TH              103.43   40  11/08/70  S
                               TRENTON     NJ 08535             06

 1  02    256-52-8737   00370  NAGLE   MARY            554.40  935   1/13/43  1
       (301) 636-8995          826 D STREET SE         340.59   10   3/18/73  S
                               BALTIMORE   MD 21034             01

 1  04    281-36-2873   01963  ARNOLD  LINDA           445.50  911   8/29/42  1
       (212) 451-4040          1569 COLONIAL TERR      356.87   10  10/19/68  S
                               NEW YORK    NY 10012             01

 1  03    298-34-4755   11602  MANHART VIRGINIA        344.80  914  10/13/38  1
       (609) 444-3094          1305 POTOMAC ST N W     250.89   60   7/06/65  S
                               TRENTON     NJ 08521             08

 1  04    322-30-0050   11931  TALL    ELAINE          492.26  917  12/25/40  1
       (212) 451-4531          1412 36TH ST NW         355.19   40  10/16/62  S
                               NEW YORK    NY 10091             06

 1  01    452-52-1419   02200  BRANDOW LYDIA           804.64  918   9/14/47  1
       (617) 332-6701          3616 B ST S E           554.31   10   7/11/72  S
                               BOSTON      MA 02011             01

 1  04    554-70-3189   11357  LARSON  RODNEY          283.92  911   2/11/39  2
       (212) 451-7382          610 H ST SW             215.47   25   8/20/66  S
                               NEW YORK    NY 10059             03
```

```
12/03/83              NEW PERSONNEL SAMPLE FILE LISTING          PAGE      2

 R      SOCIAL SECURITY                                PAY -    DPT/ DATE OF
 G          NUMBER/      EMPL                         GROSS/   J*C/  BIRTH/  SEX/
 N BRCH    TELEPHONE    NUMBER  EMPLOYEE NAME          NET     S*C    HIRE   M/S

 1  04    577-20-0461   11467  BYER    JULIE         396.68  932   4/17/39  1
       (212) 773-0799          3400 NORTH 18TH STR   259.80   10   8/25/69  M
                               NEW YORK    NY 10071           01

 2  01    579-12-0813   11376  HUSS    PATTI         360.80  921   3/13/42  1
       (813) 796-1189          1355 TEWKESBURY PLA   223.71   10   4/01/60  M
                               CLEARWATER  FL 33512           01

 2  02    579-50-4818   11710  POWELL  CAROL         243.20  911  11/10/37  1
       (404) 832-8081          5023 AMES STREET N    167.96   10   6/07/55  S
                               ATLANTA     GA 30316           03

 2  03    008-28-7725   04234  MCMAHON BARBARA       386.40  943   6/09/42  1
       (202) 715-0404          1318 24TH STREET S    283.19   25   8/21/62  M
                               WASHINGTON  DC 20015           03

 2  03    120-32-5734   03416  FORREST BILL           13.80  931  12/08/45  2
       (202) 715-0389          1545 18TH ST NW        13.19   23   7/17/63  M
                               WASHINGTON  DC 20018           01

 2  04    190-32-2101   00445  POST    JEAN          292.00  911   2/15/45  1
       (904) 986-0034          1250 4TH ST SW        206.60   10   5/11/69  S
                               JACKSONVILLEFL 32052           06

 2  03    212-48-5461   00577  PETRIK  KATHY         220.80  921   3/17/41  1
       (202) 715-1914          5005 BENTON AVE       154.70   10   6/29/60  S
                               WASHINGTON  DC 20032           08
```

12/03/83 NEW PERSONNEL SAMPLE FILE LISTING PAGE 3

R G N	BRCH	SOCIAL SECURITY NUMBER/ TELEPHONE	EMPL NUMBER	EMPLOYEE NAME	PAY - GROSS/ NET	DPT/ J*C/ S*C	DATE OF BIRTH/ HIRE	SEX/ M/S
2	05	235-72-1049	01895	VETTER DENISE	279.36	914	7/25/37	1
		(919) 489-1614		7311 KEYSTONE LANE	189.06	40	9/15/50	M
				RALEIGH NC 27591		06		
2	02	284-36-5652	03571	KRUSE MAX	242.40	911	1/01/42	2
		(404) 832-1776		2161 N PIERCE STREE	182.09	10	6/17/60	M
				ATLANTA GA 30345		01		
2	05	310-44-5370	04225	LOYAL NED	295.20	912	5/06/19	2
		(919) 489-5531		17 KENNEDY STREET	230.50	60	8/18/53	M
				RALEIGH NC 27516		08		
2	03	362-48-0393	02765	DENNING RALPH	135.85	919	11/12/49	2
		(202) 715-1832		1629 16TH ST NW APT	109.60	25	4/19/66	S
				WASHINGTON DC 20005		03		
3	01	570-10-5594	04132	WEST KATHY	736.00	940	4/04/33	1
		(816) 581-1352		1728 IRVING ST N W	429.62	60	6/27/64	S
				KANSAS CITY KS 66015		08		
3	02	577-09-1160	01743	THOMPSON JANICE	250.40	923	6/23/32	1
		(214) 941-1441		7752 EMERSON RD	187.40	10	5/23/70	S
				DALLAS TX 75235		01		
3	02	578-38-7587	01730	SMOTH CINDY	315.20	911	5/21/38	1
		(214) 941-1585		4120 18TH STREET NE	202.43	25	9/20/55	M
				DALLAS TX 75219		03		
3	03	578-54-3178	03936	NORIDGE DEBBIE	324.00	944	3/02/36	1
		(312) 646-0934		4264 E CAPITOL NE	242.25	30	8/19/68	S
				CHICAGO IL 60652		05		

```
12/03/83              NEW PERSONNEL SAMPLE FILE LISTING          PAGE      4

 R      SOCIAL SECURITY                                 PAY -    DPT/ DATE OF
 G          NUMBER/      EMPL                          GROSS/   J*C/  BIRTH/  SEX/
 N BRCH    TELEPHONE    NUMBER  EMPLOYEE NAME           NET     S*C    HIRE   M/S

 3  03    579-50-4170   01549  ROGERS  PAT            329.00  924   4/19/41  1
       (312) 646-1650          1625 FRANKLIN ST N     230.17   30  10/10/61  M
                               CHICAGO     IL 60691            05

 3  04    208-28-2315   12829  GREEN   BRENDA         365.60  911   7/21/28  1
       (612) 588-1900          2671 DOUGLAS PL S E    238.04   20   4/17/69  M
                               MINNEAPOLIS MN 55319            02

 3  01    231-68-9995   12403  KELLY   KEITH          197.60  940   9/11/34  2
       (816) 581-0031          211 E GLEBE RD #C      145.51   60   5/07/70  S
                               KANSAS CITY KS 66021            08

 3  02    418-46-1872   12641  ISAAC   RUTH           313.60  911   6/28/29  1
       (214) 941-0558          2639 15TH ST NW 305    219.91   10   9/15/65  S
                               DALLAS      TX 75213            01

 3  03    467-56-4149   03890  STRIDE  ANN            386.40  911   2/29/39  1
       (312) 646-1891          325 C STREET SE APT    272.53   10   9/08/68  S
                               CHICAGO     IL 60619            01

 3  04    477-44-4948   12318  MALLOW  TERRY          282.40  942   9/11/39  2
       (612) 588-8991          2515 K STREET NW AP    195.13   10   7/08/59  S
                               MINNEAPOLIS MN 55329            01

 3  02    215-36-5852   09609  LACH    LORRIE         310.40  923   7/19/41  1
       (214) 986-1901          3419 LORRING DRIVE     215.91   25   7/19/66  S
                               DALLAS      TX 75218            03

 3  03    228-46-5157   07781  EPERT   LINDA          310.40  918   4/13/44  1
       (312) 588-5118          1440 ROCK CREEK FOR    224.36   10  12/15/70  S
                               CHICAGO     IL 60609            01
```

```
12/03/83              NEW PERSONNEL SAMPLE FILE LISTING          PAGE      5

 R      SOCIAL SECURITY                              PAY -    DPT/ DATE OF
 G          NUMBER/      EMPL                        GROSS/   J*C/  BIRTH/  SEX/
 N BRCH    TELEPHONE    NUMBER  EMPLOYEE NAME         NET     S*C    HIRE   M/S

 3  03    269-24-7428   09481  OSMON   SAMUEL        628.00  935   6/13/42  2
       (312) 588-8995          4201 CATHEDRAL AVE    411.05   20   7/22/50  M
                               CHICAGO     IL 60618           02

 3  02    388-18-6119   07231  GRECO   LESLIE      1,004.00  914   3/19/45  1
       (214) 399-7688          2195 CANTERBURY WAY   685.23   60   6/28/63  S
                               DALLAS      TX 75227           08

 3  04    577-16-2985   08262  CROCI   JUDY          376.00  914   5/22/43  1
       (553) 444-1970          1606 C ST NE          215.95   60   9/23/64  S
                               MINNEAPOLIS MN 55339           08

 3  02    061-30-8680   05805  REYNOLDS   WILLIAM    174.15  911  11/03/31  2
       (214) 399-4040          4126 CROSSWICK TURN   134.03   10   6/09/61  S
                               DALLAS      TX 75244           01

 3  01    090-22-9192   05807  PHILPS  SUE           253.26  940  12/09/48  1
       (816) 836-3094          2954 NORTHAMPTON ST   213.76   40   8/10/73  S
                               KANSAS CITY KS 66031           06

 3  01    118-34-8805   04589  YOUNG   JANE          313.60  911   7/08/43  1
       (816) 836-4531          1545 18 STREET NW     229.69   10  11/20/70  M
                               KANSAS CITY KS 66054           01

 3  03    140-32-0779   05914  MILLER  JOAN          313.60  920   1/27/47  1
       (312) 588-6701          3 POOKS HILL RD APT   222.61   10  10/02/74  M
                               CHICAGO     IL 60643           01

 4  01    216-44-7756   05525  TALUS   RUTH          460.80  944  10/06/33  1
       (206) 225-3828          9331 CAROLINE AVENU   279.56   25   6/23/66  S
                               SEATTLE     WA 98003           03
```

```
12/03/83              NEW PERSONNEL SAMPLE FILE LISTING          PAGE      6

 R      SOCIAL SECURITY                                 PAY -    DPT/ DATE OF
 G          NUMBER/      EMPL                          GROSS/   J*C/  BIRTH/  SEX/
 N BRCH    TELEPHONE    NUMBER  EMPLOYEE NAME           NET     S*C    HIRE   M/S

 4  02    484-30-8293   06239  JOHNSON LISA           712.80  942   2/05/46  1
       (714) 771-0799          806 CONNECTICUT AVE    451.92   10   4/21/64  S
                               SAN DIEGO   CA 92045             01

 4  01    577-58-0363   05482  WARD    MARINA         183.75  921   9/12/23  1
       (206) 225-9127          1725 H ST NE APT 2     141.47   10   5/28/69  M
                               SEATTLE     WA 98015             01

 4  03    579-62-1768   04935  ZOLTAN  JANET          125.00  924   8/18/28  1
       (213) 493-5966          2026 FORT DAVIS ST      25.00   40   9/14/68  S
                               LOS ANGELES CA 90091             06

 4  03    060-26-8978   10949  JONES   ALFRED         804.80  940  11/13/39  2
       (213) 493-0979          2070 BELMONT ROAD N    560.63   40   1/15/59  S
                               LOS ANGELES CA 90052             06

 4  02    104-20-0956   09764  HAFER   ARTHUR         121.95  911   8/03/40  2
       (714) 771-9876          806 CONNECTICUT AVE     96.64   60  10/07/70  M
                               SAN DIEGO   CA 92031             08

 4  04    448-24-6593   10260  JUDAR   PAULA          591.20  943  10/12/50  1
       (415) 278-1753          4333 46TH ST N W       459.57   25   5/06/65  M
                               SAN FRANCISCCA 94041             03

 4  03    537-03-4039   11211  WALTERS KAREN          424.00  901   1/10/42  1
       (213) 725-6495          1022 5 KENSINGTON P    282.45   10   3/17/71  M
                               LOS ANGELES CA 90030             01

 4  01    558-44-7609   10961  RYAN    PAMELA         399.20  914  12/12/49  1
       (206) 225-8456          1717 R N W #301        291.70   10   8/08/70  S
                               SEATTLE     WA 98009             01
```

Chapter

14

Basic Examples

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

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

The output for each job is typically some form of report. A wide variety of reports is printed to give you an idea of what can be done. For some examples, the volume of output has been condensed.

The remainder of this chapter is composed of the examples. Each example is presented in the format described in the "Applications" chapter.

Employees in Region 1

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

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

```
 1 *
 2 *   EXAMPLE 14.1
 3 *
 4 FILE  PERSNL   FB(150 1800)
 5 %PERSNL
35 *
36 *
37 JOB
38   IF  REGION = 1
39     PRINT
40   END-IF
41 *
42 REPORT    LINESIZE 70
43   TITLE     'EMPLOYEES IN REGION 1'
44   LINE      NAME-FIRST  NAME-LAST  EMP#  BRANCH
```

```
------------------------------------------------------------------
11/10/83              EMPLOYEES IN REGION 1            PAGE      1

               FIRST       LAST     EMPLOYEE
                NAME       NAME      NUMBER    BRANCH

              GLORIA     WIMN        12267       01
              NANCY      BERG        11473       02
              GEORGE     CORNING     02688       03
              MARY       NAGLE       00370       02
              LINDA      ARNOLD      01963       04
              VIRGINIA   MANHART     11602       03
              ELAINE     TALL        11931       04
              LYDIA      BRANDOW     02200       01
              RODNEY     LARSON      11357       04
              JULIE      BYER        11467       04
```

Proposed Salary Schedules

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

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

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

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

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

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

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

```
 1 *
 2 *   EXAMPLE 14.2
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 RAISE-PERCENT  W  3 P 2     HEADING('RAISE' '(PERCENT)')
36 RAISE-DOLLARS  W  4 P 2     HEADING('RAISE' '(DOLLARS)')
37 NEW-SALARY     W  4 P 2     HEADING('PROPOSED' 'SALARY')
38 *
39 *
40 JOB
41   IF  REGION NQ 4                      . * REJECT UNDESIRED RECORDS
43     GOTO JOB
44   END-IF
45   IF  JOB-CATEGORY = 10                . * SET RAISE AMT BASED ON
47     RAISE-PERCENT = 7.00               . * JOB-CATEGORY
49   ELSE
50     RAISE-PERCENT = 9.00
51   END-IF
52 *             CALCULATE RAISE IN DOLLARS AND NEW GROSS PAY
53   RAISE-DOLLARS = RAISE-PERCENT * PAY-GROSS / 100 + .005
54   NEW-SALARY = PAY-GROSS + RAISE-DOLLARS
55   PRINT  DETAIL-BY-BRANCH              . * PRINT DESIRED REPORTS
57   PRINT  SUMMARY-BY-CATEGORY
58 *
59 REPORT  DETAIL-BY-BRANCH               LINESIZE 78
60   SEQUENCE  BRANCH PAY-GROSS D
61   CONTROL   BRANCH
```

```
62   TITLE  1  'PROPOSED SALARY SCHEDULE FOR REGION 4 EMPLOYEES'
63   TITLE  2  'DETAIL BY BRANCH -- DESCENDING PAY-GROSS'
64   LINE   1  BRANCH  NAME-LAST  PAY-GROSS  RAISE-DOLLARS RAISE-PERCENT -
               NEW-SALARY
65   BEFORE-BREAK.  PROC
67     RAISE-PERCENT = RAISE-PERCENT / TALLY + .005
68   END-PROC
69 *
70 REPORT  SUMMARY-BY-CATEGORY  SUMMARY   LINESIZE 78
71   SEQUENCE  BRANCH JOB-CATEGORY
72   CONTROL   BRANCH JOB-CATEGORY
73   TITLE  1  'PROPOSED SALARY SCHEDULE FOR REGION 4'
74   TITLE  2  'SUMMARY BY JOB-CATEGORY AND BRANCH'
75   LINE   1  BRANCH  JOB-CATEGORY  PAY-GROSS  NEW-SALARY  RAISE-DOLLARS
```

```
11/10/83        PROPOSED SALARY SCHEDULE FOR REGION 4 EMPLOYEES   PAGE 1
                   DETAIL BY BRANCH -- DESCENDING PAY-GROSS

         LAST         GROSS          RAISE         RAISE    PROPOSED
BRANCH   NAME          PAY          (DOLLARS)     (PERCENT)  SALARY

  01     TALUS          460.80          41.47         9.00     502.27
         RYAN           399.20          27.94         7.00     427.14
         WARD           183.75          12.86         7.00     196.61
  01                  1,043.75          82.27         7.67   1,126.02
  02     JOHNSON        712.80          49.90         7.00     762.70
         HAFER          121.95          10.98         9.00     132.93
  02                    834.75          60.88         8.00     895.63

  03     JONES          804.80          72.43         9.00     877.23
         WALTERS        424.00          29.68         7.00     453.68
         ZOLTAN         125.00          11.25         9.00     136.25
  03                  1,353.80         113.36         8.33   1,467.16

  04     JUDAR          591.20          53.21         9.00     644.41
  04                    591.20          53.21         9.00     644.41

                      3,823.50         309.72         8.11   4,133.22
```

```
11/10/83             PROPOSED SALARY SCHEDULE FOR REGION 4        PAGE 1
                       SUMMARY BY JOB-CATEGORY AND BRANCH

              JOB          GROSS         PROPOSED          RAISE
      BRANCH  CATEGORY      PAY           SALARY          (DOLLARS)

        01       10          582.95          623.75           40.80
        01       25          460.80          502.27           41.47
        01                 1,043.75        1,126.02           82.27

        02       10          712.80          762.70           49.90
        02       60          121.95          132.93           10.98
        02                   834.75          895.63           60.88

        03       10          424.00          453.68           29.68
        03       40          929.80        1,013.48           83.68
        03                 1,353.80        1,467.16          113.36

        04       25          591.20          644.41           53.21
        04                   591.20          644.41           53.21

                           3,823.50        4,133.22          309.72
```

Employee Letters

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

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

Note: The ease with which the letter is specified.

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

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

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

```
 1 *
 2 *   EXAMPLE 14.3
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 OLD-SALARY  PAY-GROSS PAY-GROSS   MASK(S '$$$$$.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'  -
               NEW-SALARY '.'
76   LINE  19  'THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES'
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.                                          11/11/83

KAREN    WALTERS
1022 5 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.                                          11/11/83

ALFRED   JONES
2070 BELMONT ROAD NW
LOS ANGELES  CA 90052

DEAR ALFRED

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

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

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

SINCERELY,


FRANK K. WILLIAMS
PRESIDENT
```

```
ABC SYSTEMS, INC.                                          11/11/83

JANET    ZOLTAN
2026 FORT DAVIS ST S
LOS ANGELES  CA 90091

DEAR JANET

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

IN YOUR PARTICULAR CASE THE INCREASE IS   9.00  %
OF YOUR GROSS SALARY OF   $125.00 .  THIS EQUATES
TO    $11.25 , OR A NEW GROSS SALARY OF   $136.25 .
THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES
YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,


FRANK K. WILLIAMS
PRESIDENT
```

ABC SYSTEMS, INC. 11/11/83

ARTHUR HAFER
806 CONNECTICUT AVE
SAN DIEGO CA 92031

DEAR ARTHUR

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

IN YOUR PARTICULAR CASE THE INCREASE IS 9.00 %
OF YOUR GROSS SALARY OF $121.95 . THIS EQUATES
TO $10.98 , OR A NEW GROSS SALARY OF $132.93 .
THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES
YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

ABC SYSTEMS, INC. 11/11/83

LISA JOHNSON
806 CONNECTICUT AVE
SAN DIEGO CA 92045

DEAR LISA

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

IN YOUR PARTICULAR CASE THE INCREASE IS 7.00 %
OF YOUR GROSS SALARY OF $712.80 . THIS EQUATES
TO $49.90 , OR A NEW GROSS SALARY OF $762.70 .
THE EXECUTIVE BOARD OF ABC SYSTEMS CONGRATULATES
YOU AND LOOKS FORWARD TO AN EVEN BETTER COMING YEAR.

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

```
ABC SYSTEMS, INC.                                          11/11/83

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

DEAR PAULA

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

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

SINCERELY,



FRANK K. WILLIAMS
PRESIDENT
```

```
ABC SYSTEMS, INC.                                          11/11/83

RUTH     TALUS
9331 CAROLINE AVE
SEATTLE      WA 98003

DEAR RUTH

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

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

SINCERELY,



FRANK K. WILLIAMS
PRESIDENT
```

ABC SYSTEMS, INC. 11/11/83

PAMELA RYAN
1717 R NW #301
SEATTLE WA 98009

DEAR PAMELA

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

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

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

ABC SYSTEMS, INC. 11/11/83

MARINA WARD
1725 H ST NE APT 2
SEATTLE WA 98015

DEAR MARINA

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

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

SINCERELY,

FRANK K. WILLIAMS
PRESIDENT

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


JANET      ZOLTAN            ARTHUR     HAFER
2026 FORT DAVIS ST S         806 CONNECTICUT AVE
LOS ANGELES    CA   90091    SAN DIEGO      CA   92031


LISA       JOHNSON           PAULA      JUDAR
806 CONNECTICUT AVE          4333 46TH ST N W
SAN DIEGO      CA   92045    SAN FRANCISC   CA   94041


RUTH       TALUS             PAMELA     RYAN
9331 CAROLINE AVENUE         1717 R N W #301
SEATTLE        WA   98003    SEATTLE        WA   98009


MARINA     WARD
1725 H ST NE APT 2
SEATTLE        WA   98015
```

Mailing Labels

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

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

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

```
 1 *
 2 *   EXAMPLE 14.4
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 ZIP-PREFIX  ADDR-ZIP  3  N          . * REDEFINE FIRST 3 DIGITS OF ZIP
37 *
38 *
39 JOB
40   IF  REGION EQ 1 2                 . * SELECT DESIRED RECORDS
42     PRINT  MAILING-LABEL            . * PRINT MAILING LABEL
44   END-IF
45 *
46 REPORT  MAILING-LABEL    LABELS (ACROSS 3  SIZE 28)  SPACE 1
47   SEQUENCE  ADDR-ZIP                . * SORT ON ZIP CODE
49   CONTROL   ZIP-PREFIX              . * BREAK ON ZIP PREFIX
51   LINE   1  EMP# REGION BRANCH
52   LINE   3  NAME-FIRST NAME-LAST
53   LINE   4  ADDR-STREET
54   LINE   5  ADDR-CITY  ADDR-STATE   ADDR-ZIP
```

```
12267 1 01                  02200 1 01

GLORIA   WIMN               LYDIA    BRANDOW
430 M ST SW 107             3616 B ST S E
BOSTON       MA 02005       BOSTON       MA 02011

11602 1 03                  02688 1 03

VIRGINIA MANHART            GEORGE   CORNING
1305 POTOMAC ST N W         3208 S 5TH
TRENTON      NJ 08521       TRENTON      NJ 08535

01963 1 04                  11357 1 04                  11467 1 04

LINDA    ARNOLD             RODNEY   LARSON             JULIE    BYER
1569 COLONIAL TERR A        610 H ST SW                 3400 NORTH 18TH STRE
NEW YORK     NY 10012       NEW YORK     NY 10059       NEW YORK     NY 10071
```

```
11931 1 04

ELAINE   TALL
1412 36TH ST NW
NEW YORK     NY 10091

02765 2 03                  04234 2 03                  03416 2 03

RALPH    DENNING            BARBARA  MCMAHON           BILL     FORREST
1629 16TH ST NW APT         1318 24TH STREET S         1545 18TH ST NW
WASHINGTON   DC 20005       WASHINGTON   DC 20015      WASHINGTON   DC 20018

00577 2 03

KATHY    PETRIK
5005 BENTON AVE
WASHINGTON   DC 20032

00370 1 02                  11473 1 02

MARY     NAGLE              NANCY    BERG
826 D STREET SE             3710 JENIFER ST N W
BALTIMORE    MD 21034       BALTIMORE    MD 21055

04225 2 05                  01895 2 05

NED      LOYAL              DENISE   VETTER
17 KENNEDY STREET           7311 KEYSTONE LANE 4
RALEIGH      NC 27516       RALEIGH      NC 27591

11710 2 02                  03571 2 02

CAROL    POWELL             MAX      KRUSE
5023 AMES STREET N E        2161 N PIERCE STREET
ATLANTA      GA 30316       ATLANTA      GA 30345

00445 2 04

JEAN     POST
1250 4TH ST SW
JACKSONVILLE FL 32052

11376 2 01

PATTI    HUSS
1355 TEWKESBURY PLAC
CLEARWATER   FL 33512
```

Tally Reports

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

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

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

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

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

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

```
 1 *
 2 *   EXAMPLE 14.5
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35   SEX-CODE         W  6 A     HEADING 'SEX'
36   GROSS-RANGE      W  3 P     HEADING ('SALARY RANGE' 'HUNDRED $ INCR')
37   TOTAL-EMPLOYEES  S  3 P 0
38   PERCENT          W  3 P 2   HEADING('PERCENT' 'OF' 'TOTAL')
39 *
40 *
41 JOB
42   TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1
43 *
44   IF  SEX EQ 1                       . * SET PROPER SEX CODE
46     SEX-CODE = 'FEMALE'
47   ELSE
48     SEX-CODE = 'MALE'
49   END-IF
50   PRINT  SEX-MARITAL-STAT-RPT        . * PRINT REPORT
52 *
53   PRINT JOB-CATEGORY-RPT             . * PRINT REPORT
55 *
```

```
 56   PRINT SALARY-CODE-RPT             . * PRINT REPORT
 58 *
 59   GROSS-RANGE = PAY-GROSS / 100.00  . * CALCULATE GROSS SALARY
 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
 89   BEFORE-BREAK. PROC                . * CALCULATE 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
102   BEFORE-BREAK. PROC                . * CALCULATE 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
116   BEFORE-BREAK. PROC                . * CALCULATE PERCENT
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'
129   LINE    1 ADDR-CITY  TALLY  PERCENT
130   BEFORE-BREAK. PROC                . * CALCULATE PERCENT
133     PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
134   END-PROC
```

3/04/84 TALLY OF EMPLOYEES BY SEX AND MARITAL STATUS PAGE 1

SEX	MARITAL STATUS	TALLY	PERCENT OF TOTAL
FEMALE	M	13	27.08
FEMALE	S	23	47.92
FEMALE		36	75.00
MALE	M	5	10.42
MALE	S	7	14.58
MALE		12	25.00
		48	100.00

3/04/84 TALLY OF EMPLOYEES BY JOB CATEGORY PAGE 1

JOB CATEGORY	TALLY	PERCENT OF TOTAL
10	22	45.83
20	2	4.17
23	1	2.08
25	8	16.67
30	2	4.17
40	6	12.50
60	7	14.58
	48	100.00

3/04/84 TALLY OF EMPLOYEES BY SALARY CODE PAGE 1

SALARY CODE	TALLY	PERCENT OF TOTAL
01	20	41.67
02	2	4.17
03	9	18.75
05	2	4.17
06	7	14.58
08	8	16.67
	48	100.00

3/04/84 TALLY OF EMPLOYEES BY GROSS SALARY RANGE PAGE 1

SALARY RANGE HUNDRED $ INCR	TALLY	PERCENT OF TOTAL	AVERAGE GROSS SALARY
1000	1	2.08	1,004.00
800	2	4.17	804.72
700	3	6.25	736.00
600	1	2.08	628.00
500	2	4.17	572.80
400	4	8.33	455.64
300	17	35.42	348.19
200	10	20.83	264.29
100	7	14.58	154.92
	1	2.08	13.80
	48	100.00	376.63

3/04/84 TALLY OF EMPLOYEES BY HOME CITY PAGE 1

CITY	TALLY	PERCENT OF TOTAL
ATLANTA	2	4.17
BALTIMORE	2	4.17
BOSTON	2	4.17
CHICAGO	6	12.50
CLEARWATER	1	2.08
DALLAS	6	12.50
JACKSONVILLE	1	2.08
KANSAS CITY	4	8.33
LOS ANGELES	3	6.25
MINNEAPOLIS	3	6.25
NEW YORK	4	8.33
RALEIGH	2	4.17
SAN DIEGO	2	4.17
SAN FRANCISC	1	2.08
SEATTLE	3	6.25
TRENTON	2	4.17
WASHINGTON	4	8.33
	48	100.00

Women's Phone Numbers

The National Federation of Business and Professional Women's Clubs is recruiting for a chapter in the Chicago area. They have requested a list of all female employees in the Chicago branch, along with their phone numbers.

This example is a simple process of selecting records based on the value in two fields, ADDR and SEX, then sequencing the report by name.

```
 1 *
 2 *   EXAMPLE 14.6
 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


------------------------------------------------------------------------------
11/11/83           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
```

Salary Tally Report

The Personnel Department has requested that the Salary Range Report, produced in Example 14.5, be expanded to include a bar graph of tally percent. The bar graph is generated using the MOVE statement within the BEFORE-BREAK procedure. For each two percentage points, an asterisk is plotted. If the percentage exceeds 60 percent, spaces are printed.

As illustrated in this and several of the previous examples, the BEFORE-BREAK procedure is invaluable. It permits us to modify the contents of a summary line prior to printing (a common requirement in many control reports).

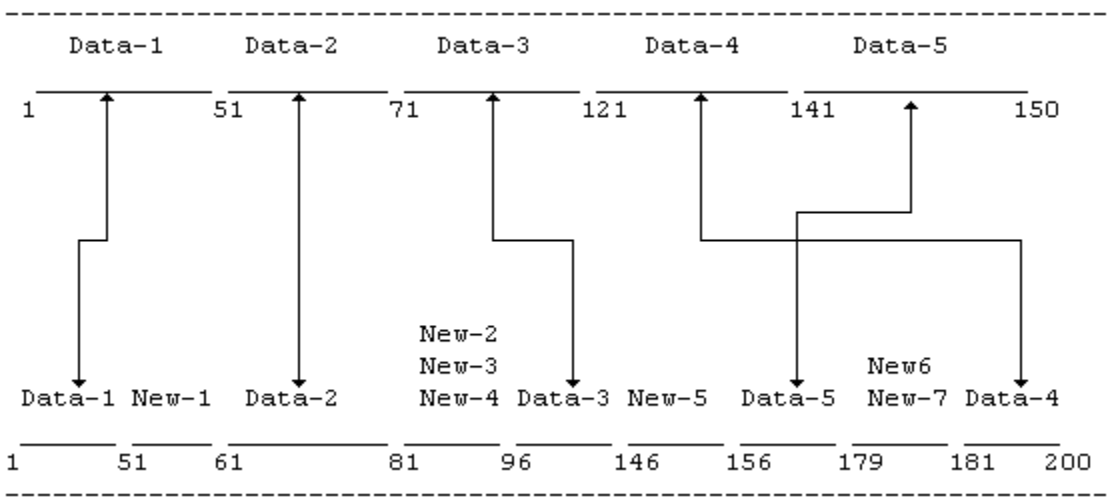
```
 1 *
 2 *    EXAMPLE 14.7
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35   GROSS-RANGE      W  3 P     HEADING ('SALARY RANGE' 'HUNDRED $ INCR')
36   TOTAL-EMPLOYEES  S  3 P 0
37   PERCENT          W  3 P 2   HEADING('PERCENT' 'OF' 'TOTAL')
38   BAR-GRAPH        S 30 A     HEADING('PERCENT OF EMPLOYEES' -
                                          'EACH ASTERISK EQUALS 2%')
39   ASTERISKS        S 30 A     VALUE('******************************')
40   ILTH             S  2 P
41 *
42 *
43 JOB
44   TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1
45 *
46   GROSS-RANGE = PAY-GROSS / 100.00  . * CALCULATE GROSS SALARY
48   GROSS-RANGE = GROSS-RANGE * 100   . *   RANGE
50   PRINT GROSS-PAY-RPT               . * PRINT THE REPORT
52 *
53 REPORT  GROSS-PAY-RPT  SUMMARY  SUMCTL DTLCOPY  LINESIZE 78
54   SEQUENCE  GROSS-RANGE  D          . * SORT REPORT
56   CONTROL   GROSS-RANGE             . * BREAK SPECIFICATION
58   TITLE   1 'TALLY OF EMPLOYEES BY GROSS SALARY RANGE'
59   LINE    1 GROSS-RANGE  TALLY  PERCENT  BAR-GRAPH
60   BEFORE-BREAK. PROC                . * CALCULATE PERCENT
63     PERCENT = TALLY * 100 / TOTAL-EMPLOYEES + .005
64     ILTH = ( PERCENT + 1 ) / 2
65     IF ILTH LE 30
66       MOVE ASTERISKS ILTH  TO  BAR-GRAPH
67     ELSE
68       MOVE SPACES  TO  BAR-GRAPH
69     END-IF
70   END-PROC
```

```
--------------------------------------------------------------------------------
12/01/83          TALLY OF EMPLOYEES BY GROSS SALARY RANGE       PAGE      1


                                    PERCENT
    SALARY RANGE                      OF          PERCENT OF EMPLOYEES
  HUNDRED $ INCR    TALLY          TOTAL         EACH ASTERISK EQUALS 2%

        1000            1           2.08    *
         800            2           4.17    **
         700            3           6.25    ***
         600            1           2.08    *
         500            2           4.17    **
         400            4           8.33    ****
         300           17          35.42    ******************
         200           10          20.83    **********
         100            7          14.58    *******
                        1           2.08    *
                       48         100.00
```

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, one MOVE LIKE statement performs all five data moves. The MOVE statement initializes the new fields.

```
 1 *
 2 *     EXAMPLE 14.8
 3 *
 4 FILE  PERSIN                        . * INPUT FILE
 6   DATA-1        1  50 A
 7   DATA-2       51  20 A
 8   DATA-3       71  50 A
 9   DATA-4      121  20 A
10   DATA-5      141  10 A
11 *
12 FILE  PERSOUT FB(200 3600)         . * REFORMATTED OUTPUT FILE
14   DATA-1        1  50 A
15   NEW-1        51  10 N 0
16   DATA-2       61  20 A
17   NEW-2        81   4 P
18   NEW-3        85   5 P
19   NEW-4        90   6 N 0
20   DATA-3       96  50 A
21   NEW-5       146  10 A
22   DATA-5      156  10 A
23   NEW-6       166  13 A
24   NEW-7       179   2 B
25   DATA-4      181  20 A
26 *
27 *
28 JOB   FINISH  WRAP-UP
29   MOVE  LIKE  PERSIN TO PERSOUT    . * MOVE LIKE NAMED FIELDS
31                                      *   FROM PERSIN TO PERSOUT
32   MOVE  ZERO  TO  NEW-1 NEW-2 NEW-3 NEW-4 NEW-7
33                                      * INITIALIZE NUMERIC FIELDS
34   MOVE  SPACE TO  NEW-5 NEW-6      . * INITIALIZE ALPHA FIELDS
36   PUT  PERSOUT                     . * OUTPUT THE REFORMATTED FILE
38 *
39   WRAP-UP.  PROC
41     DISPLAY NEWPAGE 'TOTAL INPUT  RECORDS = ' RECORD-COUNT(PERSIN)
42     DISPLAY SKIP 2  'TOTAL OUTPUT RECORDS = ' RECORD-COUNT(PERSOUT)
43   END-PROC

------------------------------------------------------------------------

TOTAL INPUT  RECORDS =            48


TOTAL OUTPUT RECORDS =            48

------------------------------------------------------------------------
```

Average Regional Gross Salary

The region codes of the personnel file represent regions of the United States. In most cases it is more desirable to output a text description of the region than to print the code. The conversion is performed by the CA-Easytrieve Plus table handling facility.

