SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS REVISION

Operating System/3 (OS/3)

ESCORT

User Guide/Programmer Reference

UP-8855 Rev. 1

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) ESCORT User Guide/Programmer Reference", UP-8855 Rev. 1.

This revision describes the following ESCORT features for release 8.0:

- password commands;
- 44-character file names for tutorial mode and for program mode;
- screen menu method;
- additional error messages;
- saving a session workfile at session termination;
- recovering a saved session workfile at initialization of another session;
- additional program, job and structure commands; and
- additional tutorial mode capabilities.

All other changes are corrections applicable to ESCORT prior to the 8.0 release.

Destruction Notice: If you are going to OS/3 release 8.0, use this revision and destroy all previous copies. If you are not going to OS/3 release 8.0, retain the copy you are now using and store this revision for future use.

Copies of UP-8855 and UP-8855–A will be available for 6 months after the release of 8.0. Should you need additional copies of this edition, you should order them within 90 days of the release of 8.0. When ordering the previous edition of a manual, be sure to identify the exact revision and update packages desired and indicate that they are needed to support an earlier release.

Additional copies may be ordered by your local Sperry Univac representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists A00, A17, B00, B17, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U, 76, and 76U (Cover and 295 pages)	Library Memo for UP-8855 Rev. 1
		RELEASE DATE: September, 1982

ESCORT

OS/3

User Guide/
Programmer Reference

H

SPERRY UNIVAC

UP-8855 Rev. 1

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PAGE STATUS SUMMARY

ISSUE: UP-8855 Rev. 1
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Preface	1 thru 4	
Contents	1 thru 10	
PART 1	Title Page	
1	1 thru 15	
2	1 thru 5	
PART 2	Title Page	
3	1 thru 33	
PART 3	Title Page	
4	1 thru 59	
PART 4	Title Page	
5	1 thru 24	
6	1 thru 4 4a, 4b 5 thru 10 10a thru 10d 11 thru 23	
7	1 thru 41	
8	1 thru 4	
9	1 thru 8	
Appendix A	1 thru 3	
Appendix B	1	
Appendix C	1	
Appendix D	1 thru 6	
Appendix E	1 thru 18	
Appendix F	1, 2	
Appendix G	1 thru 7	
Glossary	1 thru 4	
Index	1 thru 14	
User Comment Sheet		

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This manual is one of a series that instructs and guides you in the use of the SPERRY UNIVAC Operating System/3 (OS/3). It describes the purpose and use of the ESCORT programming language.

The manual has four parts:

- Part 1 deals with the general concepts of ESCORT and how to call ESCORT.
- Part 2 deals with tutorial mode, which aids the beginning user.
- Part 3 deals with program mode and the various commands and statements it includes.
- Part 4 deals with ESCORT programming considerations and includes relevant appendixes and a glossary.

The parts contain the following:

- PART 1. WHERE TO BEGIN
  - Section 1. Introduction to ESCORT

    Describes the operating modes and the basic language concepts of ESCORT.
  - Section 2. Logging on the System and Calling ESCORT

    Describes how to gain entry into OS/3 and call ESCORT.

- PART 2. CONCEPTS OF TUTORIAL MODE

  - Section 3. Interactive Tutorial Mode

    Describes how to use the tutorial mode of operation and gives sample sessions that help describe this learning tool.

- PART 3. CONCEPTS OF PROGRAM MODE

  - Section 4. Interactive Program Mode

    Describes the program mode of operation (direct entry and screen menu methods) and gives sample sessions for the most commonly used and the required screen menus.

- PART 4. SPECIFICS OF PROGRAM MODE

  - Section 5. ESCORT Commands

    Describes the types of commands and their functions.

  - Section 6. ESCORT Files and Structures

    Describes the two different types of files and their access methods, how to build a file structure, and how to compose structure statements.

  - Section 7. Programs and Program Statements

    Describes the different program statements and their associated clauses.

  - Section 8. Jobs

    Describes jobs and how to enter and manipulate them with ESCORT commands.

  - Section 9. How to Create Customized Input/Output Forms

    Tells how to create customized forms for input and output.

  - Appendix A. Program Statement Formats

    Lists the complete format of each program statement.

- Appendix B. Reserved Words

  Lists reserved words and explains their use.