In this example, the Personnel Department has requested a report of average gross salaries for each region. The input records are read and totals calculated for the number of employees and the gross salaries. The SEARCH statement obtains the text description of the region code, and the information is output on a report.

Note: The SEQUENCE statement specifies the region code while the CONTROL break is based on REGION-TEXT. This enables the report to be ordered on region code while still printing the region text.

Also, most of the printed values are generated in the BEFORE-BREAK procedure. The order of the first two statements in that procedure is mandatory because the second statement modifies the AVERAGE-GROSS.

```
 1 *
 2 *   EXAMPLE 14.9
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35   AVERAGE-GROSS    W  4 P 2   HEADING ('AVERAGE' 'GROSS' 'SALARY')
36   TOTAL-GROSS      S  6 P 2
37   PERCENT-GROSS    W  3 P 2   HEADING ('PERCENT OF' 'COMPANY' 'GROSS')
38   PERCENT-TALLY    W  3 P 2   -
                                 HEADING('PERCENT OF' 'COMPANY' 'EMPLOYEES')
39   SALARY-RATIO     W  3 P 3   HEADING('RATIO OF' '%-GROSS /' '%-TALLY')
40   TOTAL-EMPLOYEES  S  3 P 0
41   REGION-TEXT      W 10 A     HEADING('COMPANY' 'REGION')
42 *
43 FILE  RGNID  TABLE  INSTREAM  . * DEFINE INSTREAM REGION TABLE
45   ARG 1 1 N.   DESC 3 10 A.     * DEFINE TABLE SPECIAL FIELD IDS
48 1 NORTHEAST
   2 SOUTHEAST
   3 CENTRAL
   4 WEST
   ENDTABLE
49 *
50 *
51 JOB
52   TOTAL-EMPLOYEES = TOTAL-EMPLOYEES + 1 . * CALCULATE TOTAL EMPLOYEES
54   AVERAGE-GROSS = PAY-GROSS          . * AVERAGE = GROSS FOR EACH RECD
56   TOTAL-GROSS = TOTAL-GROSS + PAY-GROSS
57                                        * CALCULATE TOTAL GROSS FOR COMP
58 *                SEARCH TABLE FOR MATCHING REGION INFORMATION
59   SEARCH  RGNID  WITH REGION  GIVING REGION-TEXT
60 *
61   PRINT  AVG-SALARY-RPT              . * PRINT THE REPORT
63 *
64 REPORT  AVG-SALARY-RPT  SUMMARY         LINESIZE 78
65   SEQUENCE  REGION                   . * SORT REPORT
67   CONTROL   REGION-TEXT              . * BREAK SPECIFICATION
69   TITLE   1 'AVERAGE GROSS SALARY BY REGION'
70   HEADING   TALLY   ('NUMBER' 'OF' 'EMPLOYEES')
```

```
71   LINE    1 REGION-TEXT  TALLY  PERCENT-TALLY -
               AVERAGE-GROSS  PERCENT-GROSS  SALARY-RATIO
72   BEFORE-BREAK. PROC                . * CALCULATE PERCENT
75     PERCENT-GROSS = AVERAGE-GROSS * 100 / TOTAL-GROSS + .005
76     AVERAGE-GROSS = AVERAGE-GROSS / TALLY + .005
77     PERCENT-TALLY = TALLY * 100 / TOTAL-EMPLOYEES + .005
78     SALARY-RATIO  = PERCENT-GROSS / PERCENT-TALLY + .0005
79   END-PROC


------------------------------------------------------------------------------
11/12/83               AVERAGE GROSS SALARY BY REGION            PAGE      1


              NUMBER      PERCENT OF      AVERAGE      PERCENT OF    RATIO OF
 COMPANY        OF         COMPANY         GROSS        COMPANY     %-GROSS /
  REGION     EMPLOYEES    EMPLOYEES       SALARY         GROSS       %-TALLY

NORTHEAST        10          20.83          460.12        25.45        1.222
SOUTHEAST        10          20.83          246.98        13.66         .656
CENTRAL          19          39.58          378.08        39.74        1.004
WEST              9          18.75          424.83        21.15        1.128
                 48         100.00          376.63       100.00        1.000
```

Central Region Employees

The Personnel Department has requested an alphabetical list of employees in the central region. The report is to include the employees' name, social security number, department code, and department name. In addition, Personnel needs a list of the central region employees grouped by department name.

To solve this problem, we must know that each employee is assigned to a particular company department, the number of which is contained within each employee record. In addition to the number, each department has a unique department name, such as Engineering, Marketing, and so forth. A table of department numbers and the corresponding names is available in a table file named DPTCODE.

First, we select all employees in Region 3 (Central Region). For each such employee, we search the DPTCODE table for the corresponding department name. If no entry is found, we insert a dummy department name (*NO TABLE ENTRY) and issue a PRINT to an error report. Regardless whether a department name is found, we issue a PRINT statement to both the ALPHA-LIST and the RPT-BY-DEPT reports.

ALPHA-LIST is a simple list, sequenced by name.

The RPT-BY-DEPT is a control report with breaks on DEPT.

Note: The use of the HEADING statement supplies alternate report headings for the specified fields; this is the only way to change the heading for TALLY.

Also, printing is suppressed for the summary line in the MISSING-DEPT-CODE report.

```
 1 *
 2 *   EXAMPLE 14.10
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 *
36   DEPT-NAME   W 15  A    HEADING ('DEPARTMENT' 'NAME')
37 *
38 FILE    DPTCODE   TABLE               . * TABLE FILE DEFINITION
40   ARG  1 3 N.   DESC  5 15 A
42 *
43 *
44 JOB
45   IF  REGION NE 3                     . * SELECT REGION 3 EMPLOYEES
47     GO TO JOB                         . * SKIP ALL OTHERS
49   END-IF
50   SEARCH DPTCODE  WITH DEPT  GIVING DEPT-NAME
51                                        * GET DEPT NAME FROM TABLE
52   IF  NOT DPTCODE                     . * IF NO DEPT NAME PRESENT
54     DEPT-NAME = '*NO TABLE ENTRY'     . * INDICATE MISSING ENTRY
56     PRINT  MISSING-DEPT-CODE          . * OUTPUT ERROR REPORT
58   END-IF
59   PRINT  ALPHA-LIST                   . * PRINT ALPHA LISTING
61   PRINT  RPT-BY-DEPT                  . * OUTPUT REPORT BY DEPARTMENT
63 *
64 REPORT   ALPHA-LIST                 LINESIZE 78
65   SEQUENCE  NAME-LAST NAME-FIRST
66   TITLE     'CENTRAL REGION EMPLOYEES'
67   LINE      NAME-LAST NAME-FIRST   SSN  DEPT  DEPT-NAME
68 *
69 REPORT   RPT-BY-DEPT   SUMCTL NONE   LINESIZE 78
70   SEQUENCE  DEPT-NAME NAME-LAST       . * SEQUENCE BY DEPT AND NAME
72   CONTROL   DEPT-NAME                 . * CONTROL BREAK ON DEPT
74   TITLE     'CENTRAL REGION EMPLOYEES BY DEPARTMENT'
75   HEADING   TALLY    ('NUMBER' 'OF' 'EMPLOYEES')
76   LINE      DEPT-NAME  BRANCH  NAME-LAST NAME-FIRST  TALLY
77 *
78 REPORT   MISSING-DEPT-CODE  SUMMARY  LINESIZE 78
79   SEQUENCE  DEPT                      . * SEQUENCE BY DEPT
81   CONTROL   FINAL NOPRINT DEPT        . * CONTROL BREAK ON DEPT
83   TITLE     'CENTRAL REGION MISSING DEPARTMENT DESCRIPTIONS'
84   HEADING   DEPT     ('MISSING' 'DEPARTMENT' 'CODES')
85   HEADING   TALLY    ('NUMBER' 'OF' 'EMPLOYEES')
86   LINE      DEPT  TALLY
```

11/12/83 CENTRAL REGION EMPLOYEES PAGE 1

LAST NAME	FIRST NAME	SOCIAL SECURITY NUMBER	DEPT	DEPARTMENT NAME
CROCI	JUDY	577-16-2985	914	ENGINEERING
EPERT	LINDA	228-46-5157	918	DATA PROCESSING
GRECO	LESLIE	388-18-6119	914	ENGINEERING
GREEN	BRENDA	208-28-2315	911	MARKETING
ISAAC	RUTH	418-46-1872	911	MARKETING
KELLY	KEITH	231-68-9995	940	PRINTING
LACH	LORRIE	215-36-5852	923	MAILROOM
MALLOW	TERRY	477-44-4948	942	*NO TABLE ENTRY
MILLER	JOAN	140-32-0779	920	RECEIVING
NORIDGE	DEBBIE	578-54-3178	944	*NO TABLE ENTRY
OSMON	SAMUEL	269-24-7428	935	RECEIVING
PHILPS	SUE	090-22-9192	940	PRINTING
REYNOLDS	WILLIAM	061-30-8680	911	MARKETING
ROGERS	PAT	579-50-4170	924	PERSONNEL
SMOTH	CINDY	578-38-7587	911	MARKETING
STRIDE	ANN	467-56-4149	911	MARKETING
THOMPSON	JANICE	577-09-1160	923	MAILROOM
WEST	KATHY	570-10-5594	940	PRINTING
YOUNG	JANE	118-34-8805	911	MARKETING

11/12/83 CENTRAL REGION EMPLOYEES BY DEPARTMENT PAGE 1

DEPARTMENT NAME	BRANCH	LAST NAME	FIRST NAME	NUMBER OF EMPLOYEES
*NO TABLE ENTRY	04	MALLOW	TERRY	
	03	NORIDGE	DEBBIE	
				2
DATA PROCESSING	03	EPERT	LINDA	
				1
ENGINEERING	04	CROCI	JUDY	
	02	GRECO	LESLIE	
				2
MAILROOM	02	LACH	LORRIE	
	02	THOMPSON	JANICE	
				2
MARKETING	04	GREEN	BRENDA	
	02	ISAAC	RUTH	
	02	REYNOLDS	WILLIAM	
	02	SMOTH	CINDY	
	03	STRIDE	ANN	
	01	YOUNG	JANE	
				6
PERSONNEL	03	ROGERS	PAT	
				1
PRINTING	01	KELLY	KEITH	
	01	PHILPS	SUE	
	01	WEST	KATHY	
				3
RECEIVING	03	MILLER	JOAN	
	03	OSMON	SAMUEL	
				2
				19

11/12/83 CENTRAL REGION MISSING DEPARTMENT DESCRIPTIONS PAGE 1

MISSING DEPARTMENT CODES	NUMBER OF EMPLOYEES
942	1
944	1

Inventory Report by City

An Inventory Master File is available for our use. This file contains information on a diverse inventory. The Material Procurement Department has requested an inventory report, ordered by the city in which the parts are located. Also, the groups by city need to be separated by a blank line, but no totals by city are desired.

The job to perform this request is quite simple; all processing is performed in the report section. The NOPRINT option on the CONTROL statement is used to suppress printing the summary lines.

```
 1 *
 2 *   EXAMPLE 14.11
 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
```

```
11/18/83          INVENTORY BY CITY ORDERED BY PART NUMBER       PAGE  1

                    PART
 CITY             NUMBER          PART DESCRIPTION

CHICAGO          000-15-428     BOOKS, SCHOOL COPY
                 000-16-490     BAGS, GOLF CLUB

E MOLIN          000-10-944     PANEL, SOLAR
                 000-53-100     REFRIGERATORS, HOUSEHOLD
                 000-79-740     BEDS, WOODEN
                 000-81-190     DESKS, STEEL
                 000-82-150     TABLES, PICNIC

HAMMOND          000-70-750     CARPETS, FABRIC (20' X 40')

INDIANP          000-15-980     FAUCETS, BATH TUB
                 000-51-260     PIPE, IRON OR STEEL (3" X 96")
                 000-60-680     BATTERIES, ELECTRIC DRY CELL
                 001-78-200     AIR BRAKES
                 001-79-000     AXLE SHAFTS
                 001-83-800     BRAKE DRUMS
                 001-84-900     CYLINDER SLEEVES
                 001-85-400     DRIVE SHAFTS

KANS CT          000-17-037     SIDING, ALUMINUM (24" X 72")

MAMMOND          000-19-360     WALLBOARD, FIBERBOARD (48" X 96")

MEMPHIS          001-84-200     BUMPERS
                 001-85-200     DOORS
                 001-86-600     FENDERS
                 001-88-800     HUBS

MUSKEGN          000-11-576     MACHINES, CALCULATING
                 000-12-268     DRYERS, HAIR
                 000-62-270     HUMIDIFIERS, PORTABLE

ST PAUL          000-12-440     MOWERS, LAWN
                 000-13-325     SAWS, CHAIN
```

Expanded Inventory Report

After reviewing the previous report, the Materials Department decided they would like an expanded report that 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 that 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.

Both reports are produced 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 14.12
 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
56   PRINT  SMY-BY-CITY
57 *
58 REPORT   INV-BY-CITY  SPREAD      LINESIZE 80
59   SEQUENCE  LOCATION-CITY PART-NUMBER
60   CONTROL   LOCATION-CITY
61   TITLE   1 'INVENTORY BY CITY ORDERED BY PART NUMBER'
62   LINE    1 LOCATION-CITY  PART-NUMBER  -
               ITEM-LAST-INVENTORY-QUANTITY  ITEM-SELLING-PRICE   -
               ITEM-EXT-VALUE
63 *
64 REPORT   SMY-BY-CITY   SUMMARY    LINESIZE 80
65   SEQUENCE  LOCATION-CITY
66   CONTROL   LOCATION-CITY
67   TITLE   1 'INVENTORY VALUE SUMMARY BY CITY'
68   LINE    1 LOCATION-CITY  ITEM-EXT-VALUE  PERCENT
69   BEFORE-BREAK. PROC
71     PERCENT = ITEM-EXT-VALUE * 100 / TOTAL-EXT-VALUE + .005
72   END-PROC
```

```
11/18/83          INVENTORY BY CITY ORDERED BY PART NUMBER        PAGE 1


                             LAST          SELLING
             PART         INVENTORY        PRICE           EXTENDED
  CITY       NUMBER        QUANTITY       (DOLLARS)         VALUE

 CHICAGO    000-15-428         41,353          12.95        535,521.35
            000-16-490            238          49.95         11,888.10
 CHICAGO                       41,591          62.90        547,409.45

 E MOLIN    000-10-944            854          54.99         46,961.46
            000-53-100            181         879.95        159,270.95
            000-79-740             81         870.00         70,470.00
            000-81-190             35         389.95         13,648.25
            000-82-150            134         199.89         26,785.26
 E MOLIN                        1,285       2,394.78        317,135.92
```

HAMMOND	000-70-750	358	425.00	152,150.00
HAMMOND		358	425.00	152,150.00
INDIANP	000-15-980	3,150	14.29	45,013.50
	000-51-260	14,389	15.25	219,432.25
	000-60-680	654	54.90	35,904.60
	001-78-200	385	59.88	23,053.80
	001-79-000	385	59.88	23,053.80
	001-83-800	439	43.59	19,136.01
	001-84-900	86	31.59	2,716.74
	001-85-400	109	81.45	8,878.05
INDIANP		19,597	360.83	377,188.75
KANS CT	000-17-037	2,218	8.99	19,939.82
KANS CT		2,218	8.99	19,939.82
MAMMOND	000-19-360	2,810	18.95	53,249.50
MAMMOND		2,810	18.95	53,249.50

11/18/83 INVENTORY BY CITY ORDERED BY PART NUMBER PAGE 2

CITY	PART NUMBER	LAST INVENTORY QUANTITY	SELLING PRICE (DOLLARS)	EXTENDED VALUE
MEMPHIS	001-84-200	653	99.88	65,221.64
	001-85-200	2,210	195.50	432,055.00
	001-86-600	3,403	159.88	544,071.64
	001-88-800	3,952	55.95	221,114.40
MEMPHIS		10,218	511.21	1,262,462.68
MUSKEGN	000-11-576	88	119.66	10,530.08
	000-12-268	805	38.88	31,298.40
	000-62-270	245	98.97	24,247.65
MUSKEGN		1,138	257.51	66,076.13
ST PAUL	000-12-440	819	243.69	199,582.11
	000-13-325	799	159.66	127,568.34
ST PAUL		1,618	403.35	327,150.45
		80,833	4,443.52	3,122,762.70

11/18/83 INVENTORY VALUE SUMMARY BY CITY PAGE 1

CITY	EXTENDED VALUE	PERCENT OF TOTAL VALUE
CHICAGO	547,409.45	17.53
E MOLIN	317,135.92	10.16
HAMMOND	152,150.00	4.87
INDIANP	377,188.75	12.08
KANS CT	19,939.82	.64
MAMMOND	53,249.50	1.71
MEMPHIS	1,262,462.68	40.43
MUSKEGN	66,076.13	2.12
ST PAUL	327,150.45	10.48
	3,122,762.70	100.00

Error Correction

After reviewing the Inventory by City report in Example 14.12, shown in the sample report, an error has been detected in the Inventory Master File. The location for part number 000-19-360 is currently MAMMOND instead of the correct city HAMMOND. A CA-Easytrieve Plus job can correct it easily.

The required job reads the existing file, finds the record in error, makes the correction, generates an audit trail to reflect the change, and outputs an updated master file. All of the records in the updated file are identical to the current file, except the record for part number 000-19-360.

```
 1 *
 2 *    EXAMPLE 14.13
 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


11/23/83        INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL      PAGE   1

                      PART
                     NUMBER      CITY     UPDATE-STATUS

                   000-19-360   MAMMOND      BEFORE
                   000-19-360   HAMMOND      AFTER
```

Inventory Reduction

A new accountant for the company wants to reduce the inventory of truck parts (commodity group 19720) by 15 percent. She thinks that this would save a substantial amount of money (since the interest rate is so high) and, therefore, has requested a report that indicates how much could be saved.

The Inventory Reduction report is produced by the following steps:

1. Select all items in commodity group 19720.
2. Determine the maximum quantity of inventory reduction that 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 that provides this information, ordered by decreasing savings.

```
 1 *
 2 *   EXAMPLE 14.14
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5 %INVMSTR
44 *
45   MIN-STOCK-LEVEL        W   4 P 0
46   STOCK-REDUCTION-QUANT  W   4 P 0
47   PROPOSED-STOCK-QUANT   W   4 P 0   -
                            HEADING('PROPOSED' 'STOCK' 'QUANTITY')
48   STOCK-VALUE-SAVINGS    W   5 P 2  HEADING('STOCK VALUE' 'SAVINGS')
49   STOCK-INT-SAVINGS      W   5 P 2  HEADING('STOCK INTEREST' 'SAVINGS')
50 *
51 JOB
52   IF ITEM-MFGD-COMMODITY-GROUP  NE  19720   . * REJECT UNWANTED RECDS
54     GOTO JOB
55   END-IF
56   MIN-STOCK-LEVEL = 1.2 * ITEM-REORDER-POINT + .5
57   IF  ITEM-LAST-INVENTORY-QUANTITY  LE  MIN-STOCK-LEVEL
58     STOCK-REDUCTION-QUANT = 0        . * NO REDUCTION IF ALREADY AT MIN
60     PROPOSED-STOCK-QUANT = ITEM-LAST-INVENTORY-QUANTITY
61     PERFORM  REDUCTION-REPORT
62     GOTO JOB
63   END-IF
64 *
65   STOCK-REDUCTION-QUANT = .15 * ITEM-LAST-INVENTORY-QUANTITY
66   PROPOSED-STOCK-QUANT =   -
         ITEM-LAST-INVENTORY-QUANTITY - STOCK-REDUCTION-QUANT
67   IF PROPOSED-STOCK-QUANT LT MIN-STOCK-LEVEL
68     PROPOSED-STOCK-QUANT = MIN-STOCK-LEVEL
69     STOCK-REDUCTION-QUANT = ITEM-LAST-INVENTORY-QUANTITY  -
                             - PROPOSED-STOCK-QUANT
70   END-IF
71   PERFORM  REDUCTION-REPORT
72 *
73 REDUCTION-REPORT. PROC
75   STOCK-VALUE-SAVINGS = STOCK-REDUCTION-QUANT * LAST-PURCHASE-PRICE
76   STOCK-INT-SAVINGS = .015 * STOCK-VALUE-SAVINGS
77   PRINT  SAVINGS-REPORT
78 END-PROC
79 *
80 REPORT  SAVINGS-REPORT       SKIP 1   LINESIZE 80
81   SEQUENCE  STOCK-VALUE-SAVINGS  D
82   CONTROL
83   TITLE   1 'STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720'
84   LINE    1 PART-NUMBER  ITEM-LAST-INVENTORY-QUANTITY -
               PROPOSED-STOCK-QUANT  -
               STOCK-VALUE-SAVINGS  STOCK-INT-SAVINGS
85   LINE    2 PART-DESCRIPTION
```

```
11/23/83       STOCK REDUCTION ANALYSIS FOR COMMODITY GROUP 19720   PAGE      1

                  LAST          PROPOSED
   PART        INVENTORY         STOCK          STOCK VALUE        STOCK INTEREST
  NUMBER       QUANTITY        QUANTITY           SAVINGS             SAVINGS

001-86-600          3,403           2,893          40,774.50              611.61
FENDERS

001-85-200          2,210           1,879          33,060.28              495.90
DOORS

001-88-800          3,952           3,360          16,155.68              242.33
HUBS

001-84-200            653             556           4,413.50               66.20
BUMPERS

001-78-200            385             328           1,707.15               25.60
AIR BRAKES

001-79-000            385             328           1,707.15               25.60
AXLE SHAFTS

001-83-800            439             374           1,462.50               21.93
BRAKE DRUMS

001-85-400            109              93             640.00                9.60
DRIVE SHAFTS

001-84-900             86              86                .00                 .00
CYLINDER SLEEVES

                   11,622           9,897          99,920.76           1,498.77
```

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 job to update the appropriate records and produce an audit trail of the changes.

There are a variety of ways to update files. One method is the technique used in "Error Correction, Example 14.13," 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 the "Advanced Techniques" chapter.

Table Files

An excellent technique to update a moderate number of records is to use a table file. In this example, an instream table is defined. The argument equals the part number and the description contains the new quantity and date of inventory.

As data is read from the master file, a check is made against the table for a match.

- If no match is found, the record is written unmodified.
- If a match occurs, the quantity and inventory date are changed, the updated record is written, and an audit report is generated.

In addition, if the inventory for a particular item has been depleted by more than 20 percent of its original value, a management report is generated.

```
 1 *
 2 *   EXAMPLE 14.15
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5 %INVMSTR
44 *
45   TABLE-DESC                 W 20 A
46     NEW-DATE   TABLE-DESC       6 N 0   MASK 'Z9/99/99'   -
                                  HEADING('NEW' 'INVENTORY' 'DATE')
47     NEW-QUANT  TABLE-DESC +7    5 N 0  -
                                  HEADING('NEW' 'INVENTORY' 'QUANTITY')
48     PERCENT-DROP   W 3 P 2   HEADING('PERCENT' 'DROP IN' 'INVENTORY')
49 *
50 FILE    NEWMSTR   FB(200 3000)
51 *
52 FILE    UPDTBL    TABLE  INSTREAM
53   ARG 1 8 N.    DESC 10 20 A
55 00178200 103181 00312
   00179000 101581 00434
   00183800 110581 00311
   00184200 111581 00472
   00184900 102281 00081
   00185200 092781 02103
   00185400 111081 00073
   00186600 111981 03401
   00188800 110681 04027
   ENDTABLE
56 *
57 JOB
58   SEARCH  UPDTBL  WITH PART-NUMBER  GIVING TABLE-DESC
59 *
60   IF  UPDTBL                    . * IF MATCH FOUND
62     PRINT  AUDIT-TRAIL          . * OUTPUT AUDIT TRAIL
64     PERFORM EXCESS-CHECK        . * CHECK FOR LARGE QUANT VARIATION
66     ITEM-LAST-INVENTORY-DATE = NEW-DATE  . * UPDATE DATE AND
68     ITEM-LAST-INVENTORY-QUANTITY = NEW-QUANT  . * QUANTITY
70   END-IF
71 *
72   PUT  NEWMSTR  FROM INVMSTR       . * OUTPUT UPDATED FILE
74 *
75 EXCESS-CHECK.  PROC
77   IF NEW-QUANT < .8 * ITEM-LAST-INVENTORY-QUANTITY
78     PERCENT-DROP = 100 -
                    - (NEW-QUANT * 100 / ITEM-LAST-INVENTORY-QUANTITY)
79     PRINT  MGMT-WARNING       . * IF UNUSUAL DROP IN QUANTITY
81   END-IF                      . * INFORM THE MANAGEMENT
83 END-PROC
84 *
85 REPORT   AUDIT-TRAIL              LINESIZE 80
86   TITLE   1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
87   LINE    1 PART-NUMBER  ITEM-LAST-INVENTORY-DATE      -
               ITEM-LAST-INVENTORY-QUANTITY               -
               NEW-DATE     NEW-QUANT
88 *
89 REPORT   MGMT-WARNING             LINESIZE 80
```

```
90   TITLE   1 'INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY'
91   LINE    1 PART-NUMBER  LOCATION-CITY                -
               ITEM-LAST-INVENTORY-QUANTITY              -
               NEW-QUANT        PERCENT-DROP


11/20/83         INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL     PAGE   1

                          LAST         LAST         NEW         NEW
             PART      INVENTORY    INVENTORY    INVENTORY   INVENTORY
            NUMBER       DATE       QUANTITY      DATE      QUANTITY

          001-84-900    9/30/81          86     10/22/81         81
          001-85-200    8/31/81       2,210      9/27/81      2,103
          001-85-400    8/31/81         109     11/10/81         73
          001-86-600   10/30/81       3,403     11/19/81      3,401
          001-88-800   10/30/81       3,952     11/06/81      4,027
          001-79-000    9/30/81         385     10/15/81        434
          001-83-800    9/30/81         439     11/05/81        311
          001-84-200    9/30/81         653     11/15/81        472
          001-78-200    9/30/81         385     10/31/81        312

------------------------------------------------------------------------

11/20/83    INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY    PAGE    1

                                    LAST         NEW        PERCENT
             PART                INVENTORY    INVENTORY    DROP IN
            NUMBER      CITY     QUANTITY     QUANTITY    INVENTORY

          001-85-400   INDIANP        109          73        33.02
          001-83-800   INDIANP        439         311        29.15
          001-84-200   MEMPHIS        653         472        27.71
```

Reorder Notification Report

The Materials Department needs a program that reorders parts automatically when quantities get below a specified level. The program should provide three reports:

- A master activity report for the materials department,
- A set of purchase orders to initiate the ordering, and
- A receiving report for each warehouse that 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 the product still provide for a simple program. Each record in the inventory master is read.

- If the item does not require reordering, it is output as it is to the new master file.
- If a reorder is required, the desired quantity is established, the LAST-PURCHASE data is updated, an extended total for the item is calculated, the reports are written, and the updated master file record is output.

The three reports generated from this program demonstrate the power and flexibility of the product. The first report is a simple control report that lists all items ordered.

Note: Use the SUM statement to explicitly specify which fields to total at control breaks. It does not make sense to total the purchase quantity or estimated item price.

The second report demonstrates how a form with variable information is generated. All data that is constant on a page is defined in a long TITLE. Variable information is defined through LINE statements. Final totals are suppressed. A new page and renumbering are requested at each vendor control break.

Note: The use of control variables is in the title lines.

The final report is again a simple control report, but controlled on warehouse location, instead of vendor.

Note: Again, the use of the control variable is on the title line.

```
 1 *
 2 *   EXAMPLE 14.16
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5 %INVMSTR
44 *
45   PO#                S  10 N  HEADING('PURCHASE' 'ORDER' 'NUMBER')
46     PO-DATE      PO#     6 N
47     PO-SEQ       PO# +6  4 N
48   EXTENDED-TOTAL     W   5 P 2  HEADING('EXTENDED' 'TOTAL')
49 *
50 FILE    NEWMSTR   FB(200 3000)
51 *
52 JOB
53 *
54   IF ITEM-LAST-INVENTORY-QUANTITY > 4.0 * ITEM-REORDER-POINT
55     PUT  NEWMSTR  FROM INVMSTR  . *  OUTPUT NEW MASTER RECORD IF NO
57     GOTO JOB                    . *  CHANGE, AND GET NEXT RECORD
59   END-IF
60 *
61   IF ITEM-LAST-INVENTORY-QUANTITY < ITEM-REORDER-POINT
62     LAST-PURCHASE-QUANTITY = 1.2 * LAST-PURCHASE-QUANTITY
63   END-IF
64 *
65   %GETDATE  PO-DATE           . * GET DATE IN   MMDDYY  FORMAT
84   LAST-PURCHASE-DATE = PO-DATE . *  SET NEW PURCHASE DATE
```

```
 86   EXTENDED-TOTAL = LAST-PURCHASE-QUANTITY * LAST-PURCHASE-PRICE
 87 *
 88   PRINT  ACTIVITY-REPORT      . * PRINT MASTER ACTIVITY REPORT
 90   PRINT  PURCHASE-ORDERS      . * PRINT PURCHASE ORDERS
 92   PRINT  RECEIVING-REPORTS    . * PRINT RECEIVING REPORTS
 94 *
 95   PUT  NEWMSTR  FROM INVMSTR  . * OUTPUT UPDATED FILE
 97 *
 98 *
 99 REPORT   ACTIVITY-REPORT   SKIP 1  SUMCTL TAG   LINESIZE 80
100   SEQUENCE  VENDOR-NUMBER  PART-NUMBER
101   CONTROL   VENDOR-NUMBER
102   SUM       EXTENDED-TOTAL
103   TITLE   1 'PURCHASE ORDER ACTIVITY BY VENDOR'
104   HEADING   LAST-PURCHASE-QUANTITY 'QUANTITY'
105   HEADING   LAST-PURCHASE-PRICE    ('ESTIMATED' 'PRICE')
106   LINE    1 VENDOR-NUMBER  PART-NUMBER  -
                LAST-PURCHASE-QUANTITY  LAST-PURCHASE-PRICE  -
                EXTENDED-TOTAL
107   LINE    2 VENDOR-LOCATION-CITY -2 VENDOR-LOCATION-STATE -
                POS 2  PART-DESCRIPTION
108 *
109 REPORT  PURCHASE-ORDERS    NOADJUST  SKIP 1  SUMCTL NONE  LINESIZE 80
110   SEQUENCE  VENDOR-NUMBER  PART-NUMBER
111   CONTROL   FINAL NOPRINT  VENDOR-NUMBER RENUM
112   SUM       EXTENDED-TOTAL
113   TITLE   1 COL 25   'ABC COMPANY'
114   TITLE   2 COL 23   'PURCHASE ORDER'
115   TITLE   4 COL  1   'PO#' PO#
116   TITLE   6 COL  1   'VENDOR' VENDOR-NUMBER
117   TITLE   7 COL 10   VENDOR-LOCATION-CITY  -2  VENDOR-LOCATION-STATE
118   HEADING   LAST-PURCHASE-QUANTITY 'QUANTITY'
119   HEADING   LAST-PURCHASE-PRICE    ('ESTIMATED' 'PRICE')
120   LINE    1 PART-NUMBER  +10  -
                LAST-PURCHASE-QUANTITY  LAST-PURCHASE-PRICE  -
                EXTENDED-TOTAL
121   LINE    2 PART-DESCRIPTION
122   BEFORE-BREAK. PROC
124     PO-SEQ = PO-SEQ + 1           . * INCREMENT PO NUMBER
126   END-PROC
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

VENDOR NUMBER	PART NUMBER	QUANTITY	ESTIMATED PRICE	EXTENDED TOTAL
00-00-0-562 MILW WI	001-78-200 AIR BRAKES	600	29.95	17,970.00
	001-79-000 AXLE SHAFTS	600	29.95	17,970.00
MILW WI				
	001-85-400 DRIVE SHAFTS	300	40.00	12,000.00
MILE WI				
VENDOR-NUMBER TOTAL				47,940.00
00-00-9-128 BAY CIT MI	001-84-900 CYLINDER SLEEVES	600	16.29	9,774.00
VENDOR-NUMBER TOTAL				9,774.00
00-03-4-091 PHIL PA	001-84-200 BUMPERS	1,000	45.50	45,500.00
VENDOR-NUMBER TOTAL				45,500.00
10-03-0-443 LVILLE KY	000-81-190 DESKS, STEEL	360	195.69	70,448.40
VENDOR-NUMBER TOTAL				70,448.40

11/20/83 PURCHASE ORDER ACTIVITY BY VENDOR PAGE 2

VENDOR NUMBER	PART NUMBER	QUANTITY	ESTIMATED PRICE	EXTENDED TOTAL
34-89-7-210 DES MOI IA	000-53-100 REFRIGERATORS, HOUSEHOLD	2	450.67	901.34
VENDOR-NUMBER TOTAL				901.34
54-96-3-251 HOUST TX	000-11-576 MACHINES, CALCULATING	1,008	59.88	60,359.04
VENDOR-NUMBER TOTAL				60,359.04
65-49-8-318 TUCS AZ	000-82-150 TABLES, PICNIC	250	95.80	23,950.00
VENDOR-NUMBER TOTAL				23,950.00
FINAL TOTAL				258,872.78

```
--------------------------------------------------------------------------------

11/20/83            ABC COMPANY                                    PAGE    1
                  PURCHASE ORDER

PO#   1120810000

VENDOR   00-00-0-562
         MILW    WI


   PART                           ESTIMATED        EXTENDED
  NUMBER               QUANTITY     PRICE           TOTAL

001-78-200                600         29.95          17,970.00
AIR BRAKES

001-79-000                600         29.95          17,970.00
AXLE SHAFTS

001-85-400                300         40.00          12,000.00
DRIVE SHAFTS

                                                     47,940.00

--------------------------------------------------------------------------------

11/20/83            ABC COMPANY                                    PAGE    1
                  PURCHASE ORDER

PO#   1120810001

VENDOR   00-00-9-128
         BAY CIT MI


   PART                           ESTIMATED        EXTENDED
  NUMBER               QUANTITY     PRICE           TOTAL

001-84-900                600         16.29           9,774.00
CYLINDER SLEEVES

                                                      9,774.00

--------------------------------------------------------------------------------

11/20/83            ABC COMPANY                                    PAGE    1
                  PURCHASE ORDER

PO#   1120810002

VENDOR   00-03-4-091
         PHIL    PA


   PART                           ESTIMATED        EXTENDED
  NUMBER               QUANTITY     PRICE           TOTAL

001-84-200              1,000         45.50          45,500.00
BUMPERS

                                                     45,500.00
```

```
--------------------------------------------------------------------------

11/20/83            ABC COMPANY                                  PAGE    1
                  PURCHASE ORDER

PO#   1120810003

VENDOR   10-03-0-443
         LVILLE  KY


   PART                           ESTIMATED        EXTENDED
  NUMBER              QUANTITY     PRICE            TOTAL

000-81-190               360        195.69          70,448.40
DESKS, STEEL

                                                    70,448.40

--------------------------------------------------------------------------

11/20/83       RECEIVING REPORT FOR   E MOLIN   WAREHOUSE       PAGE    1


                      VENDOR         PART
                      NUMBER        NUMBER      QUANTITY

                   10-03-0-443   000-81-190         360
                   34-89-7-210   000-53-100           2
                   65-49-8-318   000-82-150         250

--------------------------------------------------------------------------
11/20/83       RECEIVING REPORT FOR   INDIANP   WAREHOUSE       PAGE    1


                      VENDOR         PART
                      NUMBER        NUMBER      QUANTITY

                   00-00-0-562   001-78-200         600
                   00-00-0-562   001-79-000         600
                   00-00-0-562   001-85-400         300
                   00-00-9-128   001-84-900         600

--------------------------------------------------------------------------
11/20/83       RECEIVING REPORT FOR   MEMPHIS   WAREHOUSE       PAGE    1


                      VENDOR         PART
                      NUMBER        NUMBER      QUANTITY

                   00-03-4-091   001-84-200       1,000

--------------------------------------------------------------------------
11/20/83       RECEIVING REPORT FOR   MUSKEGN   WAREHOUSE       PAGE    1


                      VENDOR         PART
                      NUMBER        NUMBER      QUANTITY

                   54-96-3-251   000-11-576       1,008
```

Chapter

15 Advanced Techniques

This chapter provides examples of some of the advanced processing techniques available in CA-Easytrieve Plus. These examples illustrate the use of a more complex operating system interface.