- Appendix C. Function Key Use

  Lists the function keys and their actions under the processing modes in ESCORT.

- Appendix D. Programming Considerations

  Explains some special conventions used with ESCORT.

- Appendix E. Error Messages

  Lists error messages by category.

- Appendix F. SYSGEN Requirements and System Considerations

  Describes system generation requirements and system recommendations for efficient ESCORT operation.

- Appendix G. Custom Formatting Using Structures

  Gives an example of custom formatting using structures you design to suit your output.

This manual includes brief explanations of relevant OS/3 system concepts but, for the most part, assumes you are familiar with system concepts and operations. If you are not, refer to the appropriate OS/3 manual.

The current versions of the following manuals are helpful to the ESCORT user:

System 80

- System 80 operations handbook, UP-8859
- Consolidated data management concepts and facilities, UP-8825
- Interactive services commands and facilities user guide/programmer reference, UP-8845
- Screen format services concepts and facilities, UP-8802
- Job control user guide, UP-8065
- System installation user guide/programmer reference, UP-8859

- File cataloging concepts and facilities, UP-8860

Series 90 Systems

- 90/25 operations handbook, UP-8511
- 90/30, 90/40 operations handbook, UP-8072
- System installation user guide/programmer reference, UP-8074
- Consolidated data management concepts and facilities, UP-8825
- Interactive services commands and facilities user guide/programmer reference, UP-8845
- Screen format services concepts and facilities, UP-8802
- Job control user guide, UP-8065

Contents

PART 4. SPECIFICS OF PROGRAM MODE

APPENDIXES

PART
1

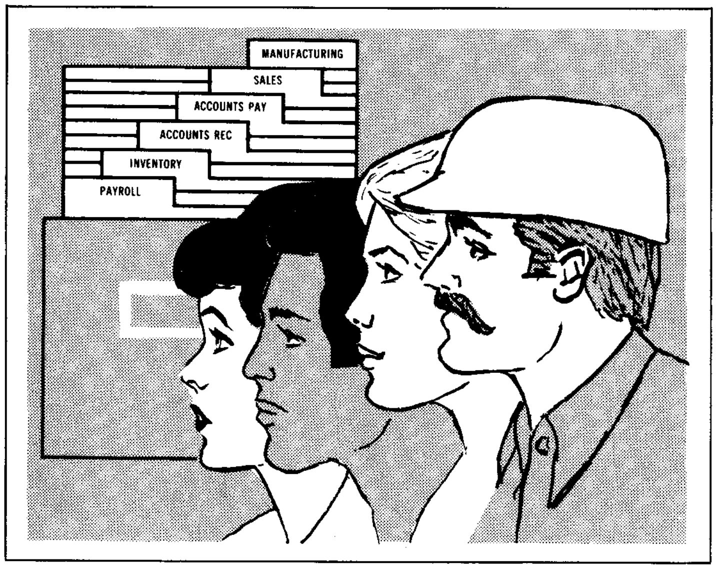
WHERE TO BEGIN

1. Introduction to ESCORT

1.1. AN EASY-TO-USE LANGUAGE

Purpose

ESCORT is an easy-to-use programming language that helps you solve common business problems.

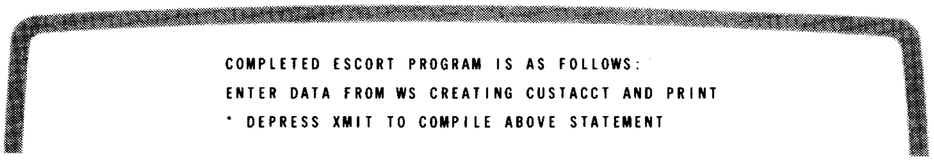


No programming experience needed

It's different from most programming languages because you don't have to be a programmer to use it. With ESCORT, you can write (or let ESCORT create) programs to handle payroll, personnel files, inventory, etc.

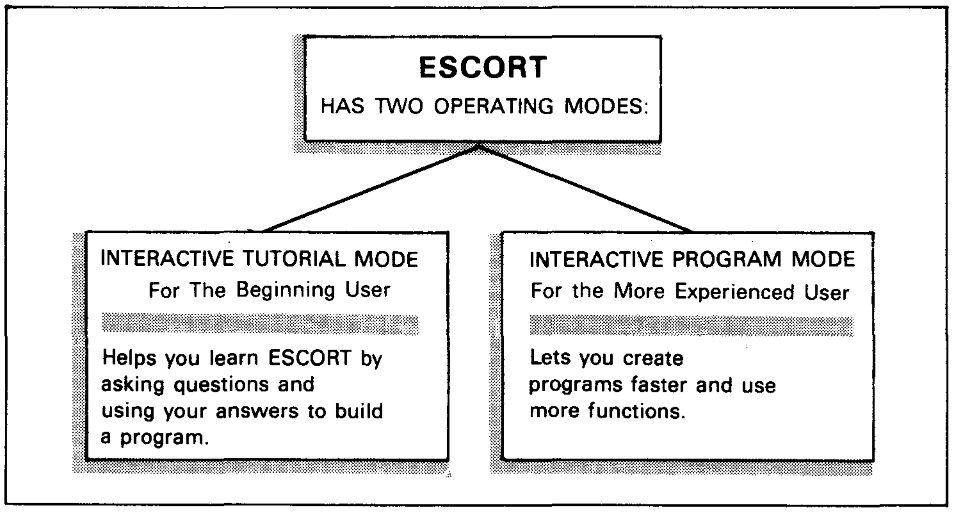
Program appearance

ESCORT programs don't look like typical computer programs; they look more like sentences. They're written in English, and you can write and understand them easily. This ESCORT screen display is much like the ones you'll see as you use this interactive language:



```
COMPLETED ESCORT PROGRAM IS AS FOLLOWS:
ENTER DATA FROM WS CREATING CUSTACCT AND PRINT
* DEPRESS XMIT TO COMPILE ABOVE STATEMENT
```

1.2. MODES OF OPERATION



ESCORT
HAS TWO OPERATING MODES:

INTERACTIVE TUTORIAL MODE
For The Beginning User

Helps you learn ESCORT by asking questions and using your answers to build a program.

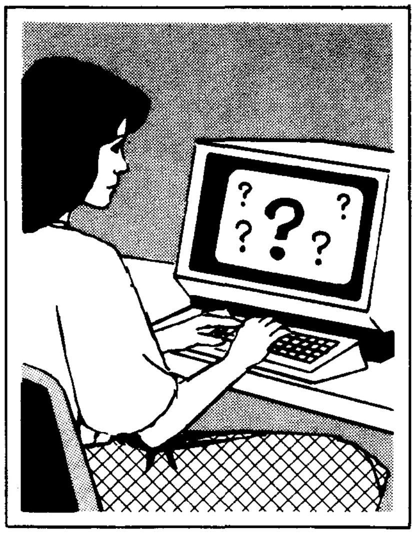
INTERACTIVE PROGRAM MODE
For the More Experienced User

Lets you create programs faster and use more functions.

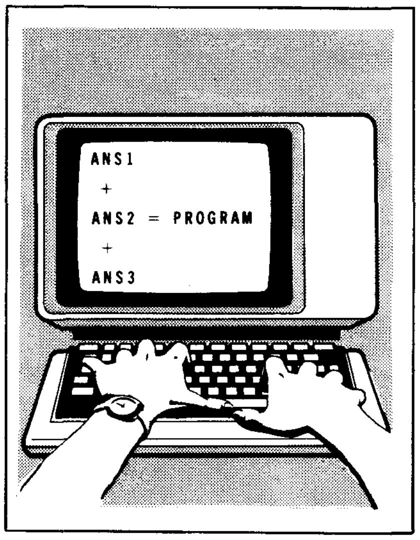
Interactive Tutorial Mode

Intended user

Interactive tutorial mode, which we'll call tutorial mode, is a learning tool for those who don't know much about programming or are just learning ESCORT.



ESCORT displays a series of questions on the workstation screen.