A background in data processing is required to fully understand some of the concepts and techniques. If you do not have this background, you should find someone with data processing experience to assist you.

The examples in this chapter stress one or two particular processing techniques. Because of this, no attempt has been made to provide examples as complete as the ones in the "Basic Examples" chapter. These examples do not have any situation posed, as in the "Basic Examples" chapter. The preamble simply describes the processing techniques.

The topics covered in this chapter include:

- GET/PUT of sequential and VSAM files
- Random access of VSAM files
  READ
  WRITE (add/replace/delete)
  POINT
  Path processing with non-unique keys
- SORT command
  SORT exit
- Synchronized file processing
- Advanced report features
  Control LEVEL
  S-fields
  TERMINATION procedure
  ENDPAGE procedure
  SUMMARY file
- Processing JCL parameters
- Macro definition and processing.

Selected Control Break Processing

Sometimes it is desirable to perform processing at control breaks, based on which variable caused the break. For example, it can be useful to output final break information in a different format from the one used for intermediate breaks. This example lists employee totals by branch. The normal summary line for the final break (LEVEL = 3) is suppressed, and a DISPLAY statement is used instead.

```
 1 *
 2 *   EXAMPLE 15.1
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 *
36 *
37 JOB
38   PRINT  EMPLOYEE-TALLY        . * PRINT REPORT FOR ALL EMPLOYEES
40 *
41 REPORT  EMPLOYEE-TALLY  SUMMARY  LINESIZE 80
42   SEQUENCE  REGION  BRANCH
43   CONTROL   FINAL NOPRINT  REGION  BRANCH
44   TITLE  1  'EMPLOYEE TALLY BY REGION AND BRANCH'
45   HEADING   TALLY  ('NUMBER OF' 'EMPLOYEES')
46   LINE   1  REGION  BRANCH  TALLY
47 *
48   BEFORE-BREAK.  PROC
50     IF  LEVEL = 3               . * IF FINAL BREAK
52       DISPLAY  SKIP 3 'TOTAL COMPANY EMPLOYEES:' TALLY
53     END-IF
54   END-PROC
```

```
1/24/84             EMPLOYEE TALLY BY REGION AND BRANCH       PAGE      1

                                                 NUMBER OF
                          REGION    BRANCH       EMPLOYEES

                            1         01             2
                            1         02             2
                            1         03             2
                            1         04             4
                            1                       10

                            2         01             1
                            2         02             2
                            2         03             4
                            2         04             1
                            2         05             2
                            2                       10

                            3         01             4
                            3         02             6
                            3         03             6
                            3         04             3
                            3                       19

                            4         01             3
                            4         02             2
                            4         03             3
                            4         04             1
                            4                        9


TOTAL COMPANY EMPLOYEES:                    48
```

Summary File Processing

At times, it is desirable to order a report on a value that is the result of a summing operation for a previous report. For example, using the Inventory Master File, a previous report listed the value of the inventory at each warehouse, ordered on the warehouse's city location.

To highlight the most valuable stock locations, this report might be more useful ordered by the decreasing value of the inventory for each warehouse.

To do this, we use the multijob and summary file facilities. The first job generates a normal summary report by location, along with a special summary file that is used as input to the second job. For comparison purposes, the two reports are identical except for the ordering.

Remember, use this technique when a report must be ordered, based on values that are calculated across groups of input records. The process is easy using the multijob and the summary file facilities.

```
 1 *
 2 *   EXAMPLE 15.2
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5 %INVMSTR
44 *
45   TOTAL-VALUE   W  6 P 2   HEADING('TOTAL VALUE' 'OF PARTS')
46 *
47 *   SUMMARY FILE DEFINITION
48 *
49 *   THE SUMMARY FILE FROM THE INV-BY-CITY REPORT HAS THE FOLLOWING
50 *   GENERAL FORMAT:
51 *      CONTROL FIELDS + TALLY + TOTAL FIELDS
52 *
53 *   FOR THIS PARTICULAR FILE IT HAS THE FOLLOWING FORMAT:
54 *
55 *    LOCATION   LENGTH     FIELD
56 *       1          7       CONTROL FIELD - LOCATION-CITY
57 *       8         10       TALLY
58 *      18         10       TOTAL FIELD - TOTAL-VALUE
59 *
60 *   NOTICE IN THE BELOW FIELD DEFINITIONS HOW THE FIELDS ARE DEFINED
61 *   TO FIT WITHIN THE FORMAT.  SINCE THE TALLY VALUE WILL NOT EXCEED
62 *   SEVEN DIGITS, PARTS-IN-CITY ONLY DEFINES THAT PART.  THE SAME IS
63 *   TRUE FOR VALUE-IN-CITY.  THE LENGTH OF THE RECORD IS 28 BYTES AND
64 *   IT IS SPOOLED TO THE VIRTUAL FILE MANAGER.
65 *
66 FILE  SMYFIL  F 27  VIRTUAL
67   CITY           1  7 A
68   PARTS-IN-CITY  14  4 P 0    HEADING('NUMBER OF' 'PART TYPES')
69   VALUE-IN-CITY  22  6 P 2    HEADING('TOTAL VALUE' 'OF PARTS')
70 *
71 JOB
72   TOTAL-VALUE = LAST-PURCHASE-PRICE * ITEM-LAST-INVENTORY-QUANTITY
73 *
74   PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE
76 *
77 REPORT  INV-BY-CITY   SUMMARY  SUMFILE SMYFIL  LINESIZE 80
78   SEQUENCE  LOCATION-CITY
79   CONTROL   LOCATION-CITY
80   TITLE   1 'INVENTORY VALUE BY CITY'
```

```
81   HEADING   TALLY ('NUMBER OF' 'PART TYPES')
82   LINE    1 LOCATION-CITY  TALLY  TOTAL-VALUE
83 *
84 *
85 JOB  INPUT  SMYFIL
86   PRINT  SMY-BY-VALUE
87 *
88 REPORT  SMY-BY-VALUE               LINESIZE 80
89   SEQUENCE  VALUE-IN-CITY D
90   CONTROL
91   TITLE 'VALUE OF INVENTORY IN EACH CITY BY DECREASING VALUE'
92   LINE  CITY  PARTS-IN-CITY  VALUE-IN-CITY
```

```
11/24/83                 INVENTORY VALUE BY CITY             PAGE      1

                          NUMBER OF       TOTAL VALUE
                 CITY     PART TYPES        OF PARTS

                CHICAGO         2             233,379.60
                E MOLIN         5             160,522.97
                HAMMOND         1              81,982.00
                INDIANP         8             191,825.42
                KANS CT         1               9,737.02
                MAMMOND         1              25,009.00
                MEMPHIS         4             630,366.23
                MUSKEGN         3              32,794.14
                ST PAUL         2             162,160.05
                               27           1,527,776.43
```

```
--------------------------------------------------------------------------------
11/24/83    VALUE OF INVENTORY IN EACH CITY BY DECREASING VALUE  PAGE   1

                           NUMBER OF        TOTAL VALUE
                 CITY      PART TYPES         OF PARTS

                MEMPHIS              4           630,366.23
                CHICAGO              2           233,379.60
                INDIANP              8           191,825.42
                ST PAUL              2           162,160.05
                E MOLIN              5           160,522.97
                HAMMOND              1            81,982.00
                MUSKEGN              3            32,794.14
                MAMMOND              1            25,009.00
                KANS CT              1             9,737.02
                                    27         1,527,776.43
```

Special Report Processing Exits

Report processing provides several special processing exits. Two of these are useful for page and report annotation.

- ENDPAGE enables you to perform processing when the end of a logical page is reached. It is useful for footers or page totals. Specifying a PAGESIZE of 12 enables more than one of these small reports to fit onto one sheet of printer paper. The footer appears at the bottom of the page, regardless of the size of the report data.
- TERMINATION permits annotation at the end of the report. Typical uses are report routing information, special final total data, or hash totals.

Example 15.3 demonstrates the use of both of these facilities. An important consideration for the TERMINATION procedure is to limit field references to S-fields, control fields, and total fields.

```
 1 *
 2 *   EXAMPLE 15.3
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5 %INVMSTR
44 *
45 JOB
46   PRINT  INV-BY-CITY             . * SELECT EACH RECORD IN FILE
48 *
49 REPORT  INV-BY-CITY  PAGESIZE 12  LINESIZE 80
50   SEQUENCE  LOCATION-CITY PART-NUMBER
51   CONTROL   FINAL NOPRINT   LOCATION-CITY  NEWPAGE
52   TITLE   1 'INVENTORY FOR ' LOCATION-CITY ' BY PART NUMBER'
53   LINE    1 PART-NUMBER  PART-DESCRIPTION
54 *
55   ENDPAGE. PROC
57     DISPLAY  'CONFIDENTIAL COMPANY INFORMATION'
58   END-PROC
59 *
60   TERMINATION. PROC
62     DISPLAY  NEWPAGE  'ROUTE  REPORT  TO:'
63     DISPLAY  SKIP 2   'R. M. HODGES'
64     DISPLAY           'MATERIALS PROCUREMENT'
65   END-PROC


11/24/83        INVENTORY FOR    CHICAGO    BY PART NUMBER    PAGE      1


                  PART
                 NUMBER              PART DESCRIPTION

               000-15-428   BOOKS, SCHOOL COPY
               000-16-490   BAGS, GOLF CLUB


CONFIDENTIAL COMPANY INFORMATION
```

```
--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    E MOLIN    BY PART NUMBER    PAGE      2


                   PART
                  NUMBER              PART DESCRIPTION

                000-10-944   PANEL, SOLAR
                000-53-100   REFRIGERATORS, HOUSEHOLD
                000-79-740   BEDS, WOODEN
                000-81-190   DESKS, STEEL
                000-82-150   TABLES, PICNIC

CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    HAMMOND    BY PART NUMBER    PAGE      3


                   PART
                  NUMBER              PART DESCRIPTION

                000-70-750   CARPETS, FABRIC (20' X 40')



CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    INDIANP    BY PART NUMBER    PAGE      4


                   PART
                  NUMBER              PART DESCRIPTION

                000-15-980   FAUCETS, BATH TUB
                000-51-260   PIPE, IRON OR STEEL (3" X 96")
                000-60-680   BATTERIES, ELECTRIC DRY CELL
                001-78-200   AIR BRAKES
                001-79-000   AXLE SHAFTS
                001-83-800   BRAKE DRUMS

CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    INDIANP    BY PART NUMBER    PAGE      5


                   PART
                  NUMBER              PART DESCRIPTION

                001-84-900   CYLINDER SLEEVES
                001-85-400   DRIVE SHAFTS


CONFIDENTIAL COMPANY INFORMATION
```

```
--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    KANS CT    BY PART NUMBER    PAGE      6


                  PART
                 NUMBER              PART DESCRIPTION

               000-17-037   SIDING, ALUMINUM (24" X 72")


CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    MAMMOND    BY PART NUMBER    PAGE      7


                  PART
                 NUMBER              PART DESCRIPTION

               000-19-360   WALLBOARD, FIBERBOARD (48" X 96")


CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    MEMPHIS    BY PART NUMBER    PAGE      8


                  PART
                 NUMBER              PART DESCRIPTION

               001-84-200   BUMPERS
               001-85-200   DOORS
               001-86-600   FENDERS
               001-88-800   HUBS

CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------------

11/24/83        INVENTORY FOR    MUSKEGN    BY PART NUMBER    PAGE      9


                  PART
                 NUMBER              PART DESCRIPTION

               000-11-576   MACHINES, CALCULATING
               000-12-268   DRYERS, HAIR
               000-62-270   HUMIDIFIERS, PORTABLE


CONFIDENTIAL COMPANY INFORMATION
```

```
11/24/83        INVENTORY FOR    ST PAUL    BY PART NUMBER    PAGE     10

                   PART
                  NUMBER              PART DESCRIPTION

                 000-12-440   MOWERS, LAWN
                 000-13-325   SAWS, CHAIN


CONFIDENTIAL COMPANY INFORMATION

--------------------------------------------------------------------------

ROUTE  REPORT  TO:


R. M. HODGES
MATERIALS PROCUREMENT
```

Sorting Input Files

The CA-Easytrieve Plus SORT facility is useful to reorder a file or its subset prior to processing. Normally, the SEQUENCE statement is used to order reports during report processing. However, if the sorted file is to be kept, or if more than one report needs to be output in the same order, a SORT activity is more efficient. For example, if you intend to generate five reports in the same order, the input file can be sorted once and no SEQUENCE statements are required. One sort is performed instead of five. Also, this eliminates report spooling if each print file is routed to a different logical printer. With large files, this approach dramatically reduces processing time and temporary disk work space.

In this example, we select all inventory records for items that cost more than $200 and sort them by commodity group. Then, we generate control, detail, and summary reports without SEQUENCE statements. A BEFORE sort exit performs the required record selection.

```
 1 *
 2 *   EXAMPLE 15.4
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5   VENDOR-CITY       112  7 A
 6   PART-PRICE         62  4 P 2
 7 *
 8 FILE    SRTMSTR   F 200   VIRTUAL
 9 %INVMSTR
48 *
49 SORT  INVMSTR  TO  SRTMSTR  BEFORE SCAN-INV  USING VENDOR-CITY
50 *
51   SCAN-INV. PROC
53     IF  PART-PRICE > 200.00      . * SELECT PART RECORDS IF
55       SELECT                     . * THEY COST MORE THAN $200
57     END-IF
58   END-PROC
59 *
60 *
```

```
61 JOB  INPUT SRTMSTR
62   PRINT  DETAIL-RPT
63   PRINT  CONTROL-RPT             . * PRINT ALL REPORTS
65   PRINT  SUMMARY-RPT
66 *
67 REPORT  DETAIL-RPT                LINESIZE 80
68   TITLE   1 'DETAIL LIST OF ALL PARTS SELLING FOR MORE THAN $200'
69   LINE    1 VENDOR-LOCATION-CITY   PART-NUMBER  PART-DESCRIPTION
70 *
71 REPORT  CONTROL-RPT               LINESIZE 80
72   CONTROL   VENDOR-LOCATION-CITY
73   TITLE   1 'CONTROLLED LIST OF PARTS SELLING FOR MORE THAN $200'
74   TITLE   2 'BY VENDOR LOCATION'
75   LINE    1 VENDOR-LOCATION-CITY  PART-NUMBER  PART-DESCRIPTION TALLY
76 *
77 REPORT  SUMMARY-RPT   SUMMARY     LINESIZE 80
78   CONTROL   VENDOR-LOCATION-CITY
79   TITLE   1 'SUMMARY LIST OF PARTS SELLING FOR MORE THAN $200'
80   TITLE   2 'BY VENDOR LOCATION'
81   LINE    1 VENDOR-LOCATION-CITY  TALLY


11/25/83     DETAIL LIST OF ALL PARTS SELLING FOR MORE THAN $200  PAGE  1


VENDOR       PART
CITY       NUMBER              PART DESCRIPTION

DES MOI   000-53-100   REFRIGERATORS, HOUSEHOLD
GR BAY    000-12-440   MOWERS, LAWN
LVILLE    000-81-190   DESKS, STEEL
NEWARK    000-70-750   CARPETS, FABRIC (20' X 40')
TUPEL     000-79-740   BEDS, WOODEN

--------------------------------------------------------------------------------

11/25/83    CONTROLLED LIST OF PARTS SELLING FOR MORE THAN $200  PAGE   1
                             BY VENDOR LOCATION


      VENDOR       PART
       CITY       NUMBER              PART DESCRIPTION             TALLY

      DES MOI   000-53-100   REFRIGERATORS, HOUSEHOLD
      DES MOI                                                         1

      GR BAY    000-12-440   MOWERS, LAWN
      GR BAY                                                          1

      LVILLE    000-81-190   DESKS, STEEL
      LVILLE                                                          1

      NEWARK    000-70-750   CARPETS, FABRIC (20' X 40')
      NEWARK                                                          1

      TUPEL     000-79-740   BEDS, WOODEN
      TUPEL                                                           1

                                                                      5
```

```
11/25/83     SUMMARY LIST OF PARTS SELLING FOR MORE THAN $200    PAGE   1
                              BY VENDOR LOCATION


                             VENDOR
                              CITY     TALLY

                             DES MOI        1
                             GR BAY         1
                             LVILLE         1
                             NEWARK         1
                             TUPEL          1
                                            5
```

Synchronized File Facility: File Update

In the "Basic Examples" chapter, we illustrated the technique for performing an update of the Inventory Master File using a table (Example 14.15). In this example, we demonstrate the use of the synchronized file processing facility to perform this update.

The synchronized update process requires the files to be in the same order by the defined key. This requirement causes us to add two Sort activities to the job. From this example, you can see that the table file method is simpler for moderate updates.

A more complete example of synchronized file updating is in the "Project Management System" chapter.

```
 1 *
 2 *   EXAMPLE 15.5
 3 *
 4 FILE    INVMSTR   FB(200 3000)
 5   MSTR-PART#  36 8 N
 6 *
 7 FILE    SRTMSTR   F 200   VIRTUAL
 8 %INVMSTR
47 *
48   PERCENT-DROP   W 3 P 2   HEADING('PERCENT' 'DROP')
49 *
50 FILE    NEWMSTR   FB(200 3000)
51 *
52 FILE    UPDTRAN   F 80
53   TRAN-PART#   1 8 N
54 *
55 FILE    SRTTRAN   F 80  VIRTUAL
56   TRAN-PART-NBR   1 8 N
57   TRAN-INV-DATE  10 6 N    HEADING('NEW' 'INVENTORY' 'DATE') -
                              MASK('Z9/99/99')
58   TRAN-INV-QUAN  17 5 N 0  HEADING('NEW' 'INVENTORY' 'QUANTITY')
59 *
60 *
61 SORT  INVMSTR  TO  SRTMSTR  USING MSTR-PART#
62 *
63 SORT  UPDTRAN  TO  SRTTRAN  USING TRAN-PART#
64 *
65 JOB   INPUT (SRTMSTR  KEY PART-NUMBER  -
                SRTTRAN  KEY TRAN-PART-NBR)
```

```
 66 *
 67   IF  MATCHED                  . * IF MATCH FOUND
 69     PRINT  AUDIT-TRAIL         . * OUTPUT AUDIT TRAIL
 71     PERFORM EXCESS-CHECK       . * CHECK FOR LARGE QUANT VARIATION
 73     ITEM-LAST-INVENTORY-DATE = TRAN-INV-DATE  . * UPDATE DATE AND
 75     ITEM-LAST-INVENTORY-QUANTITY = TRAN-INV-QUAN  . * QUANTITY
 77   END-IF
 78 *
 79 *        OUTPUT NEW MASTER IF MATCHED OR MASTER AND NO TRANSACTION
 80 *        DISPLAY ERROR MSG FOR A TRANSACTION WITH NO MATCHING MASTER
 81 *
 82   IF  SRTMSTR
 83     PUT  NEWMSTR  FROM SRTMSTR     . * OUTPUT UPDATED FILE
 85   ELSE
 86     DISPLAY 'NO MASTER FILE RECORD FOR TRANSACTION ' TRAN-PART-NBR
 87   END-IF
 88 *
 89 EXCESS-CHECK.  PROC
 91   IF TRAN-INV-QUAN < .8 * ITEM-LAST-INVENTORY-QUANTITY
 92     PERCENT-DROP = 100 -
                 - (TRAN-INV-QUAN * 100 / ITEM-LAST-INVENTORY-QUANTITY)
 93     PRINT  MGMT-WARNING       . * IF UNUSUAL DROP IN QUANTITY
 95   END-IF                      . * INFORM THE MANAGEMENT
 97 END-PROC
 98 *
 99 REPORT   AUDIT-TRAIL              LINESIZE 80
100   TITLE   1 'INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL'
101   LINE    1 PART-NUMBER  ITEM-LAST-INVENTORY-DATE      -
                ITEM-LAST-INVENTORY-QUANTITY              -
                TRAN-INV-DATE  TRAN-INV-QUAN
102 *
103 REPORT   MGMT-WARNING            LINESIZE 80
104   TITLE   1 'INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY'
105   LINE    1 PART-NUMBER  LOCATION-CITY                 -
                ITEM-LAST-INVENTORY-QUANTITY              -
                TRAN-INV-QUAN    PERCENT-DROP


11/25/83        INVENTORY MASTER FILE UPDATE -- AUDIT TRAIL      PAGE   1

                          LAST         LAST         NEW         NEW
            PART       INVENTORY    INVENTORY    INVENTORY   INVENTORY
           NUMBER        DATE       QUANTITY      DATE      QUANTITY

         001-78-200    9/30/81          385    10/31/81        312
         001-79-000    9/30/81          385    10/15/81        434
         001-83-800    9/30/81          439    11/05/81        311
         001-84-200    9/30/81          653    11/15/81        472
         001-84-900    9/30/81           86    10/22/81         81
         001-85-200    8/31/81        2,210     9/27/81      2,103
         001-85-400    8/31/81          109    11/10/81         73
         001-86-600   10/30/81        3,403    11/19/81      3,401
         001-88-800   10/30/81        3,952    11/06/81      4,027
```

```
--------------------------------------------------------------------------

11/25/83     INVENTORY WITH A 20% OR GREATER DROP IN QUANTITY    PAGE   1


                                          LAST          NEW
                PART                   INVENTORY     INVENTORY   PERCENT
                NUMBER      CITY       QUANTITY      QUANTITY     DROP

              001-83-800   INDIANP          439           311      29.15
              001-84-200   MEMPHIS          653           472      27.71
              001-85-400   INDIANP          109            73      33.02
```

Reformat Printed Output from IDCAMS

Frequently, it is useful to read the printed output of another program and format the information in a different manner. This example shows how a CA-Easytrieve Plus job reads the output of an IDCAMS utility run, processes the data, and then generates a report that would not otherwise be available.

This example shows how to extract VSAM CI/CA split information for review. The Assignment statement, using the OR feature, is a way of converting the hyphens (x'60') on the report to EBCDIC zeros (x'FO').

```
 1 *
 2 *    EXAMPLE 15.6
 3 *
 4 FILE    AMSINFO   VB(125 629)  WORKAREA 125
 5   P-CLEAR            1 125 A
 6   P-LIST-DATA        1  70 A
 7   P-CLUSTER-ID       2   9 A
 8   P-CLUSTER-NAME    18  40 A
 9   P-COMPONENT-ID     5   5 A
10   P-SPLIT-ID        38   7 A
11   P-SPLIT-TYPE      45   2 A
12   P-SPLIT-COUNT     57   5 N
13 *
14 *   WORKING STORAGE FIELDS
15 *
16   CLUSTER            W  40 A
17   COMPONENT          W   5 A
18   SPLIT-TYPE         W   2 A
19   SPLIT-COUNT        W   5 N 0
20 *
21 JOB
22   PRINT  INPUT-DATA
23 *
24   IF P-CLUSTER-ID = 'CLUSTER -'
25     CLUSTER = P-CLUSTER-NAME
26   END-IF
27 *
28   IF P-COMPONENT-ID = 'DATA ', 'INDEX'
29     COMPONENT = P-COMPONENT-ID
30   END-IF
31 *
32   IF P-SPLIT-ID = 'SPLITS-'
33     SPLIT-TYPE  = P-SPLIT-TYPE
34     SPLIT-COUNT = P-SPLIT-COUNT OR X'F0F0F0F0F0'
35   END-IF
```

```
36 *
37   IF SPLIT-COUNT NOT ZERO
38     PRINT  SPLIT-INFO
39     SPLIT-COUNT = 0
40   END-IF
41 *
42   MOVE SPACES TO P-CLEAR     * CLEAR WORKAREA AFTER EACH RECORD
44 *
45 REPORT  SPLIT-INFO  SPREAD   LINESIZE 80
46   SEQUENCE  SPLIT-TYPE  CLUSTER  COMPONENT
47   CONTROL   SPLIT-TYPE
48   TITLE   1 'VSAM CI/CA SPLIT INFORMATION'
49   LINE    1 SPLIT-TYPE  SPLIT-COUNT  COMPONENT  CLUSTER
50 *
51 REPORT  INPUT-DATA  NOHEADING  LINESIZE 80   LIMIT 50
52   TITLE   1 'TYPICAL INPUT DATA FOR RUN - FIRST 50 LINES'
53   LINE    1 P-LIST-DATA
--------------------------------------------------------------------------------
SORT (DEVICE SYSDA  ALTSEQ NO MSG DEFAULT  MEMORY MAX   WORK   3)  VFM (   16  D
--------------------------------------------------------------------------------
```

```
11/24/83                  VSAM CI/CA SPLIT INFORMATION              PAGE      1

SPLIT-TYPE   SPLIT-COUNT   COMPONENT                   CLUSTER

    CI              23       DATA      ABELMAN.MACRO.FILE
                     4       DATA      VARVERI.MACRO.LIBRARY
    CI              27

                    27



11/24/83           TYPICAL INPUT DATA FOR RUN - FIRST 50 LINES      PAGE      1


    1IDCAMS  SYSTEM SERVICES                                          TIM
    0
      LISTCAT  CATALOG(USER53.USERCAT)  ALL
    1IDCAMS  SYSTEM SERVICES                                          TIM
    -                            LISTING FROM CATALOG -- USER53.USERCAT
    0CLUSTER ------- ABELMAN.MACRO.FILE
          HISTORY
            OWNER-IDENT-------(NULL)     CREATION----------81.237
            RELEASE----------------2     EXPIRATION--------81.365
          PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
          ASSOCIATIONS
            DATA-----VSAMDSET.TE18B950.DFD81237.T927607B.TE18B950
            INDEX----VSAMDSET.TE18D390.DFD81237.T927607B.TE18D390
    0   DATA ------- VSAMDSET.TE18B950.DFD81237.T927607B.TE18B950
          HISTORY
            OWNER-IDENT-------(NULL)     CREATION----------81.237
            RELEASE----------------2     EXPIRATION--------00.000
          PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
          ASSOCIATIONS
            CLUSTER--ABELMAN.MACRO.FILE
          ATTRIBUTES
            KEYLEN----------------30     AVGLRECL-------------110     BUFS
            RKP-------------------0      MAXLRECL-------------110     EXCP
            SHROPTNS(1,3)      SPEED     SUBALLOC         NOERASE     INDE
            UNORDERED          REUSE     NONSPANNED
          STATISTICS
            REC-TOTAL------------138     SPLITS-CI-------------23     EXCP
            REC-DELETED----------884     SPLITS-CA-------------0      EXTE
            REC-INSERTED---------949     FREESPACE-%CI---------0      SYST
            REC-UPDATED-----------14     FREESPACE-%CA---------0
            REC-RETRIEVED-------1828     FREESPC-BYTES-----434176
          ALLOCATION
            SPACE-TYPE------CYLINDER     HI-ALLOC-RBA------491520
            SPACE-PRI--------------1     HI-USED-RBA-------491520
            SPACE-SEC--------------1
          VOLUME
            VOLSER------------USER53     PHYREC-SIZE---------2048     HI-A
            DEVTYPE------X'3050200B'     PHYRECS/TRK------------8     HI-U
            VOLFLAG------------PRIME     TRACKS/CA-------------30
            EXTENTS:
            LOW-CCHH-----X'01950000'     LOW-RBA----------------0     TRAC
            HIGH-CCHH----X'0195001D'     HIGH-RBA----------491519
    0   INDEX ------ VSAMDSET.TE18D390.DFD81237.T927607B.TE18D390
          HISTORY
            OWNER-IDENT-------(NULL)     CREATION----------81.237
            RELEASE----------------2     EXPIRATION--------00.000
          PROTECTION-PSWD-----(NULL)     RACF----------------(NO)
          ASSOCIATIONS
            CLUSTER--ABELMAN.MACRO.FILE
          ATTRIBUTES
```

VSAM File Processing

The next five examples (15.7 through 15.11) demonstrate the processing of VSAM files by a complete, flexible facility for processing VSAM structures, including access to ESDS, RRDS, and KSDS data sets and any defined PATH. (A thorough understanding of VSAM file concepts is required.)

To demonstrate the use of CA-Easytrieve Plus with VSAM files, two data structures are built from the Personnel Master File. The first structure is an ESDS cluster, with the same format as the sequential file used in the "Basic Examples" chapter. In addition, an alternate index is built across the ESDS, keyed on the employee number. A path is defined for the combination of the alternate index and the base ESDS cluster.

The second structure is a KSDS cluster, built from the Personnel Master File, keyed on the employee number. A non-unique alternate index is built across the KSDS cluster, keyed on the department number. Also, a path is defined for this alternate index and KSDS cluster combination.

Three steps are required to build the above two structures:

1. Define the base clusters through IDCAMS (Example 15.7A).
2. Load the base clusters by means of CA-Easytrieve Plus (Example 15.7B).
3. Define and build the alternate indexes, and define the paths through IDCAMS (Example 15.7C).

Defining and Loading VSAM Data Sets with Alternate Indexes

This example shows how to define and load a VSAM data set using CA-Easytrieve Plus. In this example, we build the two VSAM structures described previously. First, we define the base clusters through IDCAMS.

```
Define Base Clusters through IDCAMS

 *   EXAMPLE 15.7A

 DELETE (RETSYS$.PERSNL.KSDS/MSTPER)
 DELETE (RETSYS$.PERSNL.ESDS/MSTPER)
 SET MAXCC = 0
 DEFINE CLUSTER -
  (NAME(RETSYS$.PERSNL.KSDS) -
   RECORDS(50 10)  VOLUMES(USER53) -
   KEYS(5  8)      OWNER(EZTP)  -
   RECORDSIZE(150 150)                -
   UPDATEPW(UPDPER) MASTERPW(MSTPER)) -
       DATA (NAME(RETSYS$.PERSNL.KSDS.DATA)) -
       INDEX(NAME(RETSYS$.PERSNL.KSDS.INDEX))
 DEFINE CLUSTER -
  (NAME(RETSYS$.PERSNL.ESDS) -
   RECORDS(50 10)  VOLUMES(USER53) -
   NONINDEXED      OWNER(EZTP)  -
   RECORDSIZE(150 150)                -
   UPDATEPW(UPDPER) MASTERPW(MSTPER)) -
       DATA (NAME(RETSYS$.PERSNL.ESDS.DATA))
```

Load Base Clusters

Next, a CA-Easytrieve Plus job is used to load the data into the VSAM clusters from the sequential version of the Personnel Master File. A SORT is required to order the KSDS input by employee number.

```
 1 *
 2 *   EXAMPLE 15.7B
 3 *
 4 FILE    PERSNL    FB(150 1800)
 5 %PERSNL
35 *
36 FILE    PERESDS   VS(ES  PASSWORD 'UPDPER'  CREATE)
37 *
38 FILE    PERKSDS   VS(PASSWORD 'UPDPER'   CREATE)
39 *
40 FILE    TWORK     F 150   VIRTUAL
41 *
42 JOB    FINISH  WRAP-UP
43   PUT  PERESDS FROM PERSNL       . * BUILD ESDS VERSION OF PERSNL
45 *
46   WRAP-UP.  PROC
48     DISPLAY  NEWPAGE  'TOTAL RECORDS WRITTEN TO PERESDS = ' -
                         RECORD-COUNT(PERESDS)
49   END-PROC
50 *
51 SORT  PERSNL TO TWORK    USING EMP#  . * SORT PERSNL INTO EMP# ORDER
53 *
54 JOB    FINISH  WRAP-UP
55   PUT   PERKSDS FROM TWORK       . * BUILD KSDS VERSION OF PERSNL
57 *
58   WRAP-UP.  PROC
60     DISPLAY  NEWPAGE  'TOTAL RECORDS WRITTEN TO PERKSDS = ' -
                         RECORD-COUNT(PERKSDS)
61   END-PROC

-------------------------------------------------------------------------

TOTAL RECORDS WRITTEN TO PERESDS =            48

-------------------------------------------------------------------------

TOTAL RECORDS WRITTEN TO PERKSDS =            48

-------------------------------------------------------------------------
```

Defining and Building Alternate Indexes and Define Paths

Now that the base clusters are built, use IDCAMS to define and build the alternate indexes, and also to define the paths.

```
 *    EXAMPLE 15.7C

DEFINE  ALTERNATEINDEX   -
        (NAME(RETSYS$.PERSNL.ESDS.AX)        -
         RELATE(RETSYS$.PERSNL.ESDS/MSTPER)  -
         RECORDS(50 10)   VOLUMES(USER53)    -
         KEYS(5 8)     MASTERPW(MSTPER)      -
         OWNER(PRO)    RECORDSIZE(17 34)     -
         REUSE   SPEED       UNIQUEKEY)      -
    DATA   -
        (NAME(RETSYS$.PERSNL.ESDS.AX.DATA))  -
    INDEX  -
        (NAME(RETSYS$.PERSNL.ESDS.AX.INDEX))
DEFINE  ALTERNATEINDEX   -
        (NAME(RETSYS$.PERSNL.KSDS.AX)        -
         RELATE(RETSYS$.PERSNL.KSDS/MSTPER)  -
         RECORDS(50 10)   VOLUMES(USER53)    -
         KEYS(3 97)    MASTERPW(MSTPER)      -
         OWNER(PRO)    RECORDSIZE(28 99)     -
         REUSE   SPEED    NONUNIQUEKEY)      -
    DATA   -
        (NAME(RETSYS$.PERSNL.KSDS.AX.DATA))  -
    INDEX  -
        (NAME(RETSYS$.PERSNL.KSDS.AX.INDEX))
BLDINDEX  -
    IDS(RETSYS$.PERSNL.ESDS/MSTPER)  -
    ODS(RETSYS$.PERSNL.ESDS.AX/MSTPER)
BLDINDEX  -
    IDS(RETSYS$.PERSNL.KSDS/MSTPER)  -
    ODS(RETSYS$.PERSNL.KSDS.AX/MSTPER)
DEFINE  PATH  -
    (NAME(RETSYS$.PERSNL.ESDS.PATH)  -
    PATHENTRY(RETSYS$.PERSNL.ESDS.AX))
DEFINE  PATH  -
    (NAME(RETSYS$.PERSNL.KSDS.PATH)  -
    PATHENTRY(RETSYS$.PERSNL.KSDS.AX))
```

Updating a VSAM KSDS Cluster

This example demonstrates the random reading and updating of a VSAM KSDS cluster. The ESDS is in the original order of the Personnel Master file; it is a reproduction of the sequential version. The file is in order by region number.