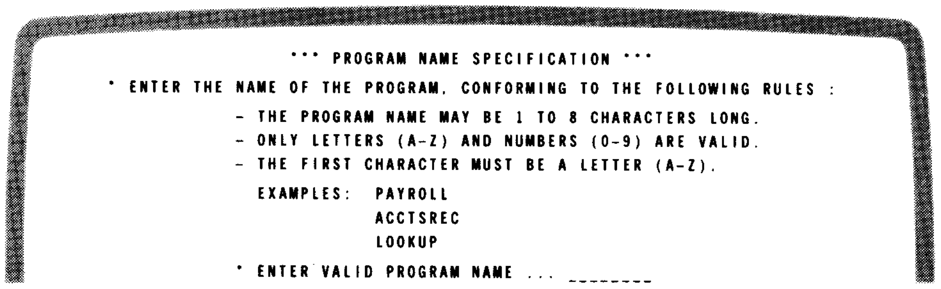


ESCORT uses your answers to create a program.

This question-and-answer technique lets you create simple programs while you learn how ESCORT works. Respond to the questions by:

Method of operation

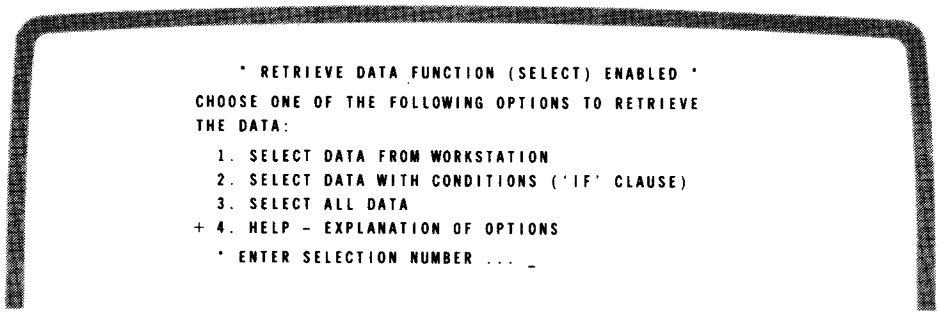
Filling in the blanks:



```
                    *** PROGRAM NAME SPECIFICATION ***
* ENTER THE NAME OF THE PROGRAM, CONFORMING TO THE FOLLOWING RULES :
           - THE PROGRAM NAME MAY BE 1 TO 8 CHARACTERS LONG.
           - ONLY LETTERS (A-Z) AND NUMBERS (0-9) ARE VALID.
           - THE FIRST CHARACTER MUST BE A LETTER (A-Z).
             EXAMPLES:  PAYROLL
                        ACCTSREC
                        LOOKUP
           * ENTER VALID PROGRAM NAME ... ________
```

OR

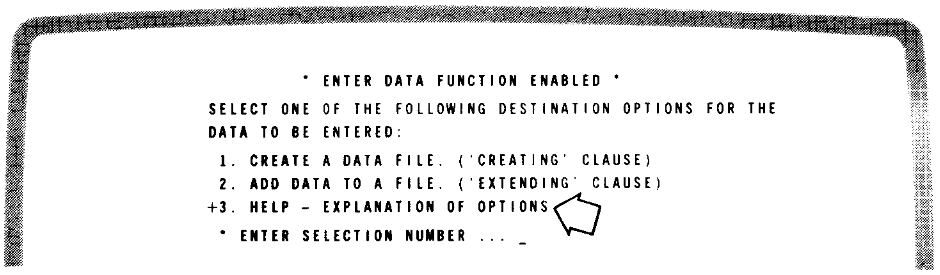
Choosing from a list of options:



```
   * RETRIEVE DATA FUNCTION (SELECT) ENABLED *
 CHOOSE ONE OF THE FOLLOWING OPTIONS TO RETRIEVE
 THE DATA:
   1. SELECT DATA FROM WORKSTATION
   2. SELECT DATA WITH CONDITIONS ('IF' CLAUSE)
   3. SELECT ALL DATA
 + 4. HELP - EXPLANATION OF OPTIONS
   * ENTER SELECTION NUMBER ... _
```

HELP

To help you when you don't understand a function, many screen displays include a choice called HELP, which provides an explanation.



```
            * ENTER DATA FUNCTION ENABLED *
SELECT ONE OF THE FOLLOWING DESTINATION OPTIONS FOR THE
DATA TO BE ENTERED:
 1. CREATE A DATA FILE. ('CREATING' CLAUSE)
 2. ADD DATA TO A FILE. ('EXTENDING' CLAUSE)
+3. HELP - EXPLANATION OF OPTIONS
 * ENTER SELECTION NUMBER ... _
```

Limitations

While tutorial mode is effective in introducing you to ESCORT and teaching you to create simple programs, it is slow and has few capabilities. Once you know the terminology of the language and feel comfortable with it, you'll use program mode.