In this example, we read all of the records with a region code of 1 from the ESDS. We then use the employee number as the key for the KSDS. The KSDS record is read, modified, and updated on the file. This is a typical random update operation using an input tickler file.

```
 1 *
 2 *   EXAMPLE 15.8
 3 *
 4 FILE    PERESDS   VS ES
 5   REGION-CODE    1 1 N
 6   EMP-NBR        9 5 N
 7 *
 8 FILE    PERKSDS   VS(PASSWORD 'UPDPER'   UPDATE)
 9 %PERSNL
39 *
40 JOB
41   IF  REGION-CODE GT 1
42     STOP                       . * STOP IF DONE WITH REGION CODE 1
44   END-IF
45 *
46   READ  PERKSDS   KEY EMP-NBR
47   PRINT  BEFORE-UPDATE         . * PRINT BEFORE-UPDATE INFO
49   PAY-GROSS = 1.05 * PAY-GROSS . * GIVE EVERYONE IN REGION 1 A 5% RAISE
51   WRITE PERKSDS   UPDATE       . * UPDATE THE FILE
53 *
54 REPORT  BEFORE-UPDATE     LINESIZE 80
55   SEQUENCE  EMP#
56   TITLE     1 'REGION 1 EMPLOYEES GROSS SALARIES BEFORE UPDATE'
57   LINE      1 EMP#  NAME-LAST  NAME-FIRST  PAY-GROSS
58 *
59 *
60 JOB   INPUT PERKSDS
61   IF  REGION = 1
62     PRINT  AFTER-UPDATE        . * SHOW UPDATED SALARIES
64   END-IF
65 *
66 REPORT  AFTER-UPDATE       LINESIZE 80
67   SEQUENCE  EMP#
68   TITLE     1 'REGION 1 EMPLOYEES GROSS SALARIES AFTER UPDATE'
69   LINE      1 EMP#  NAME-LAST  NAME-FIRST  PAY-GROSS
```

```
11/25/83      REGION 1 EMPLOYEES GROSS SALARIES BEFORE UPDATE    PAGE   1

                  EMPLOYEE     LAST      FIRST       GROSS
                   NUMBER      NAME       NAME        PAY

                   00370     NAGLE      MARY          554.40
                   01963     ARNOLD     LINDA         445.50
                   02200     BRANDOW    LYDIA         804.64
                   02688     CORNING    GEORGE        146.16
                   11357     LARSON     RODNEY        283.92
                   11467     BYER       JULIE         396.68
                   11473     BERG       NANCY         759.20
                   11602     MANHART    VIRGINIA      344.80
                   11931     TALL       ELAINE        492.26
                   12267     WIMN       GLORIA        373.60

------------------------------------------------------------------------------

11/25/83      REGION 1 EMPLOYEES GROSS SALARIES AFTER UPDATE  PAGE      1

                  EMPLOYEE     LAST      FIRST       GROSS
                   NUMBER      NAME       NAME        PAY

                   00370     NAGLE      MARY          582.12
                   01963     ARNOLD     LINDA         467.77
                   02200     BRANDOW    LYDIA         844.87
                   02688     CORNING    GEORGE        153.46
                   11357     LARSON     RODNEY        298.11
                   11467     BYER       JULIE         416.51
                   11473     BERG       NANCY         797.16
                   11602     MANHART    VIRGINIA      362.04
                   11931     TALL       ELAINE        516.87
                   12267     WIMN       GLORIA        392.28
```

Sequentially Reading VSAM File through Non-unique Alternate Index

This example reads employee records looking for a specified department. The POINT command is used against the department path to the KSDS. The STATUS parameter of the POINT command detects invalid department codes. The DO WHILE construct provides a simple method of reading records sequentially until the department code changes or we reach end-of-file.

```
 1 *
 2 *   EXAMPLE 15.9
 3 *
 4 FILE    DEPTCOD   F 80
 5   DEPARTMENT   1 3 N
 6 *
 7 FILE    PTHKSDS   VS
 8 %PERSNL
38 *
39 JOB
40   POINT  PTHKSDS  EQ DEPARTMENT  STATUS
41   IF  PTHKSDS:FILE-STATUS  NE 0
42     DISPLAY  'NO EMPLOYEES IN DEPARTMENT ' DEPARTMENT
43     GOTO JOB
44   END-IF
45 *
46   GET PTHKSDS                   . * GET FIRST DEPT RECORD
48   DO WHILE  PTHKSDS AND DEPARTMENT = DEPT . * LOOP WHILE VALID DEPT
50     PRINT  SELECTED-DEPARTMENTS     . * PRINT REPORT
52     GET   PTHKSDS                   . * GET NEXT RECORD IN DEPT
54   END-DO
55 *
56 REPORT  SELECTED-DEPARTMENTS   LINESIZE 80
57   CONTROL   DEPT NOPRINT
58   TITLE     1 'SELECTED DEPARTMENTS VIA VSAM PATH PROCESSING'
59   LINE      1 DEPT  EMP#  NAME-LAST  NAME-FIRST
```

```
11/25/83       SELECTED DEPARTMENTS VIA VSAM PATH PROCESSING  PAGE      1

                              EMPLOYEE     LAST      FIRST
                      DEPT     NUMBER      NAME       NAME

                      911      00445     POST       JEAN
                               01730     SMOTH      CINDY
                               01963     ARNOLD     LINDA
                               03571     KRUSE      MAX
                               03890     STRIDE     ANN
                               04589     YOUNG      JANE
                               05805     REYNOLDS   WILLIAM
                               09764     HAFER      ARTHUR
                               11357     LARSON     RODNEY
                               11710     POWELL     CAROL
                               12641     ISAAC      RUTH
                               12829     GREEN      BRENDA
NO EMPLOYEES IN DEPARTMENT 328
                      942      06239     JOHNSON    LISA
                               12318     MALLOW     TERRY
                      914      01895     VETTER     DENISE
                               07231     GRECO      LESLIE
                               08262     CROCI      JUDY
                               10961     RYAN       PAMELA
                               11602     MANHART    VIRGINIA
```

Updating a VSAM ESDS File

This example updates the VSAM ESDS file. The file is read sequentially and each employee in Region 2 is given a 10 percent cost-of-living raise. VSAM provides the ability to update an ESDS record in place as long as the record length does not change. A STOP is issued when the region code is greater than 2 to avoid unnecessary processing.

```
 1 *
 2 *    EXAMPLE 15.10
 3 *
 4 FILE    PERESDS   VS(ES  UPDATE)
 5 %PERSNL
35 *
36   NEW-GROSS   W  4 P 2  HEADING('NEW' 'GROSS' 'SALARY')
37 *
38 JOB
39   IF  REGION > 2
40     STOP                    . * STOP IF PAST RECORDS OF INTEREST
42   END-IF
43   IF  REGION NE 2
44     GOTO JOB                . * IGNORE IF NOT DESIRED REGION
46   END-IF
47 *
48   NEW-GROSS = 1.1 * PAY-GROSS  . * CALCULATE NEW GROSS SALARY
50   PRINT  AUDIT-REPORT          . * PRINT AUDIT REPORT
52   PAY-GROSS = NEW-GROSS        . * UPDATE GROSS SALARY
54   WRITE   PERESDS  UPDATE      . * UPDATE PERSONNEL FILE
56 *
57 REPORT  AUDIT-REPORT           LINESIZE 80
58   SEQUENCE  BRANCH EMP#
59   TITLE     1 'AUDIT REPORT -- REGION 2 SALARY CHANGE ACTIVITY'
60   LINE      1 BRANCH EMP# NAME-LAST  NAME-FIRST PAY-GROSS NEW-GROSS
```

```
61 *
62 * VALIDATE PREVIOUS UPDATE
63 JOB
64   IF  REGION > 2
65     STOP                    . * STOP IF PAST RECORDS OF INTEREST
67   END-IF
68   IF  REGION NE 2
69     GOTO JOB                . * IGNORE IF NOT DESIRED REGION
71   END-IF
72 *
73   PRINT  VALIDATION            . * PRINT VALIDATION
75 *
76 REPORT  VALIDATION             LINESIZE 80
77   SEQUENCE  BRANCH EMP#
78   TITLE     1 'UPDATE VALIDATION -- REGION 2 SALARY CHANGE'
79   LINE      1 BRANCH EMP# NAME-LAST  NAME-FIRST PAY-GROSS


11/25/83      AUDIT REPORT -- REGION 2 SALARY CHANGE ACTIVITY    PAGE   1


                                                                    NEW
               EMPLOYEE      LAST       FIRST        GROSS        GROSS
      BRANCH    NUMBER       NAME        NAME         PAY         SALARY

        01       11376      HUSS        PATTI         360.80       396.88
        02       03571      KRUSE       MAX           242.40       266.64
        02       11710      POWELL      CAROL         243.20       267.52
        03       00577      PETRIK      KATHY         220.80       242.88
        03       02765      DENNING     RALPH         135.85       149.43
        03       03416      FORREST     BILL           13.80        15.18
        03       04234      MCMAHON     BARBARA       386.40       425.04
        04       00445      POST        JEAN          292.00       321.20
        05       01895      VETTER      DENISE        279.36       307.29
        05       04225      LOYAL       NED           295.20       324.72

------------------------------------------------------------------------------

11/25/83        UPDATE VALIDATION -- REGION 2 SALARY CHANGE   PAGE      1


                      EMPLOYEE      LAST       FIRST        GROSS
             BRANCH    NUMBER       NAME        NAME         PAY

               01       11376      HUSS        PATTI         396.88
               02       03571      KRUSE       MAX           266.64
               02       11710      POWELL      CAROL         267.52
               03       00577      PETRIK      KATHY         242.88
               03       02765      DENNING     RALPH         149.43
               03       03416      FORREST     BILL           15.18
               03       04234      MCMAHON     BARBARA       425.04
               04       00445      POST        JEAN          321.20
               05       01895      VETTER      DENISE        307.29
               05       04225      LOYAL       NED           324.72
```

Deleting and Adding Records of VSAM KSDS File

This example shows how to delete and add records to a VSAM KSDS. The employee numbers of three employees must be changed. Since the file is keyed on the employee number, each record must be deleted and then added again with the new employee number. The automatic input file contains the transaction records.

In the first job activity, each record to be deleted is read. If it is not present, an error message is issued. Otherwise, the employee number is changed, a copy is placed in a sequential work file for later addition, the original record is deleted, and an audit report is generated.

In the second job activity, the records written to the work file are read by automatic input. A READ is performed to make sure no existing record in the file has the new employee number; if so, an error message is issued. Otherwise, the employee record with the new employee number is added to the file. A READ is issued to validate the ADD; this is not done typically, but is done here to show that the ADD actually occurred. An audit report is output.

```
 1 *
 2 *   EXAMPLE 15.11
 3 *
 4 FILE    TRANS   F  80
 5   OLD-EMP#      1 5 N
 6   NEW-EMP#      7 5 N
 7 *
 8 FILE    PERKSDS   VS(PASSWORD 'UPDPER'  UPDATE)
 9 %PERSNL
39 *
40 FILE    PERSTMP   F 150   VIRTUAL   . * TEMP HOLD FILE FOR RECORDS
42   TMP-EMP#  9 5 N
43 *
44 JOB
45   READ  PERKSDS  KEY OLD-EMP#  STATUS
46   IF  PERKSDS:FILE-STATUS NE 0
47     DISPLAY 'INVALID EMPLOYEE NUMBER ' OLD-EMP#
48     GOTO JOB                . * TRY NEXT TRANSACTION
50   END-IF
51 *
52   TMP-EMP# = NEW-EMP#          . * SET NEW EMP# INTO RECORD
54   PUT    PERSTMP FROM PERKSDS  . * SAVE RECORD FOR SECOND JOB
56   WRITE  PERKSDS DELETE        . * DELETE THE OLD RECORD
58   PRINT  AUDIT-REPORT          . * PRINT THE AUDIT
60 *
61 REPORT  AUDIT-REPORT           LINESIZE 80
62   TITLE     1 'AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY'
63   TITLE     2 'FIRST PASS -- DELETED RECORDS'
64   LINE      1 OLD-EMP#  NAME-LAST  NAME-FIRST
65 *
66 *
67 JOB   INPUT  PERSTMP
68   READ   PERKSDS  KEY TMP-EMP#  STATUS  . * VALIDATE KEY
70   IF  PERKSDS:FILE-STATUS EQ 0
71     DISPLAY 'EMP# ALREADY ON FILE - ' TMP-EMP#
72     GOTO JOB
73   END-IF
74   WRITE  PERKSDS ADD FROM PERSTMP       . * WRITE NEW RECORD
76   READ   PERKSDS  KEY TMP-EMP#          . * VALIDATE WRITE (OPTIONAL)
78   PRINT  AUDIT-REPORT                   . * PRINT AUDIT REPORT
80 *
81 REPORT  AUDIT-REPORT           LINESIZE 80
82   TITLE     1 'AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY'
83   TITLE     2 'SECOND PASS -- ADD RECORDS'
84   LINE      1 TEMP-EMP#  NAME-LAST  NAME-FIRST
```

```
11/25/83      AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY    PAGE   1
                          FIRST PASS -- DELETED RECORDS


                                      LAST      FIRST
                           OLD-EMP#   NAME       NAME

                            12267     WIMN      GLORIA
INVALID EMPLOYEE NUMBER 11711
                            01963     ARNOLD    LINDA

------------------------------------------------------------------------

11/25/83      AUDIT REPORT -- EMPLOYEE NUMBER CHANGE ACTIVITY    PAGE   1
                           SECOND PASS -- ADD RECORDS


                          EMPLOYEE    LAST      FIRST
                           NUMBER     NAME       NAME

                           99267      WIMN      GLORIA
                           81963      ARNOLD    LINDA
```

GETDATE Macro

A macro facility is provided to ease the burden of repetitive coding. Typically, a macro might contain file and field definitions and be used as a data dictionary. This technique is demonstrated in almost every example in this guide. Another use of the macro facility is to store functions, such as field reformatting, and report definitions, as illustrated in Examples 15.12A and 15.12B (shown below).

Example 15.12A is a macro to obtain the system date and strip out the slashes. It was also used in Example 14.14 (as shown in the "Basic Examples" chapter). The macro is defined with a single positional parameter -- the receiving field. All other required fields are defined within the macro.

```
*   EXAMPLE 15.12A

MACRO USER-DATE
*
*  GET THE CURRENT DATE AND PUT INTO USER FIELD LESS SLASHES
*
  DEFINE GETDATE-DATE           W        8  A
  DEFINE GETDATE-FIRST6 GETDATE-DATE     6  N
  DEFINE GETDATE-LAST5  GETDATE-DATE +3  5  A
  DEFINE GETDATE-LAST6  GETDATE-DATE +2  6  A
  DEFINE GETDATE-LAST3  GETDATE-DATE +5  3  A
  DEFINE GETDATE-LAST2  GETDATE-DATE +6  2  A
  GETDATE-DATE  = SYSDATE             . * MOVE ALL 8
  GETDATE-LAST3 = GETDATE-LAST2       . * SHIFT LEFT OVER NEXT /
  GETDATE-LAST6 = GETDATE-LAST5       . * SHIFT LEFT OVER FIRST /
  &USER-DATE = GETDATE-FIRST6         . * MOVE TO USER FIELD
```

Example 15.12B shows how the macro is invoked. A LIST NOMACROS statement is in the job to inhibit expanding the generated macro code.

```
 1 *
 2 *  EXAMPLE 15.12B
 3 *
 4 *
 5 * THIS EXAMPLE WILL GET THE SYSTEM DATE AND STRIP THE SLASHES FROM IT
 6 * THE FORMAT IS
 7 *
 8 *        %GETDATE  SYSTEM-DATE
 9 *
10 *         WHERE SYSTEM-DATE IS THE FIELD TO PUT THE CONVERTED DATE
11 *
12 JOB INPUT NULL
13 DEFINE SYSTEM-DATE   W      4      P
14   %GETDATE SYSTEM-DATE
32   DISPLAY NEWPAGE SYSDATE ' WAS CONVERTED TO ' SYSTEM-DATE
33   STOP

------------------------------------------------------------------------

11/24/83 WAS CONVERTED TO  112483
```

CONCAT Macro

This is an example of a macro that concatenates two fields into one, with variable spacing between fields. Following is the macro definition:

```
*    EXAMPLE 15.13A
*
MACRO RECEIVE SPACE PART2
*
* CONCAT MACRO
* FORMAT:
*        %CONCAT RECEIVE N PART2
*
*   IT PERFORMS AS:
*           RECEIVE = RECEIVE +SPC(N) + PART2
*
DEFINE CONCAT-HOLD W 254 A
DEFINE CONCAT-SCAN CONCAT-HOLD    1    A  INDEX CONCAT-NDX
DEFINE CONCAT-LENGTH    W     2   P 0
CONCAT-NDX   =   253                              . * START AT END
CONCAT-HOLD  = &RECEIVE                           . * COPY PART1
DO WHILE CONCAT-SCAN EQ ' '  AND CONCAT-NDX GE 0
  CONCAT-NDX = CONCAT-NDX  -  1                   . * FIND 1ST #BLANK
END-DO
CONCAT-NDX = CONCAT-NDX + 1 + &SPACE              . * DO SPACES FACTOR
CONCAT-LENGTH  = 253 - CONCAT-NDX
MOVE &PART2  TO CONCAT-SCAN CONCAT-LENGTH . * MOVE PART 2 AFTER
&RECEIVE = CONCAT-HOLD   . * GIVE BACK TO USER
```

The CONCAT macro is exercised by the following job:

```
 1 *
 2 *  EXAMPLE 15.13B
 3 *
 4 *
 5 * THIS EXAMPLE WILL CONCATENATE TWO FIELDS TOGETHER
 6 * THE FORMAT OF THE MACRO IS:
 7 *
 8 *        %CONCAT PART1 N PART2
 9 *
10 *         WHERE N IS THE NUMBER OF SPACES TO BE INSERTED BETWEEN
11 *         THE TWO PARTS
12 *
13 *         THE FIRST PARAMETER (PART1) IS USED AS THE RECEIVING FIELD
14 *         THE THIRD PARAMETER (PART2) MAY BE A LITERAL BUT
15 *         ENSURE THAT THE PROPER SYNTAX RULES ARE FOLLOWED FOR PASSING
16 *         QUOTES (IE TO PASS A COMMA, ''','''' WOULD BE USED)
17 *
18 JOB INPUT NULL
19 DEFINE WHOLE-THING   W      40     A
20 DEFINE SECOND-PART   W      40     A
21 DEFINE PART1         W      10     A    VALUE 'SEE HOW'
22 DEFINE PART2         W      10     A    VALUE 'IT PUTS'
23 DEFINE PART3         W      10     A    VALUE 'THE PARTS'
24 DEFINE PART4         W      10     A    VALUE 'TOGETH'
25 DEFINE PART5         W      10     A    VALUE 'ER'
26   WHOLE-THING = PART1
27   SECOND-PART = PART2
28   DISPLAY NEWPAGE WHOLE-THING
29   PERFORM CONCAT-SPACE
30   DISPLAY WHOLE-THING
31   SECOND-PART = PART3
32   PERFORM CONCAT-SPACE
33   DISPLAY WHOLE-THING
34   SECOND-PART = PART4
35   PERFORM CONCAT-SPACE
36   DISPLAY WHOLE-THING
37   SECOND-PART = PART5
38   PERFORM CONCAT-NOSPACE
39   DISPLAY WHOLE-THING
40   STOP
41 CONCAT-SPACE. PROC
43 *
44   %CONCAT WHOLE-THING 1 SECOND-PART
71 *
72 END-PROC
73 *
74 CONCAT-NOSPACE. PROC
76 *
77   %CONCAT WHOLE-THING 0 SECOND-PART
104 *
105 END-PROC

--------------------------------------------------------------------------------
SEE HOW
SEE HOW IT PUTS
SEE HOW IT PUTS THE PARTS
SEE HOW IT PUTS THE PARTS TOGETH
SEE HOW IT PUTS THE PARTS TOGETHER
```

Processing JCL Parameters

This example obtains a parameter coded on a JCL EXEC statement and uses it to control the selection of records.

A START procedure receives control at the beginning of the job. This procedure calls a subprogram that moves the PARM information into a CA-Easytrieve Plus field. A JCL parameter can be from 1 to 100 characters long. The maximum length of the field you defined (PARM-DATA in this case) must be specified in PARM-LTH. The subprogram, EZTPX01, moves up to 10 characters from the JCL PARM-DATA, in this example.

If no JCL parameter is specified on the EXEC statement, PARM-LTH includes a zero after calling EZTPX01. If more than 10 characters are specified, the PARM is truncated to 10 characters.

In this example, the expected JCL parameter is defined to be three digits; the first digit specifies the region number and the next two contain the branch number. The format and content of the parameter is validated after calling EZTPX01. A PARM-LTH of 10 was used to detect either a missing JCL parameter or one that is not the correct length. If PARM-LTH = 3 was coded, a parameter longer than 3 could not be detected, since EZTPX01 would truncate it to three characters.

Note: The use of field redefinition defines subfields of the JCL parameter. The value of the JCL parameter is 302 for this example.

The EZTPX01 routine is distributed as part of the product.

```
 1 *
 2 *    EXAMPLE 15.14
 3 *
 4 FILE  PERSNL   FB(150 1800)
 5 %PERSNL
35 *
36 *   JCL PARM DECODE DEFINITIONS
37 *
38   PARM-INFO      W             12 A
39   PARM-LTH       PARM-INFO      2 B
40   PARM-DATA      PARM-INFO +2  10 A
41 *
42   SELECT-REGION  PARM-DATA      1 N
43   SELECT-BRANCH  PARM-DATA +1   2 N
44 *
45 *
46 JOB   START(PARM-ANALYSIS)
47   IF  REGION = SELECT-REGION  AND  BRANCH = SELECT-BRANCH
48     PRINT
49   END-IF
50 *
51 *   MOVE THE JCL PARM INTO THE PARM-INFO W-FIELD AND VALIDATE IT
52 *
53 *   THE JCL PARM MUST BE 3 DIGITS OF THE FORM  RBB
54 *   WHERE R IS THE REGION NUMBER AND BB IS THE BRANCH NUMBER
55 *
56   PARM-ANALYSIS. PROC
58     PARM-LTH = 10
59     CALL  EZTPX01  USING (PARM-REGISTER  PARM-INFO)
```

```
60     IF PARM-LTH NE 3
61       DISPLAY SKIP 3 '******* MISSING OR INVALID JCL PARM'
62       STOP EXECUTE
63     END-IF
64     IF SELECT-REGION NE 1 THRU 4
65       DISPLAY SKIP 3 '******* INVALID REGION NUMBER'
66       STOP EXECUTE
67     END-IF
68     IF SELECT-BRANCH NE 1 THRU 5
69       DISPLAY SKIP 3 '******* INVALID BRANCH NUMBER'
70       STOP EXECUTE
71     END-IF
72   END-PROC
73 *
74 REPORT    LINESIZE 70
75   TITLE     'EMPLOYEES IN REGION' -2 SELECT-REGION      -
               'BRANCH' -2 SELECT-BRANCH
76   LINE      NAME-FIRST  NAME-LAST  EMP#


 3/22/84           EMPLOYEES IN REGION 3   BRANCH 02      PAGE      1

                     FIRST       LAST     EMPLOYEE
                      NAME       NAME      NUMBER

                    JANICE     THOMPSON    01743
                    CINDY      SMOTH       01730
                    RUTH       ISAAC       12641
                    LORRIE     LACH        09609
                    LESLIE     GRECO       07231
                    WILLIAM    REYNOLDS    05805
```

Chapter

16

Bank System

The BANK system is a combination of online and batch processing that illustrates the adaptability of CA-Easytrieve Plus to a wide range of environments, and demonstrates a variety of coding techniques. This system is a sample application for a BANK, but could be adapted to other applications with minor modifications. However, implementation of a system as sophisticated as this requires considerable knowledge of generalized program development and substantial experience in data processing. If your professional expertise lies primarily in other fields, you do need help from your data center to undertake this task.

A common need is to have ready access to data; that is, the ability to quickly locate and modify specific items. This requires the means to search a file for a certain record or set of records and, once located, to update the record. This ready access is typically called online processing. Online processing offers speed and flexibility advantages over batch processing. When you are working online you can access any record, look at it on a terminal, and perform any function based on the values in the record at that time. A batch job requires you to specify in advance not only which records to access, but what functions to perform on them.

Online access is most appropriate for accessing a small number of records and performing varied activities, such as deleting one record, changing the name on another, and adding a middle initial to a third. A batch job is more efficient to list all records from a file, or to read all records from a file and to choose certain individuals to receive a letter.

The BANK system encompasses these two types:

- Online processing to access specific records
- Batch processing to produce the reports.

Online Processing

The online portion of this system runs under TSO, CMS, or ICCF. The discussion in this chapter assumes TSO. The data is stored on a key-sequenced VSAM data set (KSDS) created by running the IDCAMS utility with the following commands:

```
DEFINE CLUSTER (NAME ('your.bank.masterfile') -
       KEYS (6 0)  -
       REUSE -
       VOLUMES (volser) -
       RECORDSIZE (200 200) -
       RECORDS (50 50) )
```

This group of statements defines the file to the system and must be followed by an initialization job to prepare the file for processing, as illustrated in Example 16.1 below.

Initialize Customer File

```
 1 *
 2 * EXAMPLE 16.1
 3 *
 4 %BANKLIB
45 *
46   NEXT-ID    7    6     N
47 *
48 JOB INPUT NULL
49 *
50 *  THIS JOB INITIALIZES THE CUSTOMER FILE
51 *  IT WRITES THE FIRST RECORD ON THE FILE
52 *  WITH THE NEXT AVAILABLE CUST-ID OF 1
53 *
54   CUST-ID  = 0    . * SET RECORD 0
56   NEXT-ID  = 1    . * SET NEXT RECORD TO 1
58   PUT    CUST     . * ADD THE RECORD TO THE FILE
60   STOP            . * ALL DONE
```

As previously mentioned, the master file used is a key-sequenced data set whose key is a derived field - the record number. Therefore, the first record added to the file has a key of 000001, the second 000002, and so forth.

The record number (key) of the next available record on the file is stored in field 'NEXT-ID' of record zero. This field is set to a value of 000001 by the initialization job. As data is added to the file by subsequent processing, this field is updated. The key field is the first six bytes of each record. The other fields are:

```
PERS-TITLE     (i.e. MR, MRS, MISS, MS)
FIRST-NAME
LAST-NAME
MIDDLE-INITIAL
ADDRESS LINE 1
ADDRESS LINE 2
CITY
STATE
ZIP
LOCAL INDICATOR   (YES,NO)
CREDIT RATING     (0-9)
SAVINGS ACCOUNT   (YES,NO)
CHECKING          (YES,NO)
SAFE DEPOSIT      (YES,NO)
C AND D           (YES,NO)
ALL SAVERS        (YES,NO)
VISA              (YES,NO)
MASTER CARD       (YES,NO)
MONEY MARKET      (YES,NO)
TREASURY BILL     (YES,NO)
```

Once the IDCAMS and initialization jobs are run, you can begin adding data to the file.

The first step is to log onto TSO and create the Job Control Setup command list (CLIST), as follows:

```
*   BANK CLIST FOR TSO
FREE F(SYSIN SYSPRINT PANDD1 CUST TERMIN EZTVFM)
ALLOC F(PANDD1) DA ('your macro library') SHR
ALLOC F(SYSPRINT) DA(*)
ALLOC F(TERMIN) DA(*)
ALLOC F(CUST) DA('your bank masterfile') SHR
ALLOC F(SYSIN) DA(*)
ALLOC F(EZTVFM) SP(2,2) CYL
CALL 'your program library(EZTPA00)'
```

The next step is to type:

```
EXEC BANK
```

The system introduces itself and prepares to accept commands. The valid commands and their descriptions are:

ADD

This command is used to add a new record to a file. After you enter the ADD keyword, the system displays the attributes (number, name, length, and type) of each field of the next available record, in order by field number, and asks you to enter a value for that field.

The field type codes are:

A = alphabetic - only letters allowed
N = numeric - only numbers allowed
X = mixed - any characters allowed
Q = (yes,no) - only Y or N allowed

When you have entered data into the last field of the new record, the system displays its record number (key) as a customer ID.

BROWSE

This command enables you to specify desired values in up to 20 fields of a record. After these field values are specified, the system reads the entire file and displays those records, one at a time, with fields to match your specification.

To search for a field, enter the field number. The system displays the attributes of that field and asks you to enter a value. When you finish entering the search values for one record, you can enter NEXT to begin entering search values for the next record, or END to terminate the search and begin the BROWSE activity.

Only those records that have all fields equal to the search data can be displayed. Repeat END to terminate BROWSE. The last record read is the current record that can be used by DISPLAY, DEL, or UPD.

DEL

This command deletes a record. After you enter the DEL keyword, the system displays the ID of the current record and asks if this is the one you want.

- If you enter YES, that record is deleted.
- If you enter NO, the system requests the ID of the desired record, and deletes the record you specify.

DISPLAY

This command displays the fields of the current record or any other record you specify. After you enter the DISPLAY keyword, the system displays the ID of the current record and asks if this is the one you want.

- If you answer YES, the fields of that record are displayed.
- If you answer NO, the system requests the ID of the desired record and displays the fields of the record you specify.

ECHO

This command causes all terminal input to be displayed on the terminal.

END

This command terminates the current activity; when used on the primary command line it terminates the session. When in BROWSE or UPDate processing, this command terminates the field value specification phase and begins the BROWSE or UPDate activity.

ESC

This command is valid at anytime and returns you to the major command entry of the session. It is useful when you are in the middle of processing a command that you do not want to continue.

NOECHO

This command inhibits the display of terminal input.

UPD

This command enables you to enter UPDate mode. If a record is active from a previous activity, its ID is displayed and the system asks if this is the one you want. If you enter NO, the system requests the ID of the desired record. Once that record is found, the system requests the field number of the field to be updated. That field's attributes are displayed and the system requests new data.

This process of specifying a field number and entering new data continues until you enter END. The system then inquires whether you are ready to update.

- If you answer NO, you return to specifying field numbers and entering new data.
- If you answer YES, the record is updated.

The CA-Easytrieve Plus coding required for the BANK system and the sample terminal session using this system are illustrated in Example 16.2, as shown in the next two exhibits. The first page of this example provides a listing of the macro %BANKLIB, which is used throughout the BANK system.