We'll tell you all about tutorial mode in Section 3.

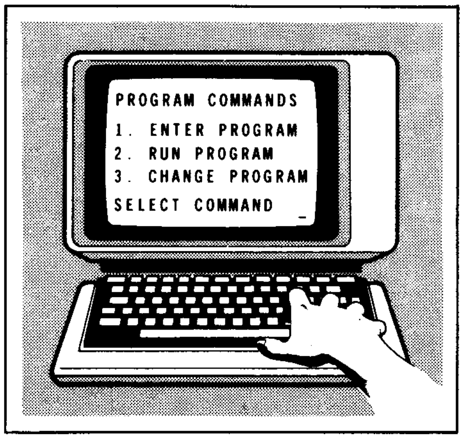
Interactive Program Mode

Intended user

As you become more familiar with ESCORT, you'll want to use interactive program mode, which we'll call program mode. It's faster than tutorial mode and provides full ESCORT capability.

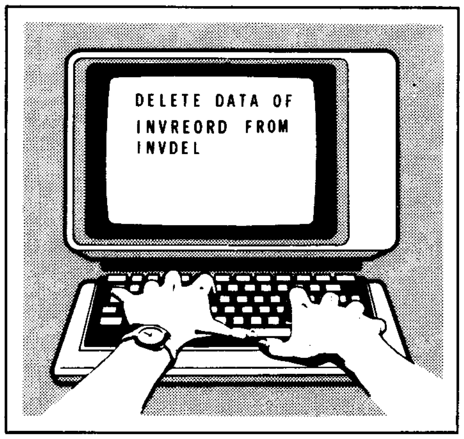
Program mode lets you create programs two ways:

Methods of operation



```
PROGRAM COMMANDS
1. ENTER PROGRAM
2. RUN PROGRAM
3. CHANGE PROGRAM
SELECT COMMAND _
```

Through a series of screen menus

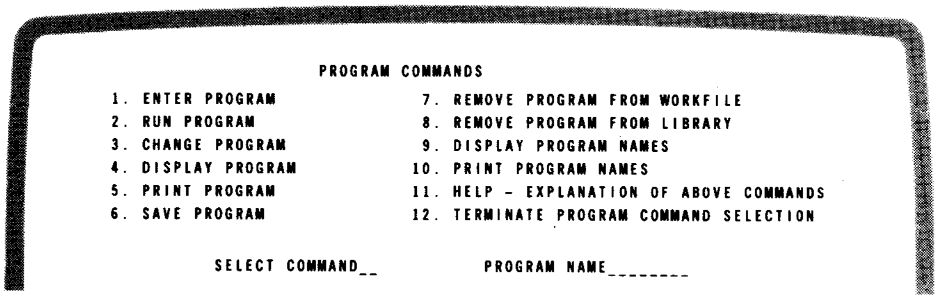


```
DELETE DATA OF
INVREORD FROM
INVDEL
```

Through direct entry of program statements

Screen menu method

The screen menu method is similar to tutorial mode but not as elementary. First, choose a command from a menu displayed on the workstation:



```
                         PROGRAM COMMANDS
1. ENTER PROGRAM                     7. REMOVE PROGRAM FROM WORKFILE
2. RUN PROGRAM                       8. REMOVE PROGRAM FROM LIBRARY
3. CHANGE PROGRAM                    9. DISPLAY PROGRAM NAMES
4. DISPLAY PROGRAM                  10. PRINT PROGRAM NAMES
5. PRINT PROGRAM                    11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE PROGRAM                     12. TERMINATE PROGRAM COMMAND SELECTION

          SELECT COMMAND__            PROGRAM NAME________
```