BANKLIB Macro

```
MACRO 0 VSAMOPT 'UPDATE'
*
*        COMMON TABLE FILE TO DEFINE ACCOUNT TYPES
*
FILE ACCTNAME   TABLE   INSTREAM
  ARG   1    2    N
  DESC  4    30   A
01 SAVINGS
02 CHECKING
03 SAFE DEPOSIT
04 CERTIFICATE OF DEPOSIT
05 ALL SAVERS
06 VISA
07 MASTER CARD
08 MONEY MARKET
09 TREASURY BILL
ENDTABLE
*
*        BANK CUSTOMER FILE DESCRIPTION
*
FILE CUST      VS(F &VSAMOPT)  WORKAREA 200
*
  CUST-ID              1       6        N
  NAME                 7       25       A
    PERS-TITLE         7       4        A
    FIRST-NAME         11      10       A
    LAST-NAME          21      15       A
    MIDDLE-INITIAL     36      1        A
  ADDRESS1             38      25       A
  ADDRESS2             63      25       A
  CITY                 88      15       A
  STATE                103     2        A
  ZIP                  105     9        A
  LOCAL                114     1        A
  CREDIT-RATING        115     1        A      -
        HEADING ('CREDIT' 'RATING')
  ACCT-DATA            116     9        A
    SAVINGS            116     1        N 0
    CHECKING           117     1        N 0
    SAFE-DEPOSIT       118     1        N 0    -
         HEADING ('SAFE' 'DEPOSIT')
    C-AND-D            119     1        N 0
    ALL-SAVERS         120     1        N 0    -
         HEADING ('ALL' 'SAVERS')
    VISA               121     1        N 0
    MASTER-CARD        122     1        N 0    -
         HEADING ('MASTER' 'CARD')
    MONEY-MARKET       123     1        N 0    -
         HEADING ('MONEY' 'MARKET')
    T-BILL             124     1        N 0
  ACCT-IND      ACCT-DATA      1        N 0   -
                  OCCURS   9     INDEX  ACCT-NDX
*
  ACCT-MAX      W       2      P   0  VALUE 9
  CUST-KEY      W       6      N
*
```

Bank File Program

```
 1 *
 2 * BANK EXAMPLE 16.2 BANK FILE UPDATE TERMINAL SESSION
 3 *
 4 %BANKLIB
45   CUST-AREA   1    200    A
46   CUST-FIELD  1    1      A    INDEX CUST-NDX
47   NEXT-ID     7    6      N
48 FILE TERMIN   WORKAREA 80
49   TERM-REC           1    80     A   -
             INDEX TERM-BEGIN-NDX           . * INPUT TERMINAL RECORD
51   TERM-BEGIN-CH      1    1      A   -
                     INDEX TERM-BEGIN-NDX    . * LEADING SPACE SCAN CHAR
53   TERM-END-CH        1    1      A   -
                     INDEX TERM-END-NDX      . * TRAILING SPACE SCAN CHAR
55   FIRST3   TERM-REC       3      A
56   CMD-IN   TERM-REC       10     A
57   KEY-IN   TERM-REC       6      N       . * INPUT CUST-ID
59   TERM-FIELD-NO TERM-REC  3      N       . * INPUT FIELD NUMBER
61   TERM-LEN    W    2      P               . * LENGTH OF TERMINAL INPUT
63   NEXT-KEY    W    6      N               . * NEXT AVAILABLE KEY
65 *
66 * SWITCHS USED DURING PROCESSING
67 *
68   FIELD-ERR   W    1      A               . * NOT DONE WITH FIELD YET
70   ERROR       W    1      A
71   CUST-ERR    W    1      A
72   ECHO-SW     W    1      A  VALUE 'Y'    . * INITIAL VALUE FOR ECHO
74   NUM-SW      W    1      A               . * INPUT HAS NUMBERS
76   ALPHA-SW    W    1      A               . * INPUT HAS ALPHAS
78   OTHER-SW    W    1      A               . * INPUT HAS NON(ALPHANUMERIC)
80   ANSWER      W    1      A               . * RESPONSE TO YES/NO
82 *
83 * COMMAND TABLES/INFORMATION
84 *
85   CUR-STATE   W    1      N                 . * CURRENT STATUS
87   CMD-ID      W    1      N
88   CMD-FND     W    1      A                 . * CMD SWITCH
90   CMD-OVHD    W    2      P  VALUE 3        . * AMT TO ADD TO CMD-LEN
92   CMD-DEFAULT W    1      N
93   CMD-HOLD    W    10     A
94 *
95 * THE FOLLOWING TABLE OF COMMANDS CONTAIN:
96 *
97 *       COMMAND-ID           1 BYTE
98 *       COMMAND-STATE        1 BYTE
99 *       COMMAND-LENGTH       1 BYTE
100 *      COMMAND-NAME         ? BYTES
101 *
102 *      THIS TABLE IS LOOPED THRU TO VALIDATE AND DETERMINE WHAT THE
103 *      USER REQUEST IS
104 *      A NEW COMMAND CAN BE ADDED BEFORE THE LAST COMMAND WHICH
105 *      MUST HAVE A CMD-LEN OF 0
106 *      MAINLINE CODE PERFORMS THE ROUTINE DEPENDING ON COMMAND-ID
107 *
```

```
108 CMD-TBL       W   100  A  VALUE   -
       '003END+
        107DISPLAY+
        203ADD+
        303UPD+
        403DEL+
        504ECHO+
        606NOECHO+
        706BROWSE+
        073END+
        174NEXT+
        000'                                        . * END OF COMMAND TABLE
110   CMD-DATA    CMD-TBL      20    A   -
             INDEX CMD-NDX
111     CMD-NO    CMD-DATA     1     N
112     CMD-STATE CMD-DATA  +1 1     N
113     CMD-LEN   CMD-DATA  +2 1     N
114     CMD-NAME  CMD-DATA  +3 1     A
115 *
116 * THIS AREA IS USED TO MOVE THE CORRECT NUMBER OF BYTES FOR THE
117 * VALID COMMANDS TO PROMPT THE USER
118 *
119   DSP-LINE    W     100    A
120    DSP-LIT DSP-LINE 22     A   VALUE 'VALID COMMANDS ARE:ESC'
121    DSP-DATA  DSP-LINE +22 78   A
122    DSP-COMMA DSP-DATA      1  A   INDEX DSP-NDX
123    DSP-CMD   DSP-DATA +2   10 A   INDEX DSP-NDX
124 *
125 * THE NEXT AREA IS THE OBJECT OF THE FIELD TABLE LOOKUP
126 *
127  FIELD-DEFN      W     72    A
128  FIELD-ID     FIELD-DEFN     3     N
129  FIELD-DATA   FIELD-DEFN   +4  68    A
130    FIELD-NAME   FIELD-DATA     20    A
131    FIELD-LOC    FIELD-DATA +20 3     N
132    FIELD-LEN    FIELD-DATA +24 2     N
133    FIELD-TYPE   FIELD-DATA +27 1     A
134  FIELD-MAX       W     3     P     VALUE 20  . * MAX NUMBER OF FIELDS
136  FIXED-FIELD     W     40    A               . * DISPLAY AREA FOR FIELD
138  COMPARE-FIELD   W     40    A               . * HOLD AREA FOR FIELD
140  QUERY-MAX       W     2     P     VALUE 20  . * 20 QUERYS
142  QUERY-INC       W     2     P     VALUE 43  . * AMT TO INC
144  QUERY-CNT       W     2     P               . * COUNTER FOR LOOPING
146  QUERY-TABLE     W     43    A    -
               OCCURS 21 INDEX QUERY-NDX       . * TABLE OF USER QUERIES
148                                              * PLUS ONE DUMMY
149    QUERY-FIELD    QUERY-TABLE       3     N  . * FIELD NO FOR QUERY
151    COMPARE-QUERY  QUERY-TABLE +3    40    A  . * QUERY
153 *
154 * THE NEXT TABLE CONTAINS ALL VALID FIELDS TO BE ENTERED
155 * IT ALSO CONTAINS THE BEGINNING, LENGTH, TYPE, AND EDIT RULES
156 *
157 FILE FIELDTBL TABLE  INSTREAM
158 ARG     1    3    N
159 DESC    4    68   A
160 001 TITLE              007 04 A
    002 FIRST NAME         011 10 A
    003 LAST NAME          021 15 A
    004 MIDDLE INITIAL     036 01 A
    005 ADDRESS LINE 1     038 25 X
    006 ADDRESS LINE 2     063 25 X
    007 CITY               088 15 A
    008 STATE              103 02 A
    009 ZIP                105 09 X
    010 IS IT LOCAL        114 01 Q
    011 CREDIT RATING      115 01 N
```

```
    012 SAVINGS            116 01 Q
    013 CHECKING           117 01 Q
    014 SAFE DEPOSIT BOX   118 01 Q
    015 C AND D            119 01 Q
    016 ALL SAVERS         120 01 Q
    017 VISA               121 01 Q
    018 MASTER CARD        122 01 Q
    019 MONEY MARKET       123 01 Q
    020 T BILL             124 01 Q
    ENDTABLE
161 JOB INPUT NULL START HELLO  FINISH END-OF-JOB
162 *
163 * THE MAINLINE LOGIC FOLLOWS:
164 *
165 *        A COMMAND IS OBTAINED
166 *        THE COMMAND ID CORRESPONDING TO THE COMMAND IS RETURNED
167 *        THE ROUTINE TO HANDLE THE REQUEST IS PERFORMED
168 *
169   CUR-STATE  = 0                          . * SET TO FIRST LEVEL
171   CMD-DEFAULT = 0                         . * NO DEFAULT HERE
173   PERFORM GET-COMMAND
174   IF CMD-ID  =  0                         . * END
176     STOP
177   END-IF
178   IF CMD-ID  =  1                         . * DISPLAY
180     PERFORM  SELECT-RECORD                . * POSITION
182     PERFORM  DISPLAY-RECORD
183     GOTO JOB
184   END-IF
185   IF CMD-ID  =  2                         . * ADD
187     PERFORM ADD-NEW-RECORD
188     GOTO JOB
189   END-IF
190   IF CMD-ID  =  3                         . * UPD
192     PERFORM SELECT-RECORD                 . * POSITION
194     PERFORM UPDATE-RECORD                 . * GET AND ACCEPT UPDATES
196     GOTO JOB
197   END-IF
198   IF CMD-ID  =  4                         . * DELETE
200     PERFORM SELECT-RECORD                 . * POSITION
202     PERFORM DELETE-RECORD                 . * DELETE
204     GOTO JOB
205   END-IF
206   IF CMD-ID  =  5                         . * ECHO
208     ECHO-SW  =   'Y'                      . * SET ECHOSW
210     GOTO JOB
211   END-IF
212   IF CMD-ID  =  6                         . * NOECHO
214     ECHO-SW  =   'N'                      . * SET OFF
216     GOTO JOB
217   END-IF
218   IF CMD-ID = 7                           . * BROWSE
220     PERFORM BROWSE-FILE
221     GOTO JOB
222   END-IF
223 *
224 ADD-NEW-RECORD. PROC
226 *
227 * ADD A NEW CUSTOMER
228 * FOR EACH FIELD ON THE FILE GET THE VALUE FROM THE USER
229 * WHEN FINISHED, ADD TO THE FILE AND RETURN
230 *
231   CUST-KEY =  0
232   PERFORM READ-CUST
233   CUST-AREA   =    ' '
234   FIELD-ID    =    1
```

```
235   DO WHILE FIELD-ID LE FIELD-MAX
236     PERFORM GET-FIELD
237     FIELD-ID  = FIELD-ID  + 1
238   END-DO
239   CUST-ID  = NEXT-KEY
240   CUST-KEY = NEXT-KEY                    . * SAVE FOR CURENT RECORD TEST
242   WRITE CUST ADD STATUS
243   PERFORM CUST-FILE-TEST
244   DISPLAY 'CUSTOMER ' CUST-ID ' ADDED'
245   NEXT-KEY = NEXT-KEY + 1
246 END-PROC
247 *
248 BROWSE-FILE. PROC
250 *
251 * THIS ROUTINE WILL ASK FOR THE FIELDS BE THE SEARCH FIELDS
252 * AND ASK FOR THE VALUES DESIRED
253 * ALL RECORDS WITH THE DESIRED VALUE IN THE SELECTED FIELDS
254 * WILL BE LISTED UNTIL END OF FILE OR 'END' REQUESTED
255 *
256   CUST-KEY = 0                           . * RESET CURRENT RECORD
258   PERFORM READ-CUST                      . * HAVE TO GET IT
260   DISPLAY 'YOU CAN ENTER UP TO ' -
         QUERY-MAX ' QUERIES FOR THE SEARCH'
261   DISPLAY 'THE QUERIES WILL BE +
          ''ANDED'' TOGETHER'
262   QUERY-NDX = QUERY-MAX * QUERY-INC      . * POINT T0 AFTER LAST
264   QUERY-FIELD = 999                      . * SET TO 999
266   QUERY-CNT = 1                          . * BACK TO BEGINNING
268   QUERY-NDX = 0                          . * BACK TO BEGINNING
270   DO WHILE QUERY-CNT LE QUERY-MAX
271     DISPLAY 'ENTER FIELD NUMBER FOR +
                SEARCH ''END'' WHEN DONE'
272     PERFORM GET-FIELD-NUMBER
273     IF FIRST3 = 'END'
274       QUERY-FIELD = 0
275       COMPARE-QUERY = ' '
276       GOTO QUERY-DONE
277     ELSE
278       QUERY-FIELD = TERM-FIELD-NO
279       FIELD-ID = QUERY-FIELD
280       PERFORM GET-FIELD
281       MOVE TERM-REC TERM-LEN TO COMPARE-QUERY
282       QUERY-NDX = QUERY-NDX + QUERY-INC
283       QUERY-CNT = QUERY-CNT + 1
284     END-IF
285   END-DO
286   IF QUERY-CNT GT QUERY-MAX
287     DISPLAY 'MAXIMUM OF ' -
             QUERY-MAX ' QUERIES REACHED'
288   END-IF
289 QUERY-DONE
290   CUST-KEY = 1
291   POINT CUST GE  CUST-KEY   STATUS
292   PERFORM CUST-FILE-TEST
293   PERFORM NEXT-RECORD
294   CUR-STATE = CMD-ID                       . * SUBCOMMANDS OF BROWSE
296   CMD-DEFAULT  =  1                        . * NEXT IS DEFAULT
298   DO WHILE CUST                            . * READ ALL RECORDS
300     QUERY-NDX = 0                          . * SCAN ENTIRE TABLE
302     QUERY-CNT = 1                          . * SCAN ENTIRE TABLE
304     DO WHILE QUERY-CNT LE QUERY-MAX -
          AND QUERY-FIELD GT 0                 . * DO UNTIL END
306       SEARCH FIELDTBL WITH QUERY-FIELD -
            GIVING FIELD-DATA                  . * GET ATTRIBUTES
308       CUST-NDX = FIELD-LOC - 1
309       MOVE CUST-FIELD FIELD-LEN  TO COMPARE-FIELD
```

```
310       IF COMPARE-QUERY  EQ  COMPARE-FIELD  -
            OR  COMPARE-QUERY  SPACE
311         QUERY-NDX = QUERY-NDX + QUERY-INC
312         QUERY-CNT = QUERY-CNT + 1
313       ELSE
314         GOTO NOT-WANTED
315       END-IF
316     END-DO
317     DISPLAY 'CUST-ID:' CUST-ID
318     DISPLAY ' '
319     PERFORM DISPLAY-RECORD
320     PERFORM GET-COMMAND
321   NOT-WANTED
322     IF CMD-ID NE 0
323       PERFORM NEXT-RECORD
324     ELSE
325       GOTO BROWSE-END
326     END-IF
327   END-DO
328   BROWSE-END
329     DISPLAY 'BROWSE COMPLETE'
330 END-PROC
331 *
332 CONVERT-TO-YES-NO. PROC
334 *
335   IF FIELD-TYPE NE 'Q'
336     GOTO CONVERT-DONE
337   END-IF
338   IF CUST-FIELD EQ '0'
339     CUST-FIELD EQ 'N'
340   ELSE
341     CUST-FIELD EQ 'Y'
342   END-IF
343 CONVERT-DONE
344 END-PROC
345 *
346 CUST-FILE-TEST. PROC
348 *
349 * GENERAL MASTER FILE I/O TEST
350 *
351   IF CUST:FILE-STATUS NE 0
352      DISPLAY '*******FILE ERROR:CUST-----STATUS=' CUST:FILE-STATUS
353      STOP
354   END-IF
355 END-PROC
356 *
357 DELETE-RECORD. PROC
359 *
360 * THIS ROUTINE WILL DELETE THE CURRENT RECORD
361 * THE FILE MUST HAVE BEEN PREVIOUSLY POSITIONED
362 * AT THE DESIRED RECORD
363 *
364   WRITE CUST DELETE STATUS
365   PERFORM CUST-FILE-TEST
366   CUST-KEY = 0
367   DISPLAY 'RECORD ' CUST-ID ' DELETED'
368 END-PROC
369 *
370 DISPLAY-FIELD. PROC
372 *
373 * DISPLAY A FIELD IN THE CURRENT RECORD
374 * THE FIELD NUMBER WILL BE THAT OF 'FIELD-ID'
375 *
376   FIELD-ERR  = 'N'
377   SEARCH FIELDTBL WITH FIELD-ID GIVING FIELD-DATA
378   IF NOT FIELDTBL
```

```
379     FIELD-ERR = 'Y'
380     DISPLAY FIELD-ID ' NOT A VALID FIELD'
381   ELSE
382     CUST-NDX  =  FIELD-LOC  - 1
383     PERFORM CONVERT-TO-YES-NO
384     MOVE CUST-FIELD FIELD-LEN TO FIXED-FIELD
385     DISPLAY 'FIELD:' FIELD-ID +1 FIELD-NAME +1 -
          FIXED-FIELD
386   END-IF
387 END-PROC
388 *
389 DISPLAY-RECORD. PROC
391 *
392 * DISPLAY ALL FIELDS OF THE CURRENT RECORD
393 *
394   FIELD-ID = 1
395   DO WHILE FIELD-ID LE FIELD-MAX
396     PERFORM DISPLAY-FIELD
397     FIELD-ID = FIELD-ID + 1
398   END-DO
399 END-PROC
400 *
401 EDIT-ALPHA. PROC
403 *
404 * DISPLAY ERROR MSG IF NOT ALPHA
405 *
406   IF NUM-SW  =   'Y' OR OTHER-SW  = 'Y'
407     FIELD-ERR  =  'Y'
408     DISPLAY 'FIELD MUST CONTAIN ONLY LETTERS '
409   END-IF
410 END-PROC
411 *
412 EDIT-NUM. PROC
414 *
415 * DISPLAY ERROR MSG IF NOT NUMERIC
416 *
417   IF ALPHA-SW  =   'Y' OR OTHER-SW  = 'Y'
418     FIELD-ERR  =  'Y'
419     DISPLAY 'FIELD MUST CONTAIN ONLY NUMBERS '
420   END-IF
421 END-PROC
422 *
423 EDIT-YES-NO. PROC
425 *
426   IF  TERM-BEGIN-CH EQ 'N' ' '
427     TERM-BEGIN-CH = '0'       . * CONVERT TO 0
429   END-IF
430   IF TERM-BEGIN-CH EQ 'Y'
431     TERM-BEGIN-CH = '1'       . * CONVERT TO 1
433   END-IF
434   IF TERM-BEGIN-CH NE '0' '1'
435     DISPLAY 'FIELD MUST BE ''Y'' OR ''N'' '
436     FIELD-ERR  =  'Y'
437   END-IF
438 END-PROC
439 *
440 END-OF-JOB. PROC
442 *
443 * ALL DONE, READ RECORD 0 TO GET PREVIOUS NEXT KEY
444 * IF CURRENT NEXT KEY IS GT PREVIOUS THEN UPDATE RECORD 0
445 * INDICATE END OF SESSION
446   CUST-KEY   =    0
447   READ CUST KEY  CUST-KEY
448   PERFORM CUST-FILE-TEST
449   IF NEXT-ID NE NEXT-KEY
450     NEXT-ID = NEXT-KEY
```

```
451     WRITE CUST UPDATE
452     PERFORM CUST-FILE-TEST
453   END-IF
454   DISPLAY 'DATA ENTRY SESSION COMPLETE'
455 END-PROC
456 *
457 FIELD-EDIT-MOVE. PROC
459 *
460 * EDIT THE INPUT DATA, IF VALID MOVE TO OUTPUT
461 * ELSE RETURN WITH FIELD-ERR='Y'
462 *
463   FIELD-ERR = 'N'
464   IF FIELD-LEN  LT TERM-LEN
465     FIELD-ERR = 'Y'
466     DISPLAY 'INPUT CANNOT BE LONGER THAN:' FIELD-LEN
467     GOTO FIELD-EDIT-END
468   END-IF
469   IF FIELD-TYPE =  'A'
470     PERFORM EDIT-ALPHA
471   END-IF
472   IF FIELD-TYPE =  'N'
473     PERFORM EDIT-NUM
474   END-IF
475   IF FIELD-TYPE = 'Q'
476     PERFORM EDIT-YES-NO
477   END-IF
478   IF FIELD-ERR  =   'N'
479     CUST-NDX = FIELD-LOC - 1
480     MOVE  TERM-REC TERM-LEN  TO CUST-FIELD FIELD-LEN
481   END-IF
482 FIELD-EDIT-END. END-PROC
484 *
485 FIND-CUST. PROC
487 *
488 * GET A CUSTOMER ID, READ THE RECORD
489 * DO UNTIL VALID RECORD FOUND
490 *
491   CUST-ERR  =  'Y'
492   DO WHILE CUST-ERR EQ 'Y'
493     DISPLAY 'ENTER CUSTOMER ID'
494     PERFORM GET-LINE
495     IF KEY-IN NOT NUMERIC OR KEY-IN ZERO
496       DISPLAY 'INVALID CUSOMER ID, RE-ENTER'
497     ELSE
498       CUST-KEY = KEY-IN
499       PERFORM READ-CUST
500     END-IF
501   END-DO
502 END-PROC
503 *
504 GET-COMMAND. PROC
506 *
507 * DISPLAY AVAILABLE COMMANDS AND GET THE COMMAND FROM THE USER
508 * CONTINUE UNTIL VALID COMMAND ENTERED
509 *
510   CMD-FND  =   'N'
511   DO WHILE CMD-FND = 'N'
512     DSP-DATA = ' '
513     MOVE ZERO TO DSP-NDX CMD-NDX
514     DO WHILE CMD-LEN GT 0
515       IF CMD-STATE = CUR-STATE
516         DSP-COMMA = ','
517         MOVE CMD-NAME CMD-LEN TO DSP-CMD CMD-LEN
518         DSP-NDX = DSP-NDX + CMD-LEN + 2
519       END-IF
520       CMD-NDX = CMD-NDX + CMD-LEN + CMD-OVHD
```

```
521     END-DO
522     DISPLAY DSP-LINE
523     DISPLAY 'ENTER COMMAND'
524     PERFORM GET-LINE
525     IF CMD-IN SPACE AND CMD-DEFAULT GT 0
526       CMD-ID  =  CMD-DEFAULT
527       CMD-FND = 'Y'
528       GOTO NOT-FOUND
529     END-IF
530     CMD-NDX = 0
531     MOVE CMD-NAME CMD-LEN TO CMD-HOLD
532     DO WHILE CMD-HOLD NE CMD-IN OR  CMD-STATE NE CUR-STATE  -
          OR  TERM-LEN  NE CMD-LEN
533       IF  CMD-LEN EQ 0  . * ZERO LEN IS END OF TABLE
535           DISPLAY 'INVALID COMMAND:' CMD-IN
536           GOTO NOT-FOUND
537       ELSE
538           CMD-NDX = CMD-NDX + CMD-LEN + CMD-OVHD
539           MOVE CMD-NAME CMD-LEN TO CMD-HOLD
540       END-IF
541     END-DO
542     CMD-FND = 'Y'
543     CMD-ID  = CMD-NO
544   NOT-FOUND
545   END-DO
546 END-PROC
547 *
548 GET-FIELD. PROC
550 *
551 * GET ONE FIELD FROM THE USER FIELD NUMBER IS IN 'FIELD-ID'
552 * LOOP UNTIL A VALID FIELD IS ENTERED
553 *
554    SEARCH FIELDTBL WITH FIELD-ID GIVING FIELD-DATA
555    FIELD-ERR = 'Y'
556    IF NOT FIELDTBL
557      DISPLAY 'ERROR INVALID FIELD NUMBER'
558      GOTO GET-FIELD-ERR
559    END-IF
560    DO WHILE FIELD-ERR = 'Y'
561      DISPLAY 'ENTER VALUE FOR FIELD:' FIELD-ID +1 FIELD-NAME  -
         '(' FIELD-TYPE  ',' FIELD-LEN ' CHARACTERS)'
562      PERFORM GET-LINE
563      PERFORM FIELD-EDIT-MOVE
564    END-DO
565 GET-FIELD-ERR
566 END-PROC
567 *
568 GET-FIELD-NUMBER. PROC
570 *
571 * THIS ROUTINE ASKS FOR A NUMBER OF A FIELD
572 * LOOP UNTIL VALID NUMBER ENTERED
573 *
574  GET-FIELD-LOOP
575   PERFORM GET-LINE
576   IF FIRST3 = 'END'
577     GOTO GOT-FIELD-NUMBER
578   END-IF
579   IF TERM-LEN NE 3 OR TERM-FIELD-NO NOT NUMERIC
580     DISPLAY 'NUMBER MUST BE 3 DIGITS NUMERIC'
581     GOTO GET-FIELD-LOOP
582   END-IF
583 GOT-FIELD-NUMBER
584 END-PROC
585 *
586 GET-LINE. PROC
588 *
```

```
589 * GET A LINE OF USER INPUT
590 * LEFT JUSTIFY, TRUNCATE SPACES ON RIGHT
591 * SET SWITCHES IF  APLHA OR  NUMERIC OR OTHER DATA IS ENTERED
592 *
593   TERM-BEGIN-NDX  =   0
594   MOVE SPACE TO ALPHA-SW NUM-SW OTHER-SW  TERM-REC
595   DISPLAY ' '                              . * PROMPT IN SYNC
597   GET TERMIN
598   IF EOF TERMIN
599     DISPLAY 'USER REQUESTED ABORT'
600     STOP
601   END-IF
602   IF TERM-REC SPACE
603     TERM-LEN  = 1
604     GOTO GOT-LINE
605   END-IF
606   IF ECHO-SW = 'Y'
607     DISPLAY TERM-REC
608   END-IF
609 * SUPPRESS LEADING BLANKS
610   DO WHILE TERM-BEGIN-NDX   LS   79   -
        AND TERM-BEGIN-CH = ' '
611       TERM-BEGIN-NDX  =  TERM-BEGIN-NDX   +   1
612   END-DO
613                                  * HANDLE MASTER ESCAPE COMMAND
614   IF FIRST3  =  'ESC'
615     GOTO JOB
616   END-IF
617                                  * TRUNCATE TRAILING BLANKS
618   TERM-END-NDX  =   79
619   DO WHILE TERM-END-NDX  GT  TERM-BEGIN-NDX   -
        AND TERM-END-CH = ' '
620       TERM-END-NDX   =   TERM-END-NDX   -   1
621   END-DO
622                                  * SET NUMERIC, ALPHA, OTHER SWITCHES
623   TERM-LEN  =  TERM-END-NDX  -  TERM-BEGIN-NDX   + 1
624   DO WHILE TERM-END-NDX GE TERM-BEGIN-NDX
625     IF TERM-END-CH NUMERIC
626       NUM-SW  = 'Y'
627     END-IF
628     IF TERM-END-CH ALPHABETIC
629       ALPHA-SW    =   'Y'
630     END-IF
631     IF TERM-END-CH NOT NUMERIC AND TERM-END-CH NOT ALPHABETIC
632       OTHER-SW    =   'Y'
633     END-IF
634     TERM-END-NDX   =   TERM-END-NDX   -   1
635   END-DO
636   GOT-LINE
637 END-PROC
638 *
639 GET-YES-NO. PROC
641 *
642 * GET A YES OR NO ANSWER
643 *
644   ANSWER = ' '
645   DO WHILE ANSWER EQ ' '
646     PERFORM GET-LINE
647     IF TERM-REC EQ 'Y' 'N' 'NO' 'YES'
648       MOVE TERM-REC 1 TO ANSWER
649     ELSE
650       DISPLAY 'VALID ANSWER IS YES OR NO'
651       DISPLAY 'ENTER ANSWER'
652     END-IF
653   END-DO
654 END-PROC
```

```
655 *
656 HELLO. PROC
658 *
659 * START PROCEDURE, INTRODUCE YOURSELF, READ RECORD 0 TO GET
660 * NEXT FREE KEY
661 *
662 * TEST FOR EXISTANCE OF NEXT RECORD, IF IT IS THERE THEN SOME ERROR
663 * OCCURED DURING THE LAST DATA ENTRY SESSION
664 * THEN FIND NEXT AVAILABLE SLOT
665 *
666   DISPLAY NEWPAGE 'HELLO, WELCOME TO BRILLIG BANKS DATA ENTRY SYSTEM'
667   CUST-KEY     =  0
668   READ CUST KEY CUST-KEY  STATUS
669   PERFORM CUST-FILE-TEST
670   NEXT-KEY = NEXT-ID
671   POINT CUST GE NEXT-KEY                 . * TEST FOR ERROR
673   IF NOT EOF  CUST
674     DISPLAY 'INITIALIZATION ERROR, READING TILL FREE RECORD'
675   END-IF
676   DO WHILE NOT EOF  CUST
677     DISPLAY 'CUSTOMER ID:' NEXT-KEY ' FOUND, SKIPPING TO NEXT'
678     NEXT-KEY = NEXT-KEY + 1              . * IF THERE THEN ERROR
680     POINT CUST GE NEXT-KEY               . * LAST UPDATE SESSION
682   END-DO
683 END-PROC
684 *
685 NEXT-RECORD. PROC
687   GET CUST STATUS
688   PERFORM CUST-FILE-TEST
689   IF CUST
690     CUST-KEY = CUST-ID
691   ELSE
692     CUST-KEY = 0
693   END-IF
694 END-PROC
695 *
696 READ-CUST. PROC
698 *
699 * READ ONE RECORD, KEY:CUST-KEY
700 *
701   READ CUST KEY CUST-KEY   STATUS
702   IF  CUST
703     CUST-ERR = 'N'
704   ELSE
705     DISPLAY 'ERROR, CUSTOMER NOT FOUND:' CUST-KEY
706   END-IF
707 END-PROC
708 *
709 SELECT-RECORD. PROC
711   ANSWER = 'N'
712   IF CUST-KEY GT 0
713     DISPLAY 'CURRENT CUSTOMER IS:' CUST-KEY
714     DISPLAY 'IS THIS THE DESIRED RECORD(Y/N)?'
715     PERFORM GET-YES-NO
716   END-IF
717   IF ANSWER = 'Y'
718     PERFORM READ-CUST
719   ELSE
720     PERFORM FIND-CUST
721   END-IF
722 END-PROC
723 *
724 UPDATE-RECORD. PROC
726 *
727 * THIS RECORD WILL UPDATE THE FIELDS OF THE CURRENT RECORD
728 * THE USER SELECTS, BY FIELD NUMBER, THE FIELD TO UPDATE
```

```
729 * THEN ENTERS THE NEW DATA
730 * THIS CONTINUES UNTIL 'END' IS ENCOUNTERED
731 *
732   UPDATE-LOOP
733     DISPLAY 'ENTER FIELD NUMBER TO ALTER DATA, (IE 001)'
734     DISPLAY 'ENTER ''END'' WHEN READY TO UPDATE'
735     PERFORM GET-FIELD-NUMBER
736     IF FIRST3  NE  'END'
737       FIELD-ID = TERM-FIELD-NO
738       PERFORM GET-FIELD
739       GOTO UPDATE-LOOP
740     END-IF
741     DISPLAY 'READY TO UPDATE RECORD(Y/N)?'
742     PERFORM GET-YES-NO
743     IF ANSWER  =  'N'
744       GOTO UPDATE-LOOP
745     END-IF
746   WRITE CUST UPDATE STATUS
747   PERFORM CUST-FILE-TEST
748   DISPLAY 'RECORD ' CUST-ID ' SUCCESSFULLY UPDATED'
749 END-PROC

HELLO, WELCOME TO BRILLIG BANKS DATA ENTRY SYSTEM
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ADD
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)

MR
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)

TOM
ENTER VALUE FOR FIELD:003  LAST NAME          (A,15 CHARACTERS)

LEONARD
ENTER VALUE FOR FIELD:004  MIDDLE INITIAL     (A,01 CHARACTERS)

ENTER VALUE FOR FIELD:005  ADDRESS LINE 1     (X,25 CHARACTERS)

1781 ORANGE PLACE
ENTER VALUE FOR FIELD:006  ADDRESS LINE 2     (X,25 CHARACTERS)

ENTER VALUE FOR FIELD:007  CITY               (A,15 CHARACTERS)

CALCULAS
ENTER VALUE FOR FIELD:008  STATE              (A,02 CHARACTERS)

CA
ENTER VALUE FOR FIELD:009  ZIP                (X,09 CHARACTERS)

31095
ENTER VALUE FOR FIELD:010  IS IT LOCAL        (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:011  CREDIT RATING      (N,01 CHARACTERS)

5
ENTER VALUE FOR FIELD:012  SAVINGS            (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:013  CHECKING           (Q,01 CHARACTERS)

N
```

```
ENTER VALUE FOR FIELD:014  SAFE DEPOSIT BOX   (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:015  C AND D            (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:016  ALL SAVERS         (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:017  VISA               (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:018  MASTER CARD        (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:019  MONEY MARKET       (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:020  T BILL             (Q,01 CHARACTERS)

N
CUSTOMER 000001 ADDED
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ADD
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)

MISS
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)

JANE
ENTER VALUE FOR FIELD:003  LAST NAME          (A,15 CHARACTERS)

NEARY
ENTER VALUE FOR FIELD:004  MIDDLE INITIAL     (A,01 CHARACTERS)

I
ENTER VALUE FOR FIELD:005  ADDRESS LINE 1     (X,25 CHARACTERS)

887 DETOUR PLACE
ENTER VALUE FOR FIELD:006  ADDRESS LINE 2     (X,25 CHARACTERS)

ENTER VALUE FOR FIELD:007  CITY               (A,15 CHARACTERS)

MARLBOUROUGH
ENTER VALUE FOR FIELD:008  STATE              (A,02 CHARACTERS)

VA
ENTER VALUE FOR FIELD:009  ZIP                (X,09 CHARACTERS)

22211
ENTER VALUE FOR FIELD:010  IS IT LOCAL        (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:011  CREDIT RATING      (N,01 CHARACTERS)

8
ENTER VALUE FOR FIELD:012  SAVINGS            (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:013  CHECKING           (Q,01 CHARACTERS)

Y
```

```
ENTER VALUE FOR FIELD:014  SAFE DEPOSIT BOX   (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:015  C AND D            (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:016  ALL SAVERS         (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:017  VISA               (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:018  MASTER CARD        (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:019  MONEY MARKET       (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:020  T BILL             (Q,01 CHARACTERS)

Y
CUSTOMER 000002 ADDED
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

BROWSE
YOU CAN ENTER UP TO  20 QUERIES FOR THE SEARCH
THE QUERIES WILL BE 'ANDED' TOGETHER
ENTER FIELD NUMBER FOR SEARCH 'END' WHEN DONE

002
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)

TOM
ENTER FIELD NUMBER FOR SEARCH 'END' WHEN DONE

001
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)

MR
ENTER FIELD NUMBER FOR SEARCH 'END' WHEN DONE

END

CUST-ID:000001

FIELD:001  TITLE               MR
FIELD:002  FIRST NAME          TOM
FIELD:003  LAST NAME           LEONARD
FIELD:004  MIDDLE INITIAL
FIELD:005  ADDRESS LINE 1      1781 ORANGE PLACE
FIELD:006  ADDRESS LINE 2
FIELD:007  CITY                CALCULAS
FIELD:008  STATE               CA
FIELD:009  ZIP                 31095
FIELD:010  IS IT LOCAL         N
FIELD:011  CREDIT RATING       5
FIELD:012  SAVINGS             N
FIELD:013  CHECKING            N
FIELD:014  SAFE DEPOSIT BOX    N
FIELD:015  C AND D             N
FIELD:016  ALL SAVERS          Y
FIELD:017  VISA                Y
FIELD:018  MASTER CARD         Y
FIELD:019  MONEY MARKET        Y
FIELD:020  T BILL              N
```

```
VALID COMMANDS ARE:ESC, END, NEXT
ENTER COMMAND

END
BROWSE COMPLETE
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ADD
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)

MR
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)

KIP
ENTER VALUE FOR FIELD:003  LAST NAME          (A,15 CHARACTERS)

LING
ENTER VALUE FOR FIELD:004  MIDDLE INITIAL     (A,01 CHARACTERS)

ENTER VALUE FOR FIELD:005  ADDRESS LINE 1     (X,25 CHARACTERS)

227 BEETLE LN
ENTER VALUE FOR FIELD:006  ADDRESS LINE 2     (X,25 CHARACTERS)

ENTER VALUE FOR FIELD:007  CITY               (A,15 CHARACTERS)

PALMER
ENTER VALUE FOR FIELD:008  STATE              (A,02 CHARACTERS)

MA
ENTER VALUE FOR FIELD:009  ZIP                (X,09 CHARACTERS)

01072
ENTER VALUE FOR FIELD:010  IS IT LOCAL        (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:011  CREDIT RATING      (N,01 CHARACTERS)

1
ENTER VALUE FOR FIELD:012  SAVINGS            (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:013  CHECKING           (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:014  SAFE DEPOSIT BOX   (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:015  C AND D            (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:016  ALL SAVERS         (Q,01 CHARACTERS)

Y
ENTER VALUE FOR FIELD:017  VISA               (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:018  MASTER CARD        (Q,01 CHARACTERS)

N
ENTER VALUE FOR FIELD:019  MONEY MARKET       (Q,01 CHARACTERS)

N
```

```
ENTER VALUE FOR FIELD:020  T BILL             (Q,01 CHARACTERS)

Y
CUSTOMER 000003 ADDED
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

DISPLAY
CURRENT CUSTOMER IS:000003
IS THIS THE DESIRED RECORD(Y/N)?

Y

FIELD:001  TITLE               MR
FIELD:002  FIRST NAME          KIP
FIELD:003  LAST NAME           LING
FIELD:004  MIDDLE INITIAL
FIELD:005  ADDRESS LINE 1      227 BEETLE LN
FIELD:006  ADDRESS LINE 2
FIELD:007  CITY                PALMER
FIELD:008  STATE               MA
FIELD:009  ZIP                 01072
FIELD:010  IS IT LOCAL         N
FIELD:011  CREDIT RATING       1
FIELD:012  SAVINGS             Y
FIELD:013  CHECKING            Y
FIELD:014  SAFE DEPOSIT BOX    N
FIELD:015  C AND D             N
FIELD:016  ALL SAVERS          Y
FIELD:017  VISA                N
FIELD:018  MASTER CARD         N
FIELD:019  MONEY MARKET        N
FIELD:020  T BILL              Y
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

UPD
CURRENT CUSTOMER IS:000003
IS THIS THE DESIRED RECORD(Y/N)?

Y
ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

001
ENTER VALUE FOR FIELD:001  TITLE              (A,04 CHARACTERS)

MS
ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

002
ENTER VALUE FOR FIELD:002  FIRST NAME         (A,10 CHARACTERS)

LOIS
ENTER FIELD NUMBER TO ALTER DATA, (IE 001)
ENTER 'END' WHEN READY TO UPDATE

END
READY TO UPDATE RECORD(Y/N)?

Y
RECORD 000003 SUCCESSFULLY UPDATED
```

```
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

DEL
CURRENT CUSTOMER IS:000003
IS THIS THE DESIRED RECORD(Y/N)?

N
ENTER CUSTOMER ID

000001
RECORD 000001 DELETED
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

ESC
VALID COMMANDS ARE:ESC, END, DISPLAY, ADD, UPD, DEL, ECHO, NOECHO, BROWSE
ENTER COMMAND

END
DATA ENTRY SESSION COMPLETE
```

Batch Processing

The efficient batch processing capabilities of CA-Easytrieve Plus enable you to use the data you have entered into the file for many purposes. Several examples are presented here.

Detail Report

One practical use for this file is to output a list of all bank customers with a list of their individual accounts. Example 16.3, as shown in the next exhibit, illustrates the coding and the resulting report. In this example, the START parameter of the JOB statement specifies to execute the specified procedure (HELLO) after opening all the files, but before reading the first record. The HELLO procedure positions the file to the next record after record zero, which contains no data, only the key of the next available record in this file. This enables all subsequent program logic to operate only on records containing valid data.

Each record is searched for a 1 in the fields that contain the account names - field numbers 012 through 020. If it is found, that account name is stored into a working storage field named ACCT-WORK, which eventually is printed on the report along with the corresponding customer name. In the REPORT group, the CONTROL command provides a total of the number of accounts for each customer. The NOPRINT option suppresses printing the total line for the FIRST-NAME control break.

```
 1 *
 2 *  EXAMPLE 16.3 DETAIL LISTING OF CUSTOMERS
 3 *
 4 %BANKLIB
45 ACCT-ID   W    2      N
46 ACCT-WORK W    30     A
47 *
48 JOB  START HELLO
49 *
50   ACCT-NDX = 0
51   DO WHILE ACCT-NDX LT ACCT-MAX
52     IF ACCT-IND = 1
53       ACCT-ID  =  ACCT-NDX  +  1
54       SEARCH ACCTNAME WITH ACCT-ID GIVING ACCT-WORK
55       PRINT ACCT-DETAIL
56     END-IF
57     ACCT-NDX = ACCT-NDX + 1
58   END-DO
59 HELLO. PROC
61   CUST-KEY = 1
62   POINT CUST GE CUST-KEY
63 END-PROC
64 REPORT ACCT-DETAIL LINESIZE 80 SPACE 1
65   SEQUENCE LAST-NAME
66   CONTROL LAST-NAME FIRST-NAME NOPRINT
67   TITLE 'LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS'
68   HEADING ACCT-WORK ('ACCOUNT' 'NAME')
69   LINE LAST-NAME FIRST-NAME ACCT-WORK  TALLY
```

```
12/02/83            LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS       PAGE      1

                                                  ACCOUNT
          LAST-NAME     FIRST-NAME                  NAME                TALLY

        DEMO            TOM         SAFE DEPOSIT
                                    CERTIFICATE OF DEPOSIT
                                    ALL SAVERS
                                    VISA
                                    MASTER CARD
                                    MONEY MARKET
        DEMO                                                              6

        GOLFER          GOOD        MONEY MARKET
                                    TREASURY BILL
                                    MASTER CARD
                                    SAVINGS
                                    CHECKING
                                    SAFE DEPOSIT
                                    CERTIFICATE OF DEPOSIT
                                    ALL SAVERS
                                    VISA
        GOLFER                                                            9

        HANDHOLDER      HANNA       SAVINGS
                                    CHECKING
                                    SAFE DEPOSIT
                                    CERTIFICATE OF DEPOSIT
                                    ALL SAVERS
                                    VISA
                                    MASTER CARD
                                    MONEY MARKET
        HANDHOLDER                                                        8
```

```
JOHANAS         JAMES      SAFE DEPOSIT
                           CERTIFICATE OF DEPOSIT
                           ALL SAVERS
                           VISA
                           TREASURY BILL
                           CHECKING
                           SAVINGS
JOHANAS                                                        7


12/02/83           LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS       PAGE      2

                                          ACCOUNT
   LAST-NAME    FIRST-NAME                 NAME                TALLY

KIPPER          JON        SAVINGS
                           CHECKING
                           SAFE DEPOSIT
                           CERTIFICATE OF DEPOSIT
                           ALL SAVERS
KIPPER                                                         5

LENGTHY         ROD        SAFE DEPOSIT
                           CERTIFICATE OF DEPOSIT
                           VISA
                           MONEY MARKET
                           MASTER CARD
LENGTHY         ROGER      VISA
                           CERTIFICATE OF DEPOSIT
                           MASTER CARD
                           MONEY MARKET
                           SAFE DEPOSIT
LENGTHY                                                       10

LING            LOIS       TREASURY BILL
                           SAVINGS
                           ALL SAVERS
                           CHECKING
LING                                                           4

LONELY          ALICE      SAVINGS
                           CHECKING
                           SAFE DEPOSIT
                           CERTIFICATE OF DEPOSIT
                           ALL SAVERS
                           VISA
                           MONEY MARKET
                           MASTER CARD
LONELY                                                         8
```

12/02/83 LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS PAGE 3

LAST-NAME	FIRST-NAME	ACCOUNT NAME	TALLY
LONG	HARRY	SAVINGS	
		CHECKING	
		SAFE DEPOSIT	
LONG			3
MIDDLEMAN	WILLIAM	SAFE DEPOSIT	
		ALL SAVERS	
		VISA	
		MASTER CARD	
		MONEY MARKET	
		CERTIFICATE OF DEPOSIT	
MIDDLEMAN			6
NEARY	JANE	ALL SAVERS	
		CHECKING	
		TREASURY BILL	
		SAFE DEPOSIT	
		SAVINGS	
NEARY			5
NICE	ANNE	MASTER CARD	
		SAFE DEPOSIT	
		VISA	
		SAVINGS	
		MONEY MARKET	
		CHECKING	
		CERTIFICATE OF DEPOSIT	
NICE			7
RIGHT	RONALD	SAVINGS	
		CHECKING	
		ALL SAVERS	
		VISA	
		MASTER CARD	
		MONEY MARKET	
		TREASURY BILL	
RIGHT			7

```
12/02/83          LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS       PAGE      4

                                                    ACCOUNT
     LAST-NAME      FIRST-NAME                       NAME                TALLY

     ROY            ROLAND       ALL SAVERS
                                 MASTER CARD
                                 SAVINGS
                                 CHECKING
                                 VISA
                                 MONEY MARKET
     ROY                                                                   6

     SHORE          KAREN        ALL SAVERS
                                 SAVINGS
                                 MASTER CARD
                                 MONEY MARKET
     SHORE                                                                 4

     SHORT          DAVE         CHECKING
                                 SAFE DEPOSIT
                                 CERTIFICATE OF DEPOSIT
                                 ALL SAVERS
                                 VISA
                                 MASTER CARD
                                 MONEY MARKET
                                 SAVINGS
     SHORT          MARY         MASTER CARD
                                 MONEY MARKET
                                 VISA
                                 CERTIFICATE OF DEPOSIT
     SHORT                                                                12

     SHWONT         WANDA        SAVINGS
                                 VISA
                                 MONEY MARKET
                                 CERTIFICATE OF DEPOSIT
                                 ALL SAVERS
                                 MASTER CARD
     SHWONT                                                               6
```

```
12/02/83          LIST OF ALL CUSTOMERS WITH THEIR ACCOUNTS       PAGE      5

                                                    ACCOUNT
      LAST-NAME     FIRST-NAME                      NAME                TALLY

     STEWART        ROD          VISA
                                 MASTER CARD
                                 CERTIFICATE OF DEPOSIT
                                 SAFE DEPOSIT
                                 MONEY MARKET
     STEWART                                                              5

     TRUMPET        JIM          SAVINGS
                                 CHECKING
                                 TREASURY BILL
                                 ALL SAVERS
                                 VISA
                                 MASTER CARD
                                 MONEY MARKET
     TRUMPET                                                              7

     VAN GOGH       VINCENT      TREASURY BILL
                                 VISA
                                 SAVINGS
                                 CHECKING
     VAN GOGH                                                             4

     WILCHE         NANCY        MASTER CARD
                                 VISA
                                 CERTIFICATE OF DEPOSIT
                                 MONEY MARKET
                                 SAFE DEPOSIT
     WILCHE                                                               5

     YAZBASEY       CARL         SAVINGS
                                 MONEY MARKET
                                 CERTIFICATE OF DEPOSIT
                                 ALL SAVERS
                                 VISA
                                 CHECKING
     YAZBASEY                                                             6

                                                                        140
```

Mass Mailing

Another use for the file developed from the previous DETAIL Report is the generation of a letter to all customers who do not have an ALL-SAVERS account, notifying them of the potential savings this account can provide. As shown in the next exhibit, Example 16.4 illustrates the coding required, and the letter and the labels.

The PERFORM statement executes the CONCAT-NAME procedure that links the title (MS, MR, or whatever) with the name. The IF statement tests the value of field ALL SAVERS as not-equal 1. If this tests true, the customer does not have an ALL-SAVERS account and the statements beneath the IF are executed. The mailing label and the letter are printed.

```
  1 PARM LIST(NOPARM NOFILE)
  2 *
  3 * BANK EXAMPLE 16.4 MAILING EXAMPLE
  4 *
  5 %BANKLIB
 46 FILE LBLSOUT PRINTER
 47 FILE LETTERS PRINTER
 48   WHOLE-NAME  W  30  A
 49 JOB   START HELLO
 50   PERFORM CONCAT-NAME
 51   IF ALL-SAVERS NE 1
 52     PRINT MAILING-LABELS
 53     PERFORM CONCAT-COMMA
 54     PRINT ALL-SAVERS-LETTER
 55   END-IF
 56 CONCAT-NAME. PROC
 58   WHOLE-NAME = PERS-TITLE
 59   %CONCAT WHOLE-NAME  1  FIRST-NAME
 86   %CONCAT WHOLE-NAME  1  MIDDLE-INITIAL
113   %CONCAT WHOLE-NAME  1  LAST-NAME
140 END-PROC
141 *
142 CONCAT-COMMA. PROC
144   %CONCAT WHOLE-NAME 0 ''','''
171 END-PROC
172 *
173 HELLO. PROC
175   CUST-KEY = 1
176   POINT CUST GE CUST-KEY
177 END-PROC
178 *
179 REPORT ALL-SAVERS-LETTER   NOHEADING    NOADJUST -
          LINESIZE  80  PRINTER LETTERS
180   SEQUENCE ZIP
181   CONTROL CUST-ID NEWPAGE
182   LINE 1 'DEAR' WHOLE-NAME
183   LINE 3 +5 'The new ALL SAVERS CERTIFICATES are now available.'
184   LINE 4 +5 'They offer several advantages over other investments.'
185   LINE 5 +5 'We have set up a special telephone number to answer'
186   LINE 6 +5 'your questions regarding this new and exciting ONE TIME '
187   LINE 7 +5 'chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE'
188   LINE 8 +5 'during normal working hours and we will be happy to help'
189   LINE 09 +5 'you.'
190   LINE 11 +30 'SINCERELY,'
191   LINE 13 +30 'George Million'
192   LINE 14 +30 'Vice President'
193   LINE 15 +30 'Fourth National Bank of Virginia'
194 *
195 REPORT MAILING-LABELS PRINTER LBLSOUT LABELS (ACROSS 2)
196   SEQUENCE ZIP
197   LINE 1 WHOLE-NAME
198   LINE 2 ADDRESS1
199   LINE 3 ADDRESS2
200   LINE 4 CITY -2 STATE -2  ZIP
```

```
DEAR   MISS MARY SHORT,

     The new ALL SAVERS CERTIFICATES are now available.
     They offer several advantages over other investments.
     We have set up a special telephone number to answer
     your questions regarding this new and exciting ONE TIME
     chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
     during normal working hours and we will be happy to help
     you.

                              SINCERELY,

                              George Million
                              Vice President
                              Fourth National Bank of Virginia

---------------------------------------------------------------------

DEAR   MR ROGER N LENGTHY,

     The new ALL SAVERS CERTIFICATES are now available.
     They offer several advantages over other investments.
     We have set up a special telephone number to answer
     your questions regarding this new and exciting ONE TIME
     chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
     during normal working hours and we will be happy to help
     you.

                              SINCERELY,

                              George Million
                              Vice President
                              Fourth National Bank of Virginia

---------------------------------------------------------------------

DEAR   MR ROD N LENGTHY,

     The new ALL SAVERS CERTIFICATES are now available.
     They offer several advantages over other investments.
     We have set up a special telephone number to answer
     your questions regarding this new and exciting ONE TIME
     chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
     during normal working hours and we will be happy to help
     you.

                              SINCERELY,

                              George Million
                              Vice President
                              Fourth National Bank of Virginia
```

DEAR MRS HARRY K LONG,

The new ALL SAVERS CERTIFICATES are now available.
They offer several advantages over other investments.
We have set up a special telephone number to answer
your questions regarding this new and exciting ONE TIME
chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
during normal working hours and we will be happy to help
you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MRS NANCY WILCHE,

The new ALL SAVERS CERTIFICATES are now available.
They offer several advantages over other investments.
We have set up a special telephone number to answer
your questions regarding this new and exciting ONE TIME
chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
during normal working hours and we will be happy to help
you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MS ANNE NICE,

The new ALL SAVERS CERTIFICATES are now available.
They offer several advantages over other investments.
We have set up a special telephone number to answer
your questions regarding this new and exciting ONE TIME
chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE
during normal working hours and we will be happy to help
you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MR VINCENT I VAN GOGH,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

DEAR MR ROD STEWART,

The new ALL SAVERS CERTIFICATES are now available. They offer several advantages over other investments. We have set up a special telephone number to answer your questions regarding this new and exciting ONE TIME chance, at TAX FREE INTEREST. PLEASE CALL (703) ALL-SAVE during normal working hours and we will be happy to help you.

SINCERELY,

George Million
Vice President
Fourth National Bank of Virginia

MISS MARY SHORT
100 THIS PLACE
APT 200B
BELCHERTOWN MA 01003

MR ROD N LENGTHY
111 BOTTLE LN
WILBRAHAM NC 01072

MR ROGER N LENGTHY
121 TOTTLE LN
WILBRAHAM FL 01072

MRS HARRY K LONG
2000 CALCUTTA ST
SPRINGFIELD VA 22152

MRS NANCY WILCHE
1006 ROUND CIRCLE
NORTHHAMPTON WA 22672

MS ANNE NICE
171 LEE HIGHWAY
LEESBURG VA 22672

MR VINCENT I VAN GOGH
1 ERIE AVE
OAK BROOK IL 30072

MR ROD STEWART
1 MAGGIE LANE
APT 1001
CLARENTON CA 50072

Summary Report

As shown in the next exhibit, Example 16.5 produces a summary of the customer accounts by location; in this case by state. Summary reports can have a variety of uses and they can be sequenced on any field.

This example also provides totals by state, through the CONTROL command. This summary report does not list customers by name, but simply counts the total number of accounts for each account type.

```
 1 *
 2 * BANK EXAMPLE 16.5 - SUMMARY REPORT
 3 *
 4 %BANKLIB
45 *
46 *
47 JOB    START HELLO
48   PRINT  ACCT-SUMMARY
49 *
50 HELLO. PROC
52   CUST-KEY = 1
53   POINT  CUST  GE CUST-KEY
54 END-PROC
55 *
56 REPORT   ACCT-SUMMARY   SUMMARY  SPACE 1  LINESIZE 80
57   SEQUENCE   STATE
58   CONTROL    STATE
59   TITLE   1  'SUMMARY OF ACCOUNTS BY STATE'
60   LINE    1  STATE    SAVINGS                    -
                         CHECKING                   -
                         SAFE-DEPOSIT               -
                         C-AND-D                    -
                         ALL-SAVERS                 -
                         VISA                       -
                         MASTER-CARD                -
                         MONEY-MARKET               -
                         T-BILL


12/02/83                 SUMMARY OF ACCOUNTS BY STATE              PAGE      1

                                SAFE            ALL         MASTER MONEY
    STATE SAVINGS CHECKING DEPOSIT C-AND-D SAVERS VISA   CARD  MARKET T-BILL

     CA       2       2        2       2      2      3     2      2      2
     DC       1       1                        1      1     1      1      1
     FL       1       1        2       2      1      2     2      2      1
     IL       1       1                               1                  1
     MA       3       3        1       3      4      4     3      4      1
     MD       1                        1      1      1     1      1
     MI                        1       1      1      1     1      1
     NC                        1       1             1     1      1
     NH       1       1        1       1      1      1     1      1
     NY       2       1        1       1      2      1     2      2
     VA       5       5        5       3      3      2     2      2      1
     WA                        1       1             1     1      1
             17      15       15      16     16     19    17     18      7
```

Chapter

17 Project Management System

This chapter illustrates how CA-Easytrieve Plus might be used for a classical data processing application: a Project Management System. The structure is typical of many data processing systems, and is composed of:

1. The Project Master file, defined in Example 17.1.
2. The File Maintenance Program, illustrated in Example 17.1. The output from the file update process is illustrated in Example 17.2.
3. Various report programs, illustrated in Examples 17.3, 17.4, and 17.5. The reports are generated using data from the updated Master File.

There are several reasons why Project Management was chosen as an example. However, the primary consideration is that it enables us to present a complete base to a working system. Although this system is somewhat simple compared to many Project Management Systems, it is usable for many applications.

Master File Layout

The master file is a physical sequential file containing fixed length records. There are two record types, the project record and the task record. A project definition is composed of a project record and optionally, one or more task records, as illustrated below:

```
project record for project 1
task record for task 1 of project 1
task record for task 2 of project 1
task record for task 3 of project 1
  .
project record for project 2
task record for task 1 of project 2
task record for task 2 of project 2
  .
project record for project 3
task record for task 1 of project 3
task record for task 2 of project 3
  .
```

The formats of the project and task records are the same - only the data is different. This combination enables us to represent many projects on one file, with each project having its own set of tasks, as shown in the following report.

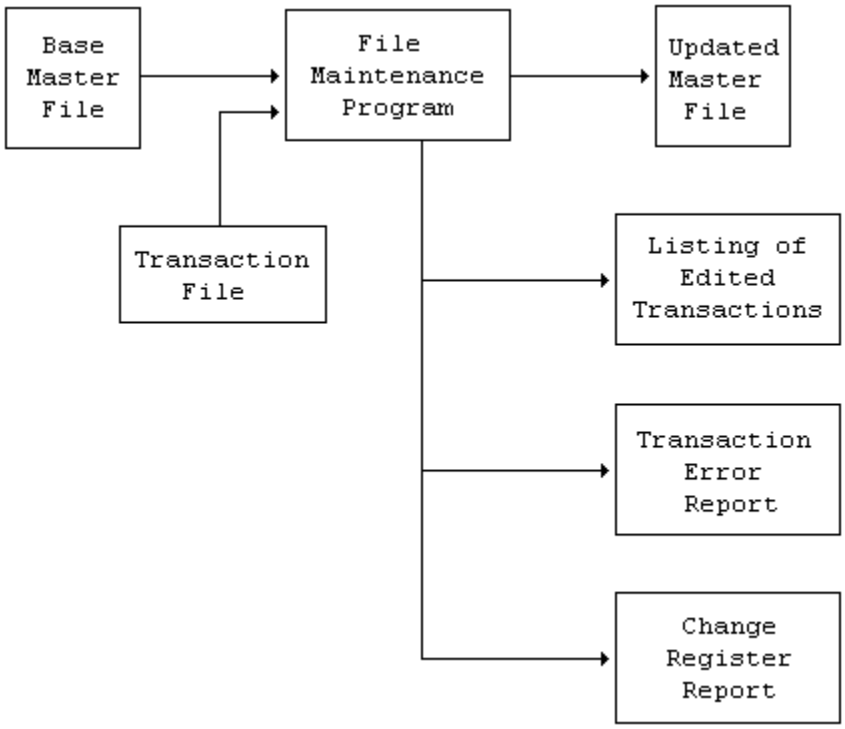
```
MACRO FILE-PARMS
*
*    PROJECT MANAGEMENT SYSTEM -- MASTER FILE DEFINITION
*
FILE PRJIN   FB (110 4400)   &FILE-PARMS
*
  PROJ-REC      1      110     A
  PROJ-TASK                          1       10     A
    PROJ-NO                          1       5      A    -
       HEADING ('PROJECT' 'NUMBER')
    TASK-NO                          6       5      A    -
       HEADING ('TASK' 'NUMBER')
    NAME                             11      25     A
    MANAGER-ID                       36      5      A    -
       HEADING ('MANAGER' 'ID')
*
  STATISTICS                         41      39     A
*
    ESTIMATED-DATA                   41      12     A
      EST-MAN-HRS                    41      4      P  1  -
        HEADING ('EST' 'HOURS')
      EST-START-DATE                 45      4      P     -
        HEADING ('EST' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
      EST-END-DATE                   49      4      P     -
        HEADING ('EST' 'END' 'DATE')   MASK(BWZ 'Z99/99/99')
*
    ACTUAL-DATA                      53      12     A
      ACT-MAN-HRS-SO-FAR             53      4      P  1  -
        HEADING ('ACTUAL' 'HOURS')
      ACT-START-DATE                 57      4      P     -
        HEADING ('ACTUAL' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
      ACT-END-DATE                   61      4      P     -
        HEADING ('ACTUAL' 'END' 'DATE')   MASK(BWZ 'Z99/99/99')
*
    LAST-TRANSACTION-DATE            65      4      P     -
        HEADING ('LAST' 'TRANS' 'DATE')   MASK(BWZ 'Z99/99/99')
    LAST-ACTIVITY-HRS                69      4      P  1
    PCT-COMPLETE                     73      2      P     -
        HEADING ('PERCENT' 'COMPLETE')
    PREDECESSOR-TABLE                75      7      A  OCCURS 5    -
               INDEX PRE-NDX
      PRE-NO         PREDECESSOR-TABLE      5      A
      PRE-PCT        PREDECESSOR-TABLE      2      P
      PRE-NO1                          75      5      A
      PRE-PCT1                         80      2      P
      PRE-NO2                          82      5      A
      PRE-PCT2                         87      2      P
      PRE-NO3                          89      5      A
      PRE-PCT3                         94      2      P
      PRE-NO4                          96      5      A
      PRE-PCT4                         101     2      P
      PRE-NO5                          103     5      A
      PRE-PCT5                         108     2      P
*
```

Programs

There are two classes of programs that operate on the Project Master -- file maintenance and report generation.

File Maintenance

The file maintenance program is a classical update program, diagrammed below:



Transactions are matched against the Base Master File. The requested operation is performed (if valid), the Updated Master File is produced, and three reports are generated that indicate the success of the run:

1. Listing of Edited Transactions
2. Transaction Error Report
3. Change Register.

These reports are illustrated in Examples 17.4 and 17.5.

Transaction Record Format

The transaction file consists of transactions to a particular project or task. The table below describes the format of the transaction record.

Name	Position	Description
TRANS-ID	1-2	Determines what to do (add, delete, and so forth)
TRANS-PROJ	3-7	Project ID of related master
TRANS-TASK	8-12	Task ID of master (if blank, performs transaction against project record)
TRANS-FIELD	13-17	Field ID of the field to be accessed
TRANS-DATA	18-?	Actual data to add/update master fields

Transaction Codes

As you can see, each transaction refers not only to a particular project/task, but also to a particular field of that record. To add a new project, one transaction is needed for each field to be added. The table below lists the valid transactions.

Transaction Code	Description
AP,AT	ADD transaction; AP=add project, AT=add task
DP,DT	DELETE transaction for project, or task
CP,CT	Change transaction for project, or task
IT	Increment transaction to add to task field

Field Types

The system sorts the transactions in order of DELETES, ADDS, CHANGES, and INCREMENTS for a given project/task. Thus, it is possible to delete a project, then ADD it again all in one run. Data in the TRANS-DATA field (positions 18-?) is edited to correspond to the field it pertains to. The table below lists the different types of fields.

Field Type	Description	Positions
DATES	date fields	18-23
PCT	percent fields	18-20
ID fields	project/task numbers, etc.	18-22
NAME	project/task name	18-42
HOURS*	hour fields	18-24 (1 decimal)
*To produce 123,456.7, enter 1234567 into positions 18-24.		

Examples 17.1 and 17.2 illustrate two functions; building a new Master File, then updating it.

1. As shown earlier in the Transaction Record Format table, Example 17.1 lists the CA-Easytrieve Plus code that both creates the new Master File, and subsequently updates it. The function performed by running the CA-Easytrieve Plus program depends upon the items contained in the Transaction file, TRANSIN, which are identified by the codes illustrated earlier in the Transaction Codes table. To create the new Master File, the program is first run with TRANSIN containing only Project (AP) and Add Task (AT) items. The input file is listed at the end of the run in the Listing of Edited Transactions described in the Change Register Report at the end of this chapter. There is no Transaction Error Report generated for this run.
2. As shown earlier in the Transaction Codes table, Example 17.2 illustrates the output reports after the program is run a second time with new items in TRANSIN, which include change task transactions (CT), increment transactions (IT), add project (AP) and add task (AT) transactions. The new input data is listed in the Listing of Edited Transactions and described in the Change Register Report later in this chapter. A duplicate transaction is listed on the Transaction Error Report at the end of this chapter.