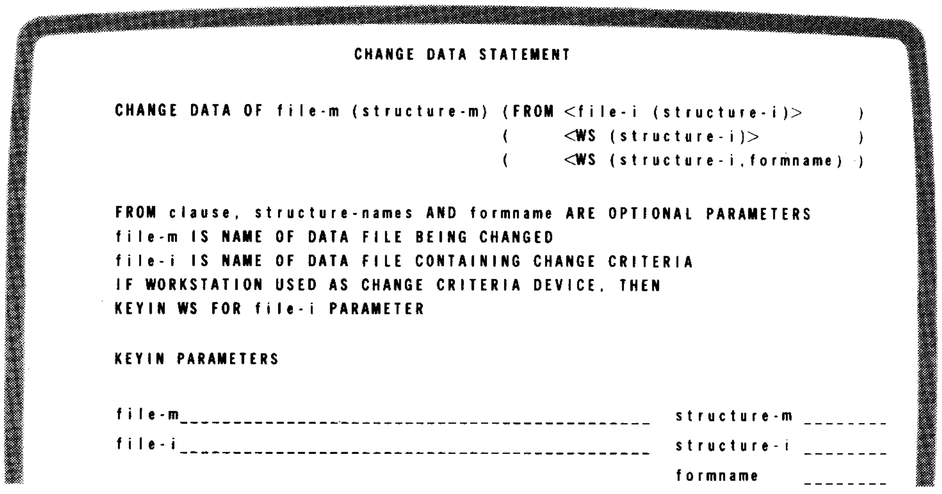
Then based on your choice, ESCORT takes you from one screen display to the next until you complete your function.

Advantages of screen menu method

This makes "writing" programs easier because all you do is key in (that is, type in) such things as file and structure names, device names, and conditional clauses. In this way, you can create complex programs without having to memorize statement syntax and organization. With this method, you have full ESCORT capability, but it's somewhat slower than the direct entry method.

Typical screen

A typical screen you might see when using the screen menu method is:

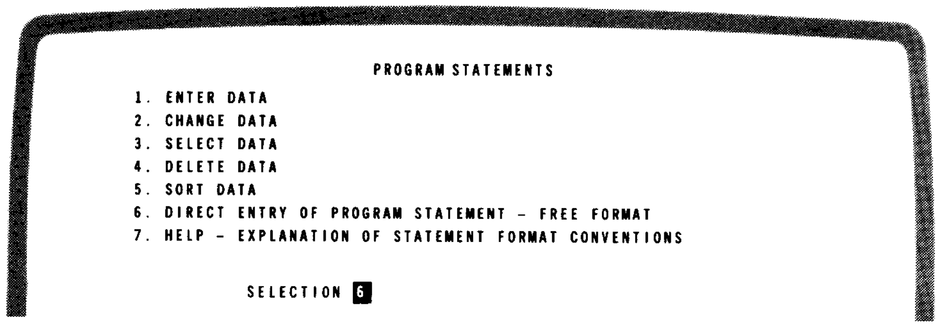


```
                         CHANGE DATA STATEMENT

CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>       )
                                    (      <WS (structure-i)>          )
                                    (      <WS (structure-i,formname)  )

FROM clause, structure-names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING CHANGED
file-i IS NAME OF DATA FILE CONTAINING CHANGE CRITERIA
IF WORKSTATION USED AS CHANGE CRITERIA DEVICE, THEN
KEYIN WS FOR file-i PARAMETER

KEYIN PARAMETERS

file-m___________________________________________  structure-m ________
file-i___________________________________________  structure-i ________
                                                   formname    ________
```

When you're comfortable enough with ESCORT, you'll want to use the direct entry method.

Direct entry method

Select the direct entry method by choosing 6, DIRECT ENTRY OF PROGRAM STATEMENT on the PROGRAM STATEMENTS menu:



```
                         PROGRAM STATEMENTS
     1. ENTER DATA
     2. CHANGE DATA
     3. SELECT DATA
     4. DELETE DATA
     5. SORT DATA
     6. DIRECT ENTRY OF PROGRAM STATEMENT - FREE FORMAT
     7. HELP - EXPLANATION OF STATEMENT FORMAT CONVENTIONS

              SELECTION 6
```

NOTE:

How keyins are indicated

Throughout this manual, we'll use reverse print to show the keyins you make on screens.

Entering program statements directly

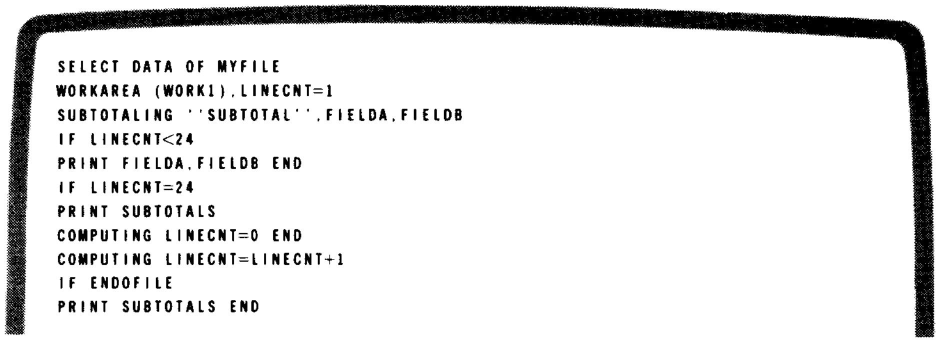
When you press the **XMIT** key, a blank screen appears and you key in program statements directly.

Advantages of direct entry method

The direct entry method is faster than the screen menu method only if you're familiar enough with ESCORT not to need guidance from menus, because you're responsible for statement syntax and organization.

Using the direct entry method, you might key in a program like this:

Example of direct entry method



```
SELECT DATA OF MYFILE
WORKAREA (WORK1),LINECNT=1
SUBTOTALING ''SUBTOTAL'',FIELDA,FIELDB
IF LINECNT<24
PRINT FIELDA,FIELDB END
IF LINECNT=24
PRINT SUBTOTALS
COMPUTING LINECNT=0 END
COMPUTING LINECNT=LINECNT+1
IF ENDOFILE
PRINT SUBTOTALS END
```

Section 4 gives programming examples using both the screen menu method and the direct entry method.

Now that you have an idea of how ESCORT operates, you're ready to learn some of the terms and concepts you'll need to know to use it effectively.

1.3. CONCEPTS OF ESCORT

Before you move into the following sections of the manual, you'll need to know a few things about the ESCORT language. First, we'll describe the parts that comprise it; then we'll show how these parts relate to one another.

The ESCORT language consists of five parts that work together so you can perform all ESCORT functions:

Basic parts of ESCORT language

- Commands
- Statements
- Programs
- Structures
- Jobs

Let's look at each of these parts briefly.

Commands defined

Commands tell ESCORT what operation you want to perform on your structures, programs, jobs, and data file passwords. Before using any other part of ESCORT, you must use a command. We'll expand on commands in 1.4 and Section 5.

Structure statements defined

Program statements defined

Statements are of two types: *structure* statements and *program* statements. Structure statements define individual fields within your data record. One or more structure statements make up a structure. Program statements are combinations of ESCORT words that name the operations to be performed on your data files. Program statements are often supplemented by clauses, which are discussed in Section 7. We'll show how statements are used in 1.4.

Structure defined

A structure is a collection of structure statements, which tell ESCORT what the records in your data files will look like. The structure statements define individual fields in your record, and the structure defines the entire record. Every file must have a structure. We describe structures in 1.4 and Section 6.

Program defined

A program is a collection of program statements and associated clauses that describe the operations to be performed on your data files. You can reference up to 12 files in any ESCORT program. In tutorial mode and screen menu program mode, ESCORT creates programs from your responses. In direct entry program mode, you write the programs. We'll tell more about programs in 1.4 and Section 7.

Job defined

A job is a group of related programs that are executed together. For example, in a payroll job, one program calculates the hours worked, one calculates pay and taxes, and the third updates a permanent payroll file. We'll tell more about jobs in Section 8.