Project Status: Example 17.1

```
 1 *
 2 * PROJECT MANAGEMENT SYSTEM EXAMPLE 17.1: ADD DATA TO FILE
 3 *
 4 PARM DEBUG(STATE FLOW FLDCHK)  LIST(NOFILE NOPARM)
 5 %PROJLIB
 6 *
 7 *    PROJECT MANAGEMENT SYSTEM -- MASTER FILE DEFINITION
 8 *
 9 FILE PRJIN   FB (110 4400)
10 *
11   PROJ-REC      1      110     A
12   PROJ-TASK                          1       10      A
13     PROJ-NO                          1       5       A    -
          HEADING ('PROJECT' 'NUMBER')
14     TASK-NO                          6       5       A    -
          HEADING ('TASK' 'NUMBER')
15     NAME                             11      25      A
16     MANAGER-ID                       36      5       A    -
          HEADING ('MANAGER' 'ID')
17 *
18   STATISTICS                         41      39      A
19 *
20     ESTIMATED-DATA                   41      12      A
21       EST-MAN-HRS                    41      4       P  1  -
           HEADING ('EST' 'HOURS')
22       EST-START-DATE                 45      4       P     -
           HEADING ('EST' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
23       EST-END-DATE                   49      4       P     -
           HEADING ('EST' 'END' 'DATE')   MASK(BWZ 'Z99/99/99')
24 *
25     ACTUAL-DATA                      53      12      A
26       ACT-MAN-HRS-SO-FAR             53      4       P  1  -
           HEADING ('ACTUAL' 'HOURS')
27       ACT-START-DATE                 57      4       P     -
           HEADING ('ACTUAL' 'START' 'DATE') MASK(BWZ 'Z99/99/99')
28       ACT-END-DATE                   61      4       P     -
           HEADING ('ACTUAL' 'END' 'DATE')   MASK(BWZ 'Z99/99/99')
29 *
30     LAST-TRANSACTION-DATE            65      4       P     -
           HEADING ('LAST' 'TRANS' 'DATE')   MASK(BWZ 'Z99/99/99')
31     LAST-ACTIVITY-HRS                69      4       P  1
32     PCT-COMPLETE                     73      2       P     -
           HEADING ('PERCENT' 'COMPLETE')
33     PREDECESSOR-TABLE                75      7       A  OCCURS 5    -
                 INDEX PRE-NDX
34       PRE-NO        PREDECESSOR-TABLE      5       A
35       PRE-PCT       PREDECESSOR-TABLE      2       P
36       PRE-NO1                        75      5       A
37       PRE-PCT1                       80      2       P
38       PRE-NO2                        82      5       A
39       PRE-PCT2                       87      2       P
40       PRE-NO3                        89      5       A
41       PRE-PCT3                       94      2       P
42       PRE-NO4                        96      5       A
43       PRE-PCT4                       101     2       P
44       PRE-NO5                        103     5       A
45       PRE-PCT5                       108     2       P
46 *
47 *
48 FILE TRANSWK  VIRTUAL RETAIN F 80
49   TRANS-REC                          1       78      A
50     TRANS-KEY                        1       17      A
51     TRANS-ID                         1       2       A
52       TRANS-ID-CMD                   1       1       A
```

```
 53       TRANS-ID-OP                    2       1      A
 54     TRANS-PROJ-TASK                  3       10     A
 55       TRANS-PROJ                     3       5      A
 56       TRANS-TASK                     8       5      A
 57     TRANS-FIELD                      13      5      A
 58     TRANS-DATA                       18      25     A
 59     TRANS-DATE                       18      6      N
 60     TRANS-L7-D1                      TRANS-DATA 7   N 1
 61     TRANS-L7-D0                      TRANS-DATA 7   N 0
 62     TRANS-L3-D0                      TRANS-DATA 3   N 0
 63     TRANS-L7                         TRANS-DATA 7   N
 64     TRANS-L3                         TRANS-DATA 3   N
 65     TRANS-SCAN       TRANS-DATA              1      A  -
              INDEX TRANS-NDX
 66     TRANS-ID-CNV                     79      2      A
 67 *
 68 * WORKING STORAGE
 69 *
 70     ADD-PROJ-ID        W                 2   A    VALUE 'AP'
 71       ADD-CMD   ADD-PROJ-ID              1   A
 72       PROJ-OP   ADD-PROJ-ID           +1 1   A
 73     ADD-TASK-ID        W                 2   A    VALUE 'AT'
 74       TASK-OP   ADD-TASK-ID              1   A
 75     CHANGE-PROJ-ID     W                 2   A    VALUE 'CP'
 76       CHANGE-CMD  CHANGE-PROJ-ID         1   A
 77       CHANGE-TASK-ID   W                 2   A    VALUE 'CT'
 78     DELETE-PROJ-ID     W                 2   A    VALUE 'DP'
 79       DELETE-CMD  DELETE-PROJ-ID         1   A
 80       DELETE-TASK-ID   W                 2   A    VALUE 'DT'
 81     INC-TASK-ID        W                 2   A    VALUE 'IT'
 82       INC-CMD     INC-TASK-ID            1   A
 83   MSG                W          55          A
 84   CHANGE-MSG         W          15          A
 85   TRANS-ERR-FLAG     W          1           A
 86   FIELD-DATA         W          25          A
 87   MASTER-WAITING     W          1           A
 88   LAST-PROJ          W   PROJ-NO
 89   LAST-TASK          W   PROJ-NO
 90   DELETED-PROJ       W   PROJ-NO
 91   DELETED-TASK       W   PROJ-NO
 92   LAST-TRANS-KEY     W          TRANS-KEY
 93   LAST-TRANS-ID      W   TRANS-ID
 94     LAST-TRANS-CMD  LAST-TRANS-ID     1     A
 95     LAST-TRANS-OP   LAST-TRANS-ID +1  1     A
 96   TRANS-ERR-COUNT    W          3           P  MASK('ZZZZ9')
 97   SYSTEM-DATE        W          4           P
 98 *
 99 * RECEIVING AREA FOR SEARCH ON TRANSACTION TABLE
100 *
101   TRANS-ATTRIBUTES     W                40  A
102     TRANS-TBL-CNV         TRANS-ATTRIBUTES  2   A
103     TRANS-MSG             TRANS-ATTRIBUTES +3 20  A
104 *
105 * RECEIVING AREA FOR SEARCH ON FIELD TABLE
106 *
107   FIELD-ATTRIBUTES   W            63        A
108     FIELD-NAME FIELD-ATTRIBUTES   25        A
109     FIELD-OFFSET FIELD-ATTRIBUTES +25 3     N 0
110     FIELD-LEN    FIELD-ATTRIBUTES +29 2     N 0
111     FIELD-TYPE   FIELD-ATTRIBUTES +32 1     A
112     FIELD-DEC    FIELD-ATTRIBUTES +34 1     N
113     FIELD-SAVE   FIELD-ATTRIBUTES +36 3     N   . * XREF TO LOCATION
115                                  * WHERE INC TRANSACTION TO BE SAVED
116   FIELD-ID           W             5        A
117 *
118 * THE FOLLOWING TABLE CONTAINS ALL FIELDS ON THE MASTER FILE
```

```
119 * REFERENCED BY THE FIELD NAME ID.
120 * ALL IMPORTANT INFORMATION ABOUT THE FIELD IS STORED IN THE TABLE
121 * ENTRY, WHEN A NEW FIELD IS ADDED TO THE RECORD IT MUST BE
122 * ASSIGNED AN ID AND PUT IN THE TABLE
123 *
124 FILE FIELDTBL   TABLE INSTREAM
125   ARG          2        5        A
126   DESC         8        63       A
127  AENDT ACTUAL END DATE           61 04 D
     AMAN  ACTUAL HOURS SO FAR       53 04 P 1 069
     ASTDT ACTUAL START DATE         57 04 D
     EENDT EST END DATE              49 04 D 1
     EMAN  EST MAN HOURS             41 04 P 1
     ESTDT EST START DATE            45 04 D
     LACHR LAST ACTIVITY HRS         69 04 P 1
     LTRDT LAST TRANSACTION DATE     65 04 D
     MGRID MANAGER ID                36 05 X
     NAME  NAME                      11 25 X
     PCTCP PCT COMPLETE              73 02 P 0
     PREN1 PREDECESSOR 1             75 05 X
     PREN2 PREDECESSOR 2             82 05 X
     PREN3 PREDECESSOR 3             89 05 X
     PREN4 PREDECESSOR 4             96 05 X
     PREN5 PREDECESSOR 5            103 05 X
     PREP1 PRE PCT1                  80 02 P 0
     PREP2 PRE PCT2                  87 02 P 0
     PREP3 PRE PCT3                  94 02 P 0
     PREP4 PRE PCT4                 101 02 P 0
     PREP5 PRE PCT5                 108 02 P 0
     PROJ  PROJECT NUMBER            01 05 X
     TASK  TASK NUMBER               06 05 X
    ENDTABLE
128 FILE TRANSTBL TABLE INSTREAM
129  ARG          2       2      A
130  DESC         5       40     A
131  AP 03 ADDED
     AT 04 ADDED
     CP 05 CHANGED
     CT 06 CHANGED
     DP 01 PROJECT DELETED
     DT 02 TASK DELETED
     IT 07 INCREMENTED
    ENDTABLE
132 FILE TRANREG  PRINTER
133 FILE PRJLST   PRINTER
134 FILE UPDRPT   PRINTER
135 FILE TRANERR  PRINTER
136 FILE TRANSIN
137   COPY TRANSWK
138 FILE PRJOUT  FB(110 4400)
139   PROJ-DATA     1      10      A  -
           INDEX PROJ-NDX
140   PROJ-DATE PROJ-DATA 4 P
141   PROJ-PACKED-L4-D1   PROJ-DATA 4  P 1
142   PROJ-PACKED-L4-D0   PROJ-DATA 4  P 0
143   PROJ-PACKED-L2-D0   PROJ-DATA 2  P 0
144   COPY PRJIN
145 *
146 JOB INPUT TRANSIN FINISH GOODBYE
147 *
148 * THIS FIRST JOB EDITS THE TRANSACTIONS AND CREATES AN OUTPUT
149 * FILE OF ALL VALID TRANSACTIONS
150 * INVALID TRANSACTIONS ARE LISTED ON THE ERROR REPORT
151 * AND SKIPPED
152 * VALID TRANSACTIONS HAVE THE TRANS ID CONVERTED TO A VALUE
153 * FROM 01 TO 07.  THIS IS USED TO SORT THE TRANSACTIONS IN
```

```
154 * DELETE, ADD, CHANGE, INCREMENT ORDER WITH A PROJECT/TASK
155 * IF MORE THAN A SPECIFIED NUMBER OF ERRORS OCCUR, THE
156 * ENTIRE RUN IS ABORTED, OTHERWISE THE JOB CONTINUES
157 *
158   SEARCH TRANSTBL WITH TRANSIN:TRANS-ID GIVING TRANS-ATTRIBUTES
159 *
160   IF NOT TRANSTBL
161     MSG = 'INVALID TRANSACTION CODE'
162     PERFORM TRANS-ERR
163   END-IF
164 *
165   IF TRANSIN:TRANS-ID-OP = PROJ-OP  -
        AND TRANSIN:TRANS-TASK NOT SPACES
166     MSG = 'TASK NUMBER MUST BE SPACES FOR PROJECT TRANSACTION'
167     PERFORM TRANS-ERR
168   END-IF
169 *
170   IF TRANSIN:TRANS-ID-CMD NE DELETE-CMD       . * ONLY DELETES HAVE
172     PERFORM FIELD-EDIT                        . * NO FIELD DATA
174   END-IF
175 *
176   IF TRANS-ERR-FLAG NE 'Y'
177     TRANSIN:TRANS-ID-CNV  =  TRANS-TBL-CNV
178     PUT TRANSWK  FROM TRANSIN
179   ELSE
180     TRANS-ERR-FLAG = ' '
181   END-IF
182 *
183 FIELD-EDIT. PROC
185 *
186   SEARCH FIELDTBL WITH TRANSIN:TRANS-FIELD GIVING FIELD-ATTRIBUTES
187 *
188   IF NOT FIELDTBL
189     MSG = 'INVALID FIELD ID'
190     GOTO FIELD-ERR
191   END-IF
192 *
193   IF FIELD-TYPE = 'A'  -
        AND TRANS-DATA NOT ALPHABETIC
194     MSG = 'DATA MUST BE ALPHABETIC'
195     GOTO FIELD-ERR
196   END-IF
197 *
198   IF FIELD-TYPE = 'D'  -
       AND TRANS-DATE NOT NUMERIC
199     MSG = 'DATE MUST BE NUMERIC'
200     GOTO FIELD-ERR
201   END-IF
202   MSG = 'DATA MUST BE NUMERIC'
203   IF FIELD-TYPE = 'P' -
       AND FIELD-LEN = 4 AND TRANS-L7 NOT NUMERIC
204     GOTO FIELD-ERR
205   END-IF
206   IF FIELD-TYPE = 'P' AND FIELD-LEN = 2 -
       AND TRANS-L3 NOT NUMERIC
207     GOTO FIELD-ERR
208   END-IF
209   IF TRANSIN:TRANS-ID-CMD = INC-CMD   -
       AND FIELD-TYPE NE 'N'    -
       AND FIELD-TYPE NE     'P'
210     MSG = 'FIELD CANNOT BE INCREMENTED'
211     GOTO FIELD-ERR
212   ELSE
213     GOTO FIELD-OK
214   END-IF
215 FIELD-ERR
```

```
216   PERFORM TRANS-ERR
217 FIELD-OK
218 END-PROC
219 *
220 GOODBYE. PROC
222 *
223 * FINAL CHECK FOR MAX NUMBER OF ERRORS
224 *
225   DISPLAY TRANERR TRANS-ERR-COUNT ' ERRORS FOUND'
226   IF TRANS-ERR-COUNT  GT  20
227     RETURN-CODE = TRANS-ERR-COUNT
228     STOP EXECUTE
229   END-IF
230 END-PROC
231 *
232 TRANS-ERR. PROC
234     TRANS-ERR-COUNT  =  TRANS-ERR-COUNT  +  1
235     PRINT TRANS-ERR
236     TRANS-ERR-FLAG = 'Y'
237 END-PROC
238 *
239 REPORT TRANS-ERR PRINTER TRANERR LINESIZE 80
240   TITLE 'TRANSACTION ERROR REPORT 1'
241   LINE 1 TRANSIN:TRANS-REC
242   LINE 2  MSG
243 *
244 * THIS JOB SORTS THE EDITED TRANSACTION FILE CREATED IN THE PREVIOUS
245 * JOB.
246 *
247 *
248 SORT TRANSWK TO TRANSWK -
       USING (TRANS-PROJ TRANS-TASK TRANS-ID-CNV)
249 *
250 JOB INPUT (PRJIN    KEY (PROJ-NO TASK-NO) -
                TRANSWK  KEY (TRANS-PROJ TRANS-TASK)) -
       START HELLO
251 *
252 * MAINLINE CODE
253 * DEPENDING ON WHAT RECORDS ARE AVAILABLE
254 * PERFORM THE ROUTINE RESPONSIBLE
255 *
256   IF TRANSWK
257     PRINT TRANS-REG
258     PERFORM EDIT-TRANS
259   END-IF
260 *
261   IF PRJIN AND NOT TRANSWK
262     PERFORM MASTER-WITHOUT-TRANS
263     GOTO JOB
264   END-IF
265 *
266   IF PRJIN AND TRANSWK
267     PERFORM MASTER-WITH-TRANS
268     GOTO JOB
269   END-IF
270 *
271   IF TRANSWK AND NOT PRJIN
272     PERFORM TRANS-WITHOUT-MASTER
273     GOTO JOB
274   END-IF
275 *
276 EDIT-TRANS. PROC
278 *
279 * CHECK FOR DUPLICATE TRANACTIONS
280 *
281   IF LAST-TRANS-KEY  =  TRANS-KEY
```

```
282     MSG = 'DUPLICATE TRANSACTION DROPPED'
283     PRINT ERROR-REPORT
284     GOTO JOB
285   ELSE
286     LAST-TRANS-KEY  =  TRANS-KEY
287   END-IF
288 END-PROC
289 *
290 FIELD-INC. PROC
292 *
293 * INCREMENT SELECTED FIELD WITH TRANSACTION DATA
294 *   SAVE INCREMENT DATA IF FIELD-SAVE IS SPECIFIED
295 *
296 * DEPENDING ON THE LENGTH AND DECIMAL PLACES FOR THE FIELD
297 * ADD THE TRANSACTION DATA TO THE CORRECT OUTPUT FIELD
298 * IF AN INVALID TYPE IS IN THE TABLE THEN ABORT
299 *
300 * IF ANY NEW PACKED FIELD LENGTHS/DECIMAL PLACES ARE ADDED
301 * THIS SECTION OF THE PROGRAM MUST BE UPDATED
302 *
303   IF FIELD-TYPE NE 'P'
304     GOTO INC-ERROR
305   END-IF
306   IF FIELD-LEN = 4  AND FIELD-DEC = 1
307     PROJ-PACKED-L4-D1 = PROJ-PACKED-L4-D1 + TRANS-L7-D1
308     GOTO INC-TEST-FOR-SAVE
309   END-IF
310   IF FIELD-LEN = 4  AND FIELD-DEC = 0
311     PROJ-PACKED-L4-D0 = PROJ-PACKED-L4-D0 + TRANS-L7-D0
312     GOTO INC-TEST-FOR-SAVE
313   END-IF
314   IF FIELD-LEN = 2  AND FIELD-DEC = 0
315     PROJ-PACKED-L2-D0 = PROJ-PACKED-L2-D0 + TRANS-L3-D0
316     GOTO INC-TEST-FOR-SAVE
317   END-IF
318 INC-ERROR
319   DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
        FIELD TYPE--CALL PROGRAMMING SUPPORT'
320   RETURN-CODE = 16
321   STOP EXECUTE
322 INC-TEST-FOR-SAVE
323   IF FIELD-SAVE NOT SPACES
324     PRJOUT:PROJ-NDX = FIELD-SAVE - 1
325     PERFORM FIELD-UPD
326   END-IF
327 END-PROC
328 *
329 FIELD-UPD. PROC
331 *
332 *  UPDATE OF ADD DATA TO SELECTED FIELD
333 *
334   IF FIELD-TYPE = 'A' 'X'
335     MOVE TRANS-DATA FIELD-LEN  TO PRJOUT:PROJ-DATA FIELD-LEN
336     GOTO UPD-END
337   END-IF
338   IF FIELD-TYPE = 'D'
339     PROJ-DATE = TRANS-DATE
340     GOTO UPD-END
341   END-IF
342   IF FIELD-TYPE = 'P'
343     PERFORM MOVE-PACKED-FIELD
344     GOTO UPD-END
345   END-IF
346   DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
        FIELD TYPE--CALL PROGRAMMING SUPPORT'
347   RETURN-CODE = 16
```

```
348   STOP EXECUTE
349 UPD-END
350 END-PROC
351 HELLO. PROC
353   %GETDATE SYSTEM-DATE            . * GET SYSTEM DATE FOR DATE STAMP
354 *
355 *  GET THE CURRENT DATE AND PUT INTO USER FIELD LESS SLASHES
356 *
357   DEFINE GETDATE-DATE           W        8  A
358   DEFINE GETDATE-FIRST6 GETDATE-DATE     6  N
359   DEFINE GETDATE-LAST5  GETDATE-DATE +3  5  A
360   DEFINE GETDATE-LAST6  GETDATE-DATE +2  6  A
361   DEFINE GETDATE-LAST3  GETDATE-DATE +5  3  A
362   DEFINE GETDATE-LAST2  GETDATE-DATE +6  2  A
363   GETDATE-DATE  = SYSDATE             . * MOVE ALL 8
365   GETDATE-LAST3 = GETDATE-LAST2       . * SHIFT LEFT OVER NEXT /
367   GETDATE-LAST6 = GETDATE-LAST5       . * SHIFT LEFT OVER FIRST /
369   SYSTEM-DATE = GETDATE-FIRST6         . * MOVE TO USER FIELD
372 END-PROC
373 *
374 INIT-NEW-RECORD. PROC
376 *
377 * INITIALIZE RECORD TO ZEROS AND SPACES
378 *
379   PROJ-REC(PRJOUT)  =   ' '
380   PROJ-TASK(PRJOUT) = TRANS-PROJ-TASK
381   MOVE ZERO TO  PRJOUT:ACT-MAN-HRS-SO-FAR    -
                   PRJOUT:ACT-END-DATE          -
                   PRJOUT:ACT-START-DATE        -
                   PRJOUT:EST-MAN-HRS           -
                   PRJOUT:EST-END-DATE          -
                   PRJOUT:EST-START-DATE        -
                   PRJOUT:LAST-TRANSACTION-DATE    -
                   PRJOUT:LAST-ACTIVITY-HRS     -
                   PRJOUT:PCT-COMPLETE          -
                   PRJOUT:PRE-PCT1              -
                   PRJOUT:PRE-PCT2              -
                   PRJOUT:PRE-PCT3              -
                   PRJOUT:PRE-PCT4              -
                   PRJOUT:PRE-PCT5
382   MASTER-WAITING EQ 'Y'
383 END-PROC
384 *
385 MASTER-DELETE. PROC
387 *
388   MASTER-WAITING = ' '                . * GET RID OF ACTIVE MASTER
390   LAST-TRANS-ID = TRANS-ID
391   SEARCH TRANSTBL WITH LAST-TRANS-ID -
       GIVING TRANS-ATTRIBUTES
392   FIELD-DATA = ' '
393   CHANGE-MSG = TRANS-MSG
394   FIELD-ID   = ' '
395   PRJOUT:PROJ-TASK = PROJ-TASK
396   PRINT  CHANGES-REPORT
397   DELETED-PROJ = TRANS-PROJ           . * SAVE WHAT WAS DELETED
399   DELETED-TASK = TRANS-TASK           . * FOR FUTURE ADDS/CHANGES
401   PRJOUT:PROJ-TASK = ' '              . * NULL OUT HERE
403 END-PROC
404 *
405 MASTER-WITH-TRANS. PROC
407 *
408   IF FIRST-DUP TRANSWK OR NOT DUPLICATE TRANSWK
409     PRJOUT:PROJ-REC     =   PRJIN:PROJ-REC
410     MASTER-WAITING = 'Y'
411   END-IF
412 *
```

```
413   IF TRANS-ID-CMD = DELETE-CMD
414     PERFORM MASTER-DELETE
415     GOTO MAST-TRANS-DELETE
416   END-IF
417 *
418   IF  TRANS-ID-CMD = ADD-CMD             . * MAY BE ERROR
420     PERFORM VALIDATE-ADD-WITH-MASTER     . * GO CHECK
422     GOTO MAST-TRANS-DONE
423   END-IF
424 *
425   IF MASTER-WAITING NE 'Y'               . * MUST HAVE BEEN DELETED
427     MSG = 'CHANGES NOT VALID TO DELETED RECORD'
428     PRINT ERROR-REPORT
429     GOTO MAST-TRANS-DONE
430   END-IF
431   PERFORM UPDATE-RECORD                  . * GO DO UPDATE
433 *
434 MAST-TRANS-DONE
435   PERFORM TEST-FOR-MASTER-WRITE          . * PUT IF LAST TRANS
437 MAST-TRANS-DELETE
438 END-PROC
439 *
440 MASTER-WITHOUT-TRANS. PROC
442 *
443   IF  PROJ-NO = DELETED-PROJ  -
       AND DELETED-TASK = ' '          . * PRIOR PROJECT DELETE
445     CHANGE-MSG    = 'TASK DELETED'
446     FIELD-DATA = ' '
447     FIELD-ID   = ' '
448     PRINT CHANGES-REPORT
449   ELSE
450     PRJOUT:PROJ-REC = PRJIN:PROJ-REC
451     PERFORM PUT-MASTER
452   END-IF
453 END-PROC
454 *
455 MOVE-PACKED-FIELD. PROC
457 *
458 * DEPENDING ON THE LENGTH AND DECIMAL PLACES FOR THE FIELD
459 * MOVE THE TRANSACTION DATA TO THE CORRECT OUTPUT FIELD
460 * IF AN INVALID TYPE IS IN THE TABLE THEN ABORT
461 *
462 * IF ANY NEW PACKED FIELD LENGTHS/DECIMAL PLACES ARE ADDED
463 * THIS SECTION OF THE PROGRAM MUST BE UPDATED
464 *
465   IF FIELD-LEN = 4  AND FIELD-DEC = 1
466      PROJ-PACKED-L4-D1 = TRANS-L7-D1
467      GOTO PACKED-MOVED
468   END-IF
469   IF FIELD-LEN = 4  AND FIELD-DEC = 0
470      PROJ-PACKED-L4-D0 = TRANS-L7-D0
471      GOTO PACKED-MOVED
472   END-IF
473   IF FIELD-LEN = 2  AND FIELD-DEC = 0
474      PROJ-PACKED-L2-D0 = TRANS-L3-D0
475      GOTO PACKED-MOVED
476   END-IF
477   DISPLAY 'PROGRAMMING ERROR, FIELD TABLE CONTAINS UNAVAILABLE +
       FIELD TYPE--CALL PROGRAMMING SUPPORT'
478   RETURN-CODE = 16
479   STOP EXECUTE
480 PACKED-MOVED
481 END-PROC
482 *
483 PUT-MASTER. PROC
485 *
```

```
486   LAST-PROJ = PRJOUT:PROJ-NO
487   LAST-TASK = PRJOUT:TASK-NO
488   PUT PRJOUT
489   PRJOUT:PROJ-REC   =  ' '
490   MASTER-WAITING =  ' '
491 END-PROC
492 *
493 TRANS-WITHOUT-MASTER. PROC
495 *
496   IF TRANS-ID = ADD-TASK-ID    -
       AND TRANS-PROJ NE LAST-PROJ
497     MSG = 'PROJECT NOT FOUND FOR NEW TASK'
498     PRINT ERROR-REPORT
499     GOTO TRANS-DONE
500   END-IF
501 *
502   IF TRANS-ID-CMD NE  ADD-CMD  -
       AND TRANS-PROJ-TASK NE PRJOUT:PROJ-TASK
503     MSG =  'TRANSACTION DOES NOT HAVE A MATCHING MASTER'
504     PRINT ERROR-REPORT
505     GOTO TRANS-DONE
506   END-IF
507 *
508   IF MASTER-WAITING NE 'Y'
509     PERFORM INIT-NEW-RECORD
510   END-IF
511   PERFORM UPDATE-RECORD
512 TRANS-DONE
513   PERFORM TEST-FOR-MASTER-WRITE
514 END-PROC
515 *
516 UPDATE-RECORD. PROC
518 *
519 *  UPDATE OR ADD DATA TO CURRENT OUTPUT RECORD
520 *
521   SEARCH FIELDTBL WITH  TRANS-FIELD           -
        GIVING FIELD-ATTRIBUTES
522   PRJOUT:PROJ-NDX   =  FIELD-OFFSET  -   1
523   IF TRANS-ID-CMD = INC-CMD
524     PERFORM FIELD-INC
525   ELSE
526     PERFORM FIELD-UPD
527   END-IF
528   SEARCH TRANSTBL WITH TRANS-ID GIVING TRANS-ATTRIBUTES
529   CHANGE-MSG = TRANS-MSG
530   FIELD-DATA = TRANS-DATA
531   FIELD-ID = TRANS-FIELD
532   PRINT CHANGES-REPORT
533 END-PROC
534 *
535 TEST-FOR-MASTER-WRITE. PROC
537 *
538 * TEST IF A MASTER IS WAITING TO WRITTEN AND THIS IS THE LAST
539 * TRANSACTION AGAINST IT
540 *
541   IF MASTER-WAITING NE 'Y'
542     GOTO NO-MASTER
543   END-IF
544   IF LAST-DUP TRANSWK OR NOT DUPLICATE TRANSWK
545     PRJOUT:LAST-TRANSACTION-DATE = SYSTEM-DATE
546     PERFORM PUT-MASTER
547   END-IF
548 NO-MASTER
549 END-PROC
550 *
551 VALIDATE-ADD-WITH-MASTER. PROC
```

```
553 *
554 * THIS ROUTINE VERIFIES THAT THE ADD TRANSACTION FOR A MATCHING
555 * MASTER WAS PRECEDED BY A DELETE TRANSACTION
556 *
557   IF DELETED-PROJ NE PROJ-NO
558     GOTO ADD-TRANS-ERR
559   END-IF
560 *
561   IF   DELETED-TASK NE TRANS-TASK AND DELETED-TASK NE ' '
562     GOTO ADD-TRANS-ERR
563   END-IF
564   IF MASTER-WAITING NE 'Y'
565     PERFORM INIT-NEW-RECORD
566   END-IF
567   PERFORM UPDATE-RECORD
568   GOTO ADD-TRANS-OK
569 ADD-TRANS-ERR
570   MSG = 'ADD TRANS INVALID FOR EXISTING PROJ/TASK'
571   PRINT ERROR-REPORT
572 ADD-TRANS-OK
573 END-PROC
574 *
575 REPORT CHANGES-REPORT PRINTER UPDRPT  LINESIZE 80 SPACE 1
576   CONTROL PRJOUT:PROJ-NO  PRJOUT:TASK-NO
577   TITLE 'CHANGE REGISTER'
578   LINE 1 PRJOUT:PROJ-NO  PRJOUT:TASK-NO  CHANGE-MSG FIELD-ID     -
           FIELD-DATA TALLY
579 *
580 REPORT ERROR-REPORT PRINTER TRANERR  SKIP 1   LINESIZE 80
581   TITLE 'TRANSACTION ERROR REPORT 2'
582   LINE 1 TRANS-REC
583   LINE 2 MSG
584 *
585 REPORT TRANS-REG  PRINTER TRANREG  LINESIZE 80
586   CONTROL
587   TITLE 'LISTING OF EDITED TRANSACTIONS'
588   LINE TRANS-REC
```

Output Reports

```
12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      1


                                      TRANS-REC
APINVSM     EENDT030581
APINVSM     ESTDT030181
APINVSM     MGRIDM0002
APINVSM     NAME INVENTORY VSAM CONVERT
ATINVSMT0001ASTDT030181
ATINVSMT0001EENDT030281
ATINVSMT0001EMAN 0000080
ATINVSMT0001ESTDT030181
ATINVSMT0001MGRIDM0001
ATINVSMT0001NAME DETERMINE SPACE NEEDS
ATINVSMT0001PCTCP010
ITINVSMT0001AMAN 0000020
ATINVSMT0002EENDT030381
ATINVSMT0002EMAN 0000080
ATINVSMT0002ESTDT030281
ATINVSMT0002MGRIDM0002
ATINVSMT0002NAME MAKE JCL CHANGES
ATINVSMT0003EENDT030381
ATINVSMT0003EMAN 0000040
ATINVSMT0003ESTDT030381
ATINVSMT0003MGRIDM0003
ATINVSMT0003NAME DEFINE VSAM SPACE
ATINVSMT0004EENDT030481
ATINVSMT0004EMAN 0000060
ATINVSMT0004ESTDT030481
ATINVSMT0004MGRIDM0002
ATINVSMT0004NAME TEST CHANGES
ATINVSMT0005EENDT030581
ATINVSMT0005EMAN 0000040
ATINVSMT0005ESTDT030481
ATINVSMT0005MGRIDM0002
ATINVSMT0005NAME PUT IN PRODUCTION
APONLST     EENDT062881
APONLST     ESTDT060181
APONLST     MGRIDM2001
APONLST     NAME ONLINE CONVERT STUDY
ATONLSTT0001EENDT061081
ATONLSTT0001EMAN 0000400
ATONLSTT0001ESTDT060181
```

```
12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      2

                                      TRANS-REC
ATONLSTT0001MGRIDM0001
ATONLSTT0001NAME DETERMINE NEEDS
ATONLSTT0002EENDT062181
ATONLSTT0002EMAN 0000800
ATONLSTT0002ESTDT061181
ATONLSTT0002MGRIDM0002
ATONLSTT0002NAME ONLINE MONITOR ANALYSIS
ATONLSTT0003EENDT062481
ATONLSTT0003EMAN 0000150
ATONLSTT0003ESTDT062281
ATONLSTT0003MGRIDM0003
ATONLSTT0003NAME DET. MANPOWER AVAIL.
ATONLSTT0004EENDT062881
ATONLSTT0004EMAN 0000150
ATONLSTT0004ESTDT062581
ATONLSTT0004MGRIDM0003
ATONLSTT0004NAME PREPARE REPORT
APTAXCH     EENDT021481
APTAXCH     ESTDT020181
APTAXCH     MGRIDM0001
APTAXCH     NAME PAYROLL TAX CHANGE
ATTAXCHT0001EENDT020481
ATTAXCHT0001EMAN 0000100
ATTAXCHT0001ESTDT020281
ATTAXCHT0001MGRIDM0001
ATTAXCHT0001NAME ANALYZE CHANGES
ATTAXCHT0002EENDT020681
ATTAXCHT0002EMAN 0000100
ATTAXCHT0002ESTDT020581
ATTAXCHT0002MGRIDM0002
ATTAXCHT0002NAME DETERMINE PROGRAM CHANGES
ATTAXCHT0002PREN1T0001
ATTAXCHT0003EENDT021081
ATTAXCHT0003EMAN 0000200
ATTAXCHT0003ESTDT020681
ATTAXCHT0003MGRIDM0002
ATTAXCHT0003NAME MAKE PROGRAM CHANGES
ATTAXCHT0003PREN1T0002
ATTAXCHT0004EENDT021381
ATTAXCHT0004EMAN 0000100
ATTAXCHT0004ESTDT021081
ATTAXCHT0004MGRIDM0002
```

```
12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      3

                                      TRANS-REC
ATTAXCHT0004NAME TEST CHANGES
ATTAXCHT0004PREN1T0003
ATTAXCHT0005EENDT021481
ATTAXCHT0005EMAN 0000080
ATTAXCHT0005ESTDT021481
ATTAXCHT0005MGRIDM0002
ATTAXCHT0005NAME PUT IN PRODUCTION
ATTAXCHT0005PREN1T0004
```

```
12/03/83                        CHANGE REGISTER                    PAGE      1


   PROJECT  TASK
   NUMBER  NUMBER   CHANGE-MSG    FIELD-ID       FIELD-DATA         TALLY

    INVSM          ADDED          EENDT   030581
                   ADDED          ESTDT   030181
                   ADDED          MGRID   M0002
                   ADDED          NAME    INVENTORY VSAM CONVERT
    INVSM                                                             4

    INVSM  T0001   ADDED          ASTDT   030181
                   ADDED          EENDT   030281
                   ADDED          EMAN    0000080
                   ADDED          ESTDT   030181
                   ADDED          MGRID   M0001
                   ADDED          NAME    DETERMINE SPACE NEEDS
                   ADDED          PCTCP   010
                   INCREMENTED    AMAN    0000020
    INVSM  T0001                                                      8

    INVSM  T0002   ADDED          EENDT   030381
                   ADDED          EMAN    0000080
                   ADDED          ESTDT   030281
                   ADDED          MGRID   M0002
                   ADDED          NAME    MAKE JCL CHANGES
    INVSM  T0002                                                      5

    INVSM  T0003   ADDED          EENDT   030381
                   ADDED          EMAN    0000040
                   ADDED          ESTDT   030381
                   ADDED          MGRID   M0003
                   ADDED          NAME    DEFINE VSAM SPACE
    INVSM  T0003                                                      5

    INVSM  T0004   ADDED          EENDT   030481
                   ADDED          EMAN    0000060
                   ADDED          ESTDT   030481
                   ADDED          MGRID   M0002
                   ADDED          NAME    TEST CHANGES
    INVSM  T0004                                                      5
```

```
12/03/83                          CHANGE REGISTER                        PAGE      2


    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     INVSM  T0005  ADDED           EENDT   030581
                   ADDED           EMAN    0000040
                   ADDED           ESTDT   030481
                   ADDED           MGRID   M0002
                   ADDED           NAME    PUT IN PRODUCTION
     INVSM  T0005                                                        5

     INVSM                                                              32

     ONLST         ADDED           EENDT   062881
                   ADDED           ESTDT   060181
                   ADDED           MGRID   M2001
                   ADDED           NAME    ONLINE CONVERT STUDY
     ONLST                                                               4
     ONLST  T0001  ADDED           EENDT   061081
                   ADDED           EMAN    0000400
                   ADDED           ESTDT   060181
                   ADDED           MGRID   M0001
                   ADDED           NAME    DETERMINE NEEDS
     ONLST  T0001                                                        5

     ONLST  T0002  ADDED           EENDT   062181
                   ADDED           EMAN    0000800
                   ADDED           ESTDT   061181
                   ADDED           MGRID   M0002
                   ADDED           NAME    ONLINE MONITOR ANALYSIS
     ONLST  T0002                                                        5

     ONLST  T0003  ADDED           EENDT   062481
                   ADDED           EMAN    0000150
                   ADDED           ESTDT   062281
                   ADDED           MGRID   M0003
                   ADDED           NAME    DET. MANPOWER AVAIL.
     ONLST  T0003                                                        5
```

```
12/03/83                          CHANGE REGISTER                      PAGE      3


    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     ONLST  T0004  ADDED           EENDT   062881
                   ADDED           EMAN    0000150
                   ADDED           ESTDT   062581
                   ADDED           MGRID   M0003
                   ADDED           NAME    PREPARE REPORT
     ONLST  T0004                                                        5

     ONLST                                                              24

     TAXCH         ADDED           EENDT   021481
                   ADDED           ESTDT   020181
                   ADDED           MGRID   M0001
                   ADDED           NAME    PAYROLL TAX CHANGE
     TAXCH                                                               4

     TAXCH  T0001  ADDED           EENDT   020481
                   ADDED           EMAN    0000100
                   ADDED           ESTDT   020281
                   ADDED           MGRID   M0001
                   ADDED           NAME    ANALYZE CHANGES
     TAXCH  T0001                                                        5

     TAXCH  T0002  ADDED           EENDT   020681
                   ADDED           EMAN    0000100
                   ADDED           ESTDT   020581
                   ADDED           MGRID   M0002
                   ADDED           NAME    DETERMINE PROGRAM CHANGES
                   ADDED           PREN1   T0001
     TAXCH  T0002                                                        6
```

```
12/03/83                          CHANGE REGISTER                        PAGE      4

    PROJECT  TASK
    NUMBER   NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     TAXCH   T0003    ADDED          EENDT    021081
                      ADDED          EMAN     0000200
                      ADDED          ESTDT    020681
                      ADDED          MGRID    M0002
                      ADDED          NAME     MAKE PROGRAM CHANGES
                      ADDED          PREN1    T0002
     TAXCH   T0003                                                         6

     TAXCH   T0004    ADDED          EENDT    021381
                      ADDED          EMAN     0000100
                      ADDED          ESTDT    021081
                      ADDED          MGRID    M0002
                      ADDED          NAME     TEST CHANGES
                      ADDED          PREN1    T0003
     TAXCH   T0004                                                         6

     TAXCH   T0005    ADDED          EENDT    021481
                      ADDED          EMAN     0000080
                      ADDED          ESTDT    021481
                      ADDED          MGRID    M0002
                      ADDED          NAME     PUT IN PRODUCTION
                      ADDED          PREN1    T0004
     TAXCH   T0005                                                         6

     TAXCH                                                                33
```

File Update Reports

Project Status: Example 17.2

```
*** PROJECT MANAGEMENT SYSTEM EXAMPLE 17.2: FILE UPDATE ***

12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      1


                                    TRANS-REC
CTINVSMT0001AENDT030281
CTINVSMT0001PCTCP100
ITINVSMT0001AMAN 0000070
CTINVSMT0002AENDT030481
CTINVSMT0002ASTDT030381
CTINVSMT0002PCTCP100
ITINVSMT0002AMAN 0000072
CTINVSMT0003ASTDT030581
CTINVSMT0003PCTCP090
ITINVSMT0003AMAN 0000042
CTONLSTT0001ASTDT060181
CTONLSTT0001PCTCP030
ITONLSTT0001AMAN 0000100
APRJE       EENDT031082
APRJE       ESTDT010482
APRJE       MGRIDM0004
APRJE       NAME ESTABLISH RJE SITE
ATRJE  T0001EENDT011582
ATRJE  T0001EMAN 0000700
ATRJE  T0001ESTDT010482
ATRJE  T0001MGRIDM0001
ATRJE  T0001NAME DETERMINE USER NEEDS
ATRJE  T0002EENDT011582
ATRJE  T0002EMAN 0000700
ATRJE  T0002ESTDT010482
ATRJE  T0002MGRIDM0002
ATRJE  T0002NAME DETERMINE EQUIP NEEDS
ATRJE  T0003EENDT012082
ATRJE  T0003EMAN 0000100
ATRJE  T0003ESTDT011882
ATRJE  T0003MGRIDM0002
ATRJE  T0003NAME GET TECHNICAL INFO
ATRJE  T0004EENDT012082
ATRJE  T0004EMAN 0000060
ATRJE  T0004ESTDT012082
ATRJE  T0004MGRIDM0002
ATRJE  T0004NAME GET COMM. LINE INFO
ATRJE  T0005EENDT012982
```

```
12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      2

                                        TRANS-REC
ATRJE  T0005EMAN 0000500
ATRJE  T0005ESTDT012182
ATRJE  T0005MGRIDM0003
ATRJE  T0005NAME MATCH EQUIP TO NEEDS
ATRJE  T0006EENDT020182
ATRJE  T0006EMAN 0000060
ATRJE  T0006ESTDT020182
ATRJE  T0006MGRIDM0002
ATRJE  T0006NAME ORDER COMMUNICATIONS LINE
ATRJE  T0007EENDT020182
ATRJE  T0007EMAN 0000020
ATRJE  T0007ESTDT020182
ATRJE  T0007MGRIDM0002
ATRJE  T0007NAME SCHEDULE LINE INSTALL
ATRJE  T0008EENDT020282
ATRJE  T0008EMAN 0000060
ATRJE  T0008ESTDT020282
ATRJE  T0008MGRIDM0002
ATRJE  T0008NAME ORDER COMPUTER EQUIPMENT
ATRJE  T0009EENDT020282
ATRJE  T0009EMAN 0000020
ATRJE  T0009ESTDT020282
ATRJE  T0009MGRIDM0002
ATRJE  T0009NAME SCHEDULE COMPUTER INSTALL
ATRJE  T0010EENDT021282
ATRJE  T0010EMAN 0000500
ATRJE  T0010ESTDT020382
ATRJE  T0010MGRIDM0001
ATRJE  T0010NAME PREPARE TRAINING PLAN
ATRJE  T0011EENDT021282
ATRJE  T0011EMAN 0000040
ATRJE  T0011ESTDT021282
ATRJE  T0011MGRIDM0001
ATRJE  T0011NAME SCHEDULE USER TRAINING
ATRJE  T0012EENDT021782
ATRJE  T0012EMAN 0000200
ATRJE  T0012ESTDT021582
ATRJE  T0012MGRIDM0003
```

```
12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      3


                                    TRANS-REC
ATRJE  T0012NAME PREPARE TEST PLAN
ATRJE  T0013EENDT021882
ATRJE  T0013EMAN 0000040
ATRJE  T0013ESTDT021882
ATRJE  T0013MGRIDM0001
ATRJE  T0013NAME ESTABLISH USER ID'S
ATRJE  T0014EENDT030182
ATRJE  T0014EMAN 0000080
ATRJE  T0014ESTDT030182
ATRJE  T0014MGRIDM0002
ATRJE  T0014NAME SUPERVISE EQUIP. DELIVERY
ATRJE  T0015EENDT030382
ATRJE  T0015EMAN 0000080
ATRJE  T0015ESTDT030382
ATRJE  T0015MGRIDM0003
ATRJE  T0015NAME PERFORM STAND-ALONE TESTS
ATRJE  T0016EENDT030682
ATRJE  T0016EMAN 0000200
ATRJE  T0016ESTDT030482
ATRJE  T0016MGRIDM0003
ATRJE  T0016NAME PERFORM RJE-HOST TESTS
ATRJE  T0017EENDT031082
ATRJE  T0017EMAN 0000240
ATRJE  T0017ESTDT030882
ATRJE  T0017MGRIDM0001
ATRJE  T0017NAME CONDUCT USER TRAINING
CPTAXCH     AENDT021381
CPTAXCH     ASTDT020281
CPTAXCH     PCTCP100
CTTAXCHT0001AENDT020481
CTTAXCHT0001ASTDT020281
CTTAXCHT0001PCTCP100
ITTAXCHT0001AMAN 0000100
CTTAXCHT0002ASTDT020481
CTTAXCHT0002ASTDT020681
CTTAXCHT0002PCTCP100
ITTAXCHT0002AMAN 0000120
CTTAXCHT0003AENDT020981
CTTAXCHT0003ASTDT020681
CTTAXCHT0003PCTCP100
ITTAXCHT0003AMAN 0000180
CTTAXCHT0004AENDT021181


12/03/83                LISTING OF EDITED TRANSACTIONS            PAGE      4


                                    TRANS-REC
CTTAXCHT0004ASTDT021081
CTTAXCHT0004PCTCP100
ITTAXCHT0004AMAN 0000100
CTTAXCHT0005ASTDT021281
CTTAXCHT0005PCTCP090
ITTAXCHT0005AMAN 0000100

    0 ERRORS FOUND
```

```
12/03/83                    TRANSACTION ERROR REPORT 2                PAGE      1


                                      TRANS-REC
CTTAXCHT0002ASTDT020681
DUPLICATE TRANSACTION DROPPED
```

```
12/03/83                           CHANGE REGISTER                     PAGE      1

    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     INVSM  T0001  CHANGED          AENDT   030281
                   CHANGED          PCTCP   100
                   INCREMENTED      AMAN    0000070
     INVSM  T0001                                                          3

     INVSM  T0002  CHANGED          AENDT   030481
                   CHANGED          ASTDT   030381
                   CHANGED          PCTCP   100
                   INCREMENTED      AMAN    0000072
     INVSM  T0002                                                          4

     INVSM  T0003  CHANGED          ASTDT   030581
                   CHANGED          PCTCP   090
                   INCREMENTED      AMAN    0000042
     INVSM  T0003                                                          3

     INVSM                                                                10
     ONLST  T0001  CHANGED          ASTDT   060181
                   CHANGED          PCTCP   030
                   INCREMENTED      AMAN    0000100
     ONLST  T0001                                                          3

     ONLST                                                                 3

     RJE           ADDED            EENDT   031082
                   ADDED            ESTDT   010482
                   ADDED            MGRID   M0004
                   ADDED            NAME    ESTABLISH RJE SITE
     RJE                                                                   4

     RJE    T0001  ADDED            EENDT   011582
                   ADDED            EMAN    0000700
                   ADDED            ESTDT   010482
                   ADDED            MGRID   M0001
                   ADDED            NAME    DETERMINE USER NEEDS
     RJE    T0001                                                          5
```

```
12/03/83                          CHANGE REGISTER                      PAGE      2

    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     RJE    T0002  ADDED           EENDT   011582
                   ADDED           EMAN    0000700
                   ADDED           ESTDT   010482
                   ADDED           MGRID   M0002
                   ADDED           NAME    DETERMINE EQUIP NEEDS
     RJE    T0002                                                        5

     RJE    T0003  ADDED           EENDT   012082
                   ADDED           EMAN    0000100
                   ADDED           ESTDT   011882
                   ADDED           MGRID   M0002
                   ADDED           NAME    GET TECHNICAL INFO
     RJE    T0003                                                        5

     RJE    T0004  ADDED           EENDT   012082
                   ADDED           EMAN    0000060
                   ADDED           ESTDT   012082
                   ADDED           MGRID   M0002
                   ADDED           NAME    GET COMM. LINE INFO
     RJE    T0004                                                        5

     RJE    T0005  ADDED           EENDT   012982
                   ADDED           EMAN    0000500
                   ADDED           ESTDT   012182
                   ADDED           MGRID   M0003
                   ADDED           NAME    MATCH EQUIP TO NEEDS
     RJE    T0005                                                        5

     RJE    T0006  ADDED           EENDT   020182
                   ADDED           EMAN    0000060
                   ADDED           ESTDT   020182
                   ADDED           MGRID   M0002
                   ADDED           NAME    ORDER COMMUNICATIONS LINE
     RJE    T0006                                                        5
```

```
12/03/83                          CHANGE REGISTER                        PAGE      3

    PROJECT  TASK
    NUMBER   NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     RJE     T0007   ADDED          EENDT    020182
                     ADDED          EMAN     0000020
                     ADDED          ESTDT    020182
                     ADDED          MGRID    M0002
                     ADDED          NAME     SCHEDULE LINE INSTALL
     RJE     T0007                                                        5

     RJE     T0008   ADDED          EENDT    020282
                     ADDED          EMAN     0000060
                     ADDED          ESTDT    020282
                     ADDED          MGRID    M0002
                     ADDED          NAME     ORDER COMPUTER EQUIPMENT
     RJE     T0008                                                        5

     RJE     T0009   ADDED          EENDT    020282
                     ADDED          EMAN     0000020
                     ADDED          ESTDT    020282
                     ADDED          MGRID    M0002
                     ADDED          NAME     SCHEDULE COMPUTER INSTALL
     RJE     T0009                                                        5

     RJE     T0010   ADDED          EENDT    021282
                     ADDED          EMAN     0000500
                     ADDED          ESTDT    020382
                     ADDED          MGRID    M0001
                     ADDED          NAME     PREPARE TRAINING PLAN
     RJE     T0010                                                        5

     RJE     T0011   ADDED          EENDT    021282

     RJE     T0011   ADDED          EMAN     0000040
                     ADDED          ESTDT    021282
                     ADDED          MGRID    M0001
                     ADDED          NAME     SCHEDULE USER TRAINING
     RJE     T0011                                                        5
```

```
12/03/83                          CHANGE REGISTER                      PAGE      4

    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     RJE    T0012  ADDED           EENDT   021782
                   ADDED           EMAN    0000200
                   ADDED           ESTDT   021582
                   ADDED           MGRID   M0003
                   ADDED           NAME    PREPARE TEST PLAN
     RJE    T0012                                                        5

     RJE    T0013  ADDED           EENDT   021882
                   ADDED           EMAN    0000040
                   ADDED           ESTDT   021882
                   ADDED           MGRID   M0001
                   ADDED           NAME    ESTABLISH USER ID'S
     RJE    T0013                                                        5

     RJE    T0014  ADDED           EENDT   030182
                   ADDED           EMAN    0000080
                   ADDED           ESTDT   030182
                   ADDED           MGRID   M0002
                   ADDED           NAME    SUPERVISE EQUIP. DELIVERY
     RJE    T0014                                                        5

     RJE    T0015  ADDED           EENDT   030382
                   ADDED           EMAN    0000080
                   ADDED           ESTDT   030382
                   ADDED           MGRID   M0003
                   ADDED           NAME    PERFORM STAND-ALONE TESTS
     RJE    T0015                                                        5
```

```
12/03/83                          CHANGE REGISTER                      PAGE      5

    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     RJE    T0016  ADDED           EENDT   030682
                   ADDED           EMAN    0000200
                   ADDED           ESTDT   030482
                   ADDED           MGRID   M0003
                   ADDED           NAME    PERFORM RJE-HOST TESTS
     RJE    T0016                                                        5

     RJE    T0017  ADDED           EENDT   031082
                   ADDED           EMAN    0000240
                   ADDED           ESTDT   030882
                   ADDED           MGRID   M0001
                   ADDED           NAME    CONDUCT USER TRAINING
     RJE    T0017                                                        5

     RJE                                                               89

     TAXCH          CHANGED         AENDT   021381
     TAXCH          CHANGED         ASTDT   020281
                    CHANGED         PCTCP   100
     TAXCH                                                             3

     TAXCH  T0001   CHANGED         AENDT   020481
                    CHANGED         ASTDT   020281
                    CHANGED         PCTCP   100
                    INCREMENTED     AMAN    0000100
     TAXCH  T0001                                                      4

     TAXCH  T0002   CHANGED         ASTDT   020481
                    CHANGED         PCTCP   100
                    INCREMENTED     AMAN    0000120
     TAXCH  T0002                                                      3
```

```
12/03/83                          CHANGE REGISTER                      PAGE      6

    PROJECT  TASK
    NUMBER  NUMBER   CHANGE-MSG    FIELD-ID        FIELD-DATA          TALLY

     TAXCH  T0003   CHANGED         AENDT   020981
                    CHANGED         ASTDT   020681
                    CHANGED         PCTCP   100
                    INCREMENTED     AMAN    0000180
     TAXCH  T0003                                                      4

     TAXCH  T0004   CHANGED         AENDT   021181
                    CHANGED         ASTDT   021081
                    CHANGED         PCTCP   100
                    INCREMENTED     AMAN    0000100
     TAXCH  T0004                                                      4

     TAXCH  T0005   CHANGED         ASTDT   021281
                    CHANGED         PCTCP   090
                    INCREMENTED     AMAN    0000100
     TAXCH  T0005                                                      3

     TAXCH                                                            21
```

Report Generation

As shown in the report output (shown later), three report programs illustrated in Examples 17.3, 17.4, and 17.5, are provided to give you an idea of what is possible using the Project Management System Master File. The function and operation of each program is documented within the example.

Project Status: Example 17.3

```
 1 *
 2 *  PROJECT MANAGEMENT SYSTEM - EXAMPLE 17.3
 3 *
 4 %PROJLIB
47 *
48 * THIS JOB PRODUCES TWO REPORTS
49 * ONE IS FOR THE PROJECT/TASK MANAGER
50 * THE OTHER IS AN OVER STATUS REPORT FOR EACH PROJECT
51 *
52 JOB
53 *
54   IF TASK-NO NOT SPACES
55     PRINT MANAGER-REPORT
56     PRINT STATUS-REPORT
57   END-IF
58 *
59 *
60 REPORT   STATUS-REPORT    LINESIZE 80    SPACE 1
61   CONTROL  PROJ-NO
62   TITLE    'PROJECT STATUS REPORT'
63   LINE     PROJ-NO TASK-NO MANAGER-ID EST-START-DATE EST-END-DATE -
               EST-MAN-HRS ACT-MAN-HRS-SO-FAR
64 *
65 REPORT   MANAGER-REPORT   LINESIZE 80    SPACE 1
66   SEQUENCE  MANAGER-ID  PROJ-NO  TASK-NO
67   CONTROL   MANAGER-ID NEWPAGE  PROJ-NO
68   TITLE     'PROJECT LIST BY MANAGER'
69   TITLE   3 'MANAGER ID:' -1 MANAGER-ID
70   LINE      PROJ-NO TASK-NO EST-START-DATE EST-END-DATE EST-MAN-HRS
71 *
```

12/03/83 PROJECT STATUS REPORT PAGE 1

PROJECT NUMBER	TASK NUMBER	MANAGER ID	EST START DATE	EST END DATE	EST HOURS	ACTUAL HOURS
INVSM	T0001	M0001	03/01/81	03/02/81	8.0	9.0
	T0002	M0002	03/02/81	03/03/81	8.0	7.2
	T0003	M0003	03/03/81	03/03/81	4.0	4.2
	T0004	M0002	03/04/81	03/04/81	6.0	.0
	T0005	M0002	03/04/81	03/05/81	4.0	.0
INVSM					30.0	20.4
ONLST	T0001	M0001	06/01/81	06/10/81	40.0	10.0
	T0002	M0002	06/11/81	06/21/81	80.0	.0
	T0003	M0003	06/22/81	06/24/81	15.0	.0
	T0004	M0003	06/25/81	06/28/81	15.0	.0
ONLST					150.0	10.0
RJE	T0001	M0001	01/04/82	01/15/82	70.0	.0
	T0002	M0002	01/04/82	01/15/82	70.0	.0
	T0003	M0002	01/18/82	01/20/82	10.0	.0
	T0004	M0002	01/20/82	01/20/82	6.0	.0
	T0005	M0003	01/21/82	01/29/82	50.0	.0
	T0006	M0002	02/01/82	02/01/82	6.0	.0
	T0007	M0002	02/01/82	02/01/82	2.0	.0
	T0008	M0002	02/02/82	02/02/82	6.0	.0
	T0009	M0002	02/02/82	02/02/82	2.0	.0
	T0010	M0001	02/03/82	02/12/82	50.0	.0
	T0011	M0001	02/12/82	02/12/82	4.0	.0
	T0012	M0003	02/15/82	02/17/82	20.0	.0
	T0013	M0001	02/18/82	02/18/82	4.0	.0
	T0014	M0002	03/01/82	03/01/82	8.0	.0
	T0015	M0003	03/03/82	03/03/82	8.0	.0
	T0016	M0003	03/04/82	03/06/82	20.0	.0
	T0017	M0001	03/08/82	03/10/82	24.0	.0
RJE					360.0	.0

12/03/83 PROJECT STATUS REPORT PAGE 2

PROJECT NUMBER	TASK NUMBER	MANAGER ID	EST START DATE	EST END DATE	EST HOURS	ACTUAL HOURS
TAXCH	T0001	M0001	02/02/81	02/04/81	10.0	10.0
	T0002	M0002	02/05/81	02/06/81	10.0	12.0
	T0003	M0002	02/06/81	02/10/81	20.0	18.0
	T0004	M0002	02/10/81	02/13/81	10.0	10.0
	T0005	M0002	02/14/81	02/14/81	8.0	10.0
TAXCH					58.0	60.0

12/03/83 PROJECT LIST BY MANAGER PAGE 1

MANAGER ID:M0001

PROJECT NUMBER	TASK NUMBER	EST START DATE	EST END DATE	EST HOURS
INVSM	T0001	03/01/81	03/02/81	8.0
INVSM				8.0
ONLST	T0001	06/01/81	06/10/81	40.0
ONLST				40.0
RJE	T0001	01/04/82	01/15/82	70.0
	T0010	02/03/82	02/12/82	50.0
	T0011	02/12/82	02/12/82	4.0
	T0013	02/18/82	02/18/82	4.0
	T0017	03/08/82	03/10/82	24.0
RJE				152.0
TAXCH	T0001	02/02/81	02/04/81	10.0
TAXCH				10.0
				210.0

12/03/83 PROJECT LIST BY MANAGER PAGE 2

MANAGER ID:M0002

PROJECT NUMBER	TASK NUMBER	EST START DATE	EST END DATE	EST HOURS
INVSM	T0002	03/02/81	03/03/81	8.0
	T0004	03/04/81	03/04/81	6.0
	T0005	03/04/81	03/05/81	4.0
INVSM				18.0
ONLST	T0002	06/11/81	06/21/81	80.0
ONLST				80.0
RJE	T0002	01/04/82	01/15/82	70.0
	T0003	01/18/82	01/20/82	10.0
	T0004	01/20/82	01/20/82	6.0
	T0006	02/01/82	02/01/82	6.0
	T0007	02/01/82	02/01/82	2.0
	T0008	02/02/82	02/02/82	6.0
	T0009	02/02/82	02/02/82	2.0
	T0014	03/01/82	03/01/82	8.0
RJE				110.0
TAXCH	T0002	02/05/81	02/06/81	10.0
	T0003	02/06/81	02/10/81	20.0
	T0004	02/10/81	02/13/81	10.0
	T0005	02/14/81	02/14/81	8.0
TAXCH				48.0
				256.0

```
12/03/83                    PROJECT LIST BY MANAGER                PAGE      3

                               MANAGER ID:M0003


                                    EST       EST
PROJECT  TASK    START      END        EST
NUMBER  NUMBER   DATE       DATE       HOURS

 INVSM  T0003   03/03/81  03/03/81          4.0
 INVSM                                     4.0

 ONLST  T0003   06/22/81  06/24/81         15.0
        T0004   06/25/81  06/28/81         15.0
 ONLST                                     30.0

 RJE    T0005   01/21/82  01/29/82         50.0
        T0012   02/15/82  02/17/82         20.0
        T0015   03/03/82  03/03/82          8.0
        T0016   03/04/82  03/06/82         20.0
 RJE                                       98.0
                                          132.0

                                         598.0
```

Project Summary

Project Summary: Example 17.4

```
 1 *
 2 * PROJECT MANAGEMENT SYSTEM - EXAMPLE 17.4
 3 *
 4 %PROJLIB
47   PCT  PCT-COMPLETE   2 P 0    . * REDEFINE PCT COMPLETE TO TOTAL
49 *
50 * THIS EXAMPLE IS A SIMPLE SUMMARY REPORT OF ALL PROJECTS
51 *
52 JOB
53 *
54   IF TASK-NO NOT SPACES         . * ONLY GET TASK RECORDS
56     PRINT PROJECT-SUMMARY
57   END-IF
58 *
59 REPORT PROJECT-SUMMARY  LINESIZE 80  SPACE 1   SUMMARY
60   CONTROL   PROJ-NO
61   TITLE     'SUMMARY LISTING OF PROJECTS AND MAN HOURS'
62   HEADING   TALLY   ('NUMBER' 'OF' 'TASKS')
63   LINE    1 PROJ-NO ACT-MAN-HRS-SO-FAR EST-MAN-HRS PCT TALLY
64 *
65   BEFORE-BREAK. PROC
67     PCT = PCT  / TALLY   . * COMPUTE AVE PCT COMPLETE
69   END-PROC
```

```
12/03/83           SUMMARY LISTING OF PROJECTS AND MAN HOURS       PAGE      1

                                                                    NUMBER
              PROJECT    ACTUAL          EST                        OF
              NUMBER      HOURS         HOURS       PCT          TASKS

               INVSM          20.4          30.0      58            5
               ONLST          10.0         150.0       7            4
               RJE              .0         360.0                   17
               TAXCH          60.0          58.0      98            5
                              90.4         598.0      26           31
```

Project Completion: Example 17.5

```
 1 *
 2 * PROJECT MANAGEMENT SYSTEM - EXAMPLE 17.5
 3 *
 4 %PROJLIB
47 *
48 * THIS JOB PRODUCES TWO REPORTS
49 * THE FIRST GIVES THE PERCENT VARIANCE OF ESTIMATED VERSUS ACTUAL
50 * COMPLETION TIMES.
51 *
52 * THE SECOND GIVES THE ESTIMATED TIME TO COMPLETE FOR THE
53 * TASKS THAT HAVE STARTED BUT NOT FINISHED YET
54 *
55 VARIANCE    W      3   P  1
56 HRS-TO-GO   W      3   P  1
57 *
58 JOB
59 *
60   IF TASK-NO SPACES OR ACT-MAN-HRS-SO-FAR = 0
61     GOTO JOB
62   END-IF
63   IF PCT-COMPLETE = 100
64     VARIANCE = ((ACT-MAN-HRS-SO-FAR - EST-MAN-HRS)/ EST-MAN-HRS) * 100
65     PRINT COMPLETE-REPORT
66   ELSE
67     HRS-TO-GO =  (ACT-MAN-HRS-SO-FAR / (PCT-COMPLETE / 100))  -
                - ACT-MAN-HRS-SO-FAR
68     PRINT PREDICTIONS
69   END-IF
70 *
71 *
72 REPORT   COMPLETE-REPORT       LINESIZE 80
73   SEQUENCE   VARIANCE D
74   CONTROL    PROJ-NO
75   TITLE    1 'LISTING OF ALL COMPLETED TASKS'
76   TITLE    2 'WITH THE VARIANCE OF ACTUAL TO ESTIMATED TIME'
77   LINE     1 PROJ-NO  TASK-NO  MANAGER-ID  VARIANCE
78 *
79   BEFORE-BREAK. PROC
81     VARIANCE  = VARIANCE / TALLY
82   END-PROC
83 *
84 REPORT   PREDICTIONS           LINESIZE 80
85   SEQUENCE   HRS-TO-GO  D
86   TITLE      'LISTING OF UNCOMPLETED TASKS AND HOURS TO COMPLETE'
87   LINE       PROJ-NO  TASK-NO  MANAGER-ID  HRS-TO-GO
88 *
```

```
12/03/83                LISTING OF ALL COMPLETED TASKS              PAGE      1
                    WITH THE VARIANCE OF ACTUAL TO ESTIMATED TIME

                     PROJECT    TASK    MANAGER
                     NUMBER    NUMBER     ID       VARIANCE

                      TAXCH    T0002     M0002          20.0
                      TAXCH                              20.0

                      INVSM    T0001     M0001          12.5
                      INVSM                              12.5

                      TAXCH    T0004     M0002            .0
                               T0001     M0001            .0

                      TAXCH                               .0

                      INVSM    T0002     M0002          10.0-
                      INVSM                              10.0-

                      TAXCH    T0003     M0002          10.0-
                      TAXCH                              10.0-

                                                          2.0

------------------------------------------------------------------------------

12/03/83       LISTING OF UNCOMPLETED TASKS AND HOURS TO COMPLETE   PAGE      1

                     PROJECT    TASK    MANAGER
                     NUMBER    NUMBER     ID      HRS-TO-GO

                      ONLST    T0001     M0001       23.3
                      TAXCH    T0005     M0002        1.1
                      INVSM    T0003     M0003         .4
```

Appendix

A

Table of Statements

This appendix contains an alphabetical list and brief description of the subset of CA-Easytrieve Plus statements covered in this *Application Guide*.

Statement Table

Statement	Function
Assignment	Establishes the value of a field.
CONTROL	Identifies control fields used for a control report.
DEFINE	Specifies data fields within a file or within working storage.
DISPLAY	Prints data from specified fields to the system printer file or a named printer file.
DO	Controls repetitive program logic based on the truth value of associated conditional expressions.
ELSE	Identifies the statement(s) to be executed when the condition in an IF statement tests false.
END-DO	Identifies the end of the statements associated with a DO statement.
END-IF	Identifies the end of the statements associated with an IF statement.
END-PROC	Identifies the end of a procedure.
FILE	Describes a file used by your program.
GET	Reads the next sequential record of a file into storage.
GOTO	Passes control to the specified location in the program.
HEADING	Specifies an alternate column heading for a field on a report.
IF	Controls the execution of associated statements depending on the truth value of conditional expressions.

Statement	Function
JOB	Defines an activity that retrieves input files, examines and manipulates data, initiates printed reports, or produces output files.
LINE	Defines the contents of a report line.
PARM	Establishes program-level environment parameters.
PERFORM	Transfers control to a procedure and returns control to the next executable statement after the procedure is executed.
POINT	Locates the position of a specified record in a file.
PRINT	Outputs data to a file for inclusion in a report.
PROC	Identifies the beginning of a procedure.
PUT	Outputs a record to a file.
READ	Provides random access to keyed and relative-record files.
RECORD	Identifies IMS/DLI database segments available for processing.
REPORT	Establishes the type and characteristics of a report.
RETRIEVE	Specifies the segments to be automatically input from an IMS/DLI database.
SEARCH	Provides access to table information.
SELECT	Used in a sort procedure to select individual records for the sort output.
SEQUENCE	Specifies the order of a report.
SORT	Inputs a sequential file and outputs the result of the sort operation onto an output file.
STOP	Terminates an activity.
TITLE	Defines report title items and their position on the title line.
WRITE	Updates or deletes existing records and adds new records in the processing of keyed and relative-record files.

Appendix

B

Cross-References

This appendix presents a cross-reference listing of CA-Easytrieve Plus statements to the specific examples in the following chapters:

- Basic Examples
- Advanced Techniques
- Bank System
- Project Management System

In addition, a second listing is provided that cross-references CA-Easytrieve Plus functions or facilities to the same examples.

Within these two larger groupings, the statements and facilities are grouped by program section; that is, Library, JOB/SORT Activity, REPORT Activity, and so forth. Within the subgroups, the statements and facilities are presented alphabetically.

The examples are numbered by a two-part designator that includes the chapter number. That is, Example 14.10 is the 10th example presented in Chapter 14, Example 15.14 is the 14th example in Chapter 16, and so on. Please refer to the Table of Contents to determine the exact page number on which to locate a specific example.

Cross-Reference List

Cross-reference Statements for LIBRARY

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
COPY	17.01	FILE	
			14.01
DEFINE (EXPLICIT)	15.12		14.02
	15.14		14.03
			14.04
DEFINE (IMPLICIT)			14.05
			14.06
	14.02		14.07
	14.03		14.08
	14.04		14.09
	14.05		14.10
	14.07		14.11
	14.08		14.12
	14.09		14.14
	14.10		14.15
	14.12		14.16
	14.14		14.17
	14.15		15.01
	14.16		15.02
	14.17		15.03
	15.02		15.04
	15.04		15.05
	15.05		15.06
	15.06		15.07
	15.08		15.08
	15.09		15.09
	15.10		15.10
	15.11		15.11
	16.00		16.04
	16.01		17.00
	16.02		17.01
	16.03		
	16.04		
	17.00		
	17.01		
	17.04		
	17.05		

Cross-reference Statements for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
ASSIGNMENT	14.02	GOTO JOB	14.02
	14.03		14.03
	14.05		14.10
	14.07		14.15
	14.09		14.17
	14.10		15.09
	14.12		15.10
	14.14		15.11
	14.15		16.02
	14.16		17.01
	14.17		17.05
	15.02		
	15.05	IF	14.01
	15.06		14.02
	15.08		14.03
	15.10		14.04
	15.11		14.05
	15.14		14.06
	16.01		14.10
	16.02		14.14
	16.03		14.15
	16.04		14.16
	16.05		14.17
	17.01		15.04
	17.05		15.05
			15.06
DEFINE (EXPLICIT)	17.01		15.08
			15.09
DISPLAY	14.08		15.10
	15.05		15.11
	15.07		16.02
	15.09		16.03
	15.11		16.04
	15.12		17.01
	15.14		17.03
	16.02		17.04
	17.01		17.05

Cross-reference Statements for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
DO WHILE	15.09	JOB	16.01
	15.14		16.02
	16.02		16.03
	16.03		16.04
			16.05
GOTO	16.02		17.01
	17.01		17.03
			17.04
JOB			17.05
	14.01		
	14.02	MOVE	14.08
	14.03		15.14
	14.04		16.02
	14.05		
	14.06	MOVE LIKE	14.08
	14.07		
	14.08	PERFORM	14.15
	14.09		14.16
	14.10		15.05
	14.11		15.14
	14.12		16.02
	14.14		16.04
	14.15		17.01
	14.16		
	14.17	POINT	15.09
	15.01		16.02
	15.02		16.03
	15.03		16.04
	15.04		16.05
	15.05		
	15.06		
	15.07		
	15.08		
	15.09		
	15.10		
	15.11		
	15.12		
	15.14		

Cross-reference Statements for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
PRINT		PROC	14.08
	14.01		14.15
	14.02		14.16
	14.03		15.04
	14.04		15.05
	14.05		15.07
	14.06		15.14
	14.07		16.02
	14.09		16.03
	14.10		16.04
	14.11		16.05
	14.12		17.01
	14.14		
	14.15	PUT	14.08
	14.16		14.14
	14.17		14.16
	15.01		14.17
	15.02		15.05
	15.03		15.07
	15.04		15.11
	15.05		16.01
	15.06		17.01
	15.08		
	15.09	READ	15.08
	15.10		15.11
	15.11		16.02
	16.03		
	16.04	SEARCH	14.09
	16.05		14.10
	17.01		14.16
	17.03		16.02
	17.04		16.03
	17.05		17.01

Cross-reference Statements for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER
SELECT	15.04
SORT	15.04
	15.05
	15.07
	17.01
STOP	15.08
	15.10
	15.12
	15.14
	16.01
STOP EXECUTE	17.01
WRITE	15.08
	15.10
	15.11
	16.02

Cross-reference Statements for REPORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
CONTROL	14.02	LINE	
	14.04		14.01
	14.05		14.02
	14.07		14.04
	14.09		14.05
	14.10		14.06
	14.11		14.07
	14.12		14.09
	14.15		14.10
	14.17		14.11
	15.01		14.12
	15.02		14.14
	15.03		14.15
	15.04		14.16
	15.06		14.17
	15.09		15.01
	16.03		15.02
	16.04		15.03
	16.05		15.04
	17.01		15.05
	17.03		15.06
	17.04		15.08
	17.05		15.09
			15.11
DISPLAY	14.11		16.03
	15.01		16.05
	15.03		17.01
			17.03
HEADING			17.04
	14.09		17.05
	14.10		
	14.17	MOVE	14.07
	15.01		
	15.02		
	16.03		
	17.04		

Cross-reference Statements for REPORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
PROC	14.02	REPORT	16.05
	14.05		17.01
	14.07		17.03
	14.09		17.04
	14.11		17.05
	14.12		
	14.17	SEQUENCE	14.02
	15.01		14.03
	15.03		14.04
	17.04		14.05
	17.05		14.06
			14.07
REPORT			14.09
	14.01		14.10
	14.02		14.11
	14.03		14.12
	14.04		14.15
	14.05		15.02
	14.06		15.03
	14.07		15.06
	14.09		15.08
	14.10		15.10
	14.11		16.03
	14.12		16.04
	14.14		16.05
	14.15		17.03
	14.16		17.05
	14.17		
	15.01	SUM	14.17
	15.02		
	15.03		
	15.04		
	15.05		
	15.06		
	15.08		
	15.09		
	15.10		
	15.11		
	16.03		
	16.04		

Cross-reference Statements for REPORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
TITLE			
	14.01		
	14.02		
	14.05		
	14.06		
	14.07		
	14.09		
	14.10		
	14.11		
	14.12		
	14.14		
	14.15		
	14.16		
	14.17		
	15.01		
	15.02		
	15.03		
	15.04		
	15.05		
	15.06		
	15.08		
	15.09		
	15.10		
	15.11		
	16.03		
	16.05		
	17.01		
	17.03		
	17.04		
	17.05		

Cross-reference Facilities for MACROS

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
KEYWORD		POSITIONAL	
SUBSTITUTION	16.00	SUBSTITUTION	15.12
			15.14
MACRO			17.00
DEFINITION			
	15.12		
	15.14		
	16.00		
	17.00		

Cross-reference Facilities for LIBRARY

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
ARRAY	16.00	MACRO CALL	
	16.02		14.01
	17.00		14.02
	17.01		14.03
			14.04
FIELD			14.05
REDEFINITION			14.06
	14.04		14.07
	14.16		14.09
	14.17		14.10
	15.12		14.11
	15.14		14.12
	16.02		14.14
	17.00		14.15
	17.01		14.16
	17.04		14.17
			15.01
HEADING			15.02
			15.03
	14.02		15.04
	14.03		15.05
	14.05		15.07
	14.07		15.08
	14.09		15.10
	14.10		15.11
	14.12		16.01
	14.15		16.02
	14.16		16.03
	15.02		16.04
	15.05		16.05
	15.10		17.01
	16.00		17.03
	17.00		17.04
	17.01		17.05
INDEXING	15.14	MASK	
	16.00		
	16.02		14.03
	17.00		14.16
	17.01		14.17
			17.00
			17.01

Cross-reference Facilities for LIBRARY

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
INITIAL VALUE	14.07	VSAM KSDS	15.07
	16.00		15.08
	16.02		15.09
	17.01		15.11
			16.00
PRINTER FILE	16.04		
	17.01	VSAM PATH	15.09
S-FIELD	14.05	W-FIELD	14.02
	14.07		14.03
	14.09		14.05
	14.12		14.07
	14.17		14.09
			14.12
TABLE -			14.14
EXTERNAL	14.10		14.15
			14.16
TABLE -			14.17
INSTREAM	14.09		15.02
	14.16		15.06
	16.00		15.10
	16.02		15.12
	17.01		15.14
			16.00
VIRTUAL FILE	15.02		16.02
	15.04		16.03
	15.05		16.04
	15.07		17.01
	15.11		17.05
	17.01		
VSAM ESDS	15.07		
	15.10		

Cross-reference Facilities for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER
BEFORE SORT PROCEDURE	15.04
BLANK FIELDS	14.08
	15.06
FILE REFORMAT	14.08
FILE UPDATING	14.14
	14.16
	14.17
	15.05
	15.08
	15.10
	15.11
	16.02
	17.01
FILE-STATUS	15.09
	15.11
	16.02
FINISH PROCEDURE	14.08
	15.07
	16.02
	17.01
INDEXING	16.02
	16.03
	17.01
MACRO CALL	14.17
	15.12
	15.14
	16.04
	17.01
MULTIPLE INPUTS	15.05
	17.01
MULTIPLE JOBS	15.02
	15.04
	15.05
	15.07
	15.08
	15.10
	15.11
	17.01
NESTED DO WHILE	16.02
NESTED PROCEDURES	16.02
	17.01
NULL INPUT	15.12
	15.14
	16.01
	16.02
PERCENT CALCULATION	14.03
	14.16
	15.05
PROCEDURE	14.15
	14.16
	15.05
	15.14
	16.02
	16.04
	17.01
RECORD-COUNT	15.07
ROUNDING	14.02
	14.03

Cross-reference Facilities for JOB/SORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
SORT	15.04	VSAM PATH	15.09
	15.05		
	15.07	ZERO FIELDS	14.08
	17.01		17.01
START PROCEDURE	16.02		
	16.03		
	16.04		
	16.05		
	17.01		
VARIANCE			
CALCULATION	17.05		
VSAM FILE INPUT	15.08		
	15.10		
	16.03		
	16.04		
	16.05		
VSAM			
FILE LOAD	15.07		
	16.01		
VSAM			
FILE UPDATE	15.08		
	15.10		
	15.11		
	16.02		

Cross-reference Facilities for REPORT

STATEMENT/ FACILITY	EXAMPLE NUMBER
BAR GRAPH	14.07
BEFORE-BREAK	14.02
	14.05
	14.07
	14.09
	14.11
	14.12
	14.17
	15.01
	17.04
	17.05
BLANK FIELDS	14.07
CONTROL REPORT	14.02
	14.04
	14.10
	14.11
	14.12
	14.15
	14.17
	15.02
	15.03
	15.04
	15.06
	15.09
	16.03
	16.04
	17.01
	17.03
	17.05
DETAIL REPORT	14.01
	14.06
	14.10
	14.14
	14.16
	15.04
	15.05
	15.06
	15.08
	15.10
	15.11
	17.01
ENDPAGE	15.03
FORM LETTER	14.03
	16.04
FORM REPORT	14.17
LEVEL	15.01
MAILING LABELS	14.03
	14.04
	16.04
MULTIPLE REPORTS	14.02
	14.03
	14.05
	14.10
	14.12
	14.16
	14.17
	15.04
	15.06
	16.04
	17.01
	17.03
	17.05

Cross-reference Facilities for REPORT

STATEMENT/ FACILITY	EXAMPLE NUMBER	STATEMENT/ FACILITY	EXAMPLE NUMBER
PERCENT CALCULATION	14.02	SUMMARY REPORT	14.02
	14.05		14.05
	14.07		14.07
	14.09		14.09
	14.12		14.10
	17.04		14.12
			15.01
PRINTER FILE	16.04		15.02
	17.01		15.04
			16.05
PROCEDURE	14.02		17.04
	14.05		
	14.07	TALLY	14.02
	14.09		14.05
	14.11		14.07
	14.12		14.09
	14.17		14.10
	15.01		15.01
	15.03		15.02
	17.04		16.03
	17.05		17.04
			17.05
RATIO CALCULATION	14.09	TERMINATION	15.03
RECORD-COUNT	14.08	TITLE VARIABLES	14.17
			17.03
RETURN-CODE	17.01		
ROUNDING	14.02		
	14.05		
	14.07		
	14.09		
	14.12		
SUMMARY FILE	15.02		

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W