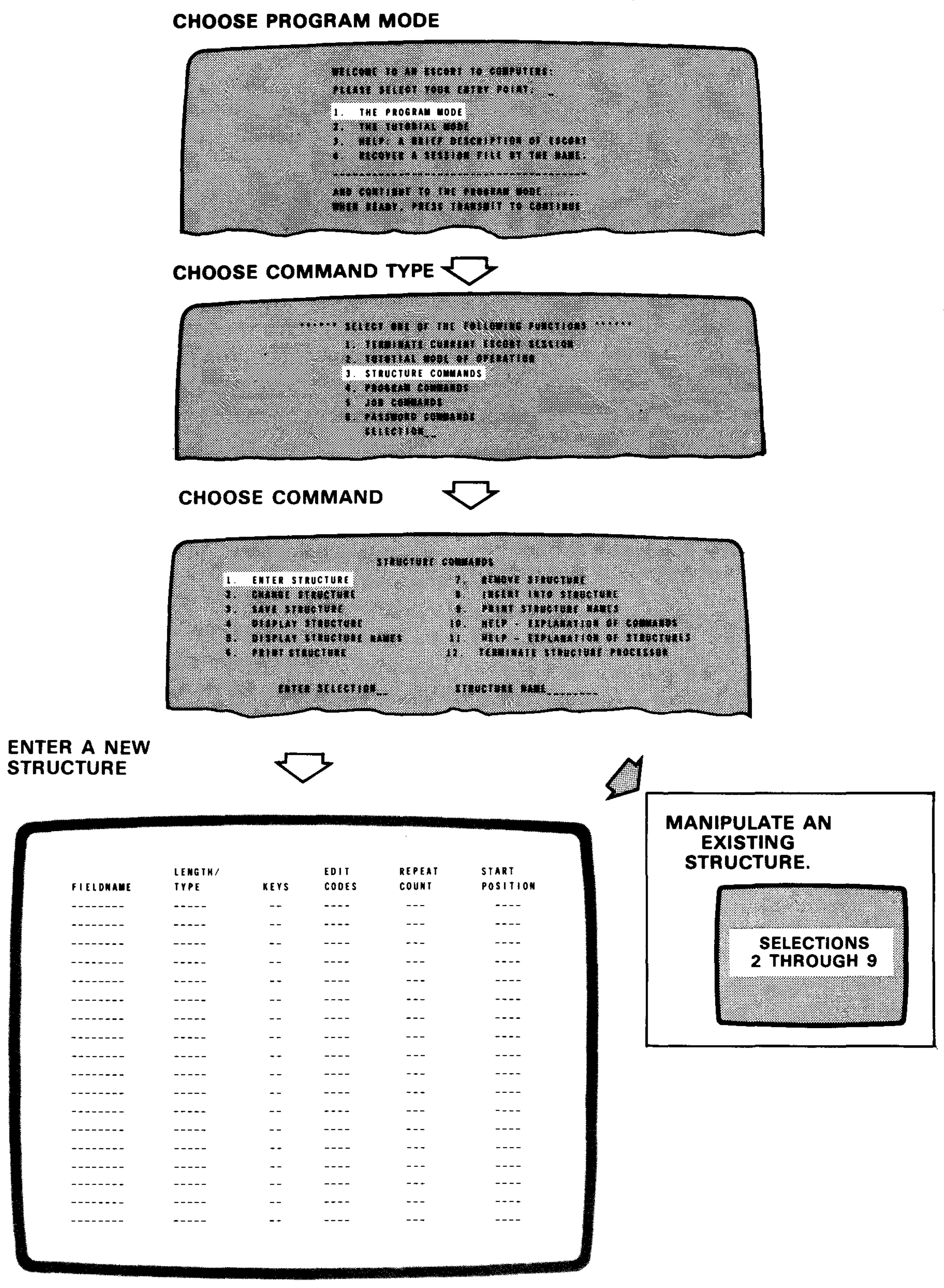
Now you're ready to see how these parts fit together.

1.4. HOW ESCORT WORKS

This subsection, which illustrates the concepts presented in 1.3, gives you an understanding of what ESCORT does to give you your program.

Program Mode

Look at the following diagram, Figure 1-1. This gives an idea of the choices you make that lead to development of your program in program mode.



```
WELCOME TO AN ESCORT TO COMPUTERS:
PLEASE SELECT YOUR ENTRY POINT: _

1.  THE PROGRAM MODE
2.  THE TUTORIAL MODE
3.  HELP: A BRIEF DESCRIPTION OF ESCORT
4.  RECOVER A SESSION FILE BY THE NAME.
-----------------------------------------
AND CONTINUE TO THE PROGRAM MODE......
WHEN READY, PRESS TRANSMIT TO CONTINUE
```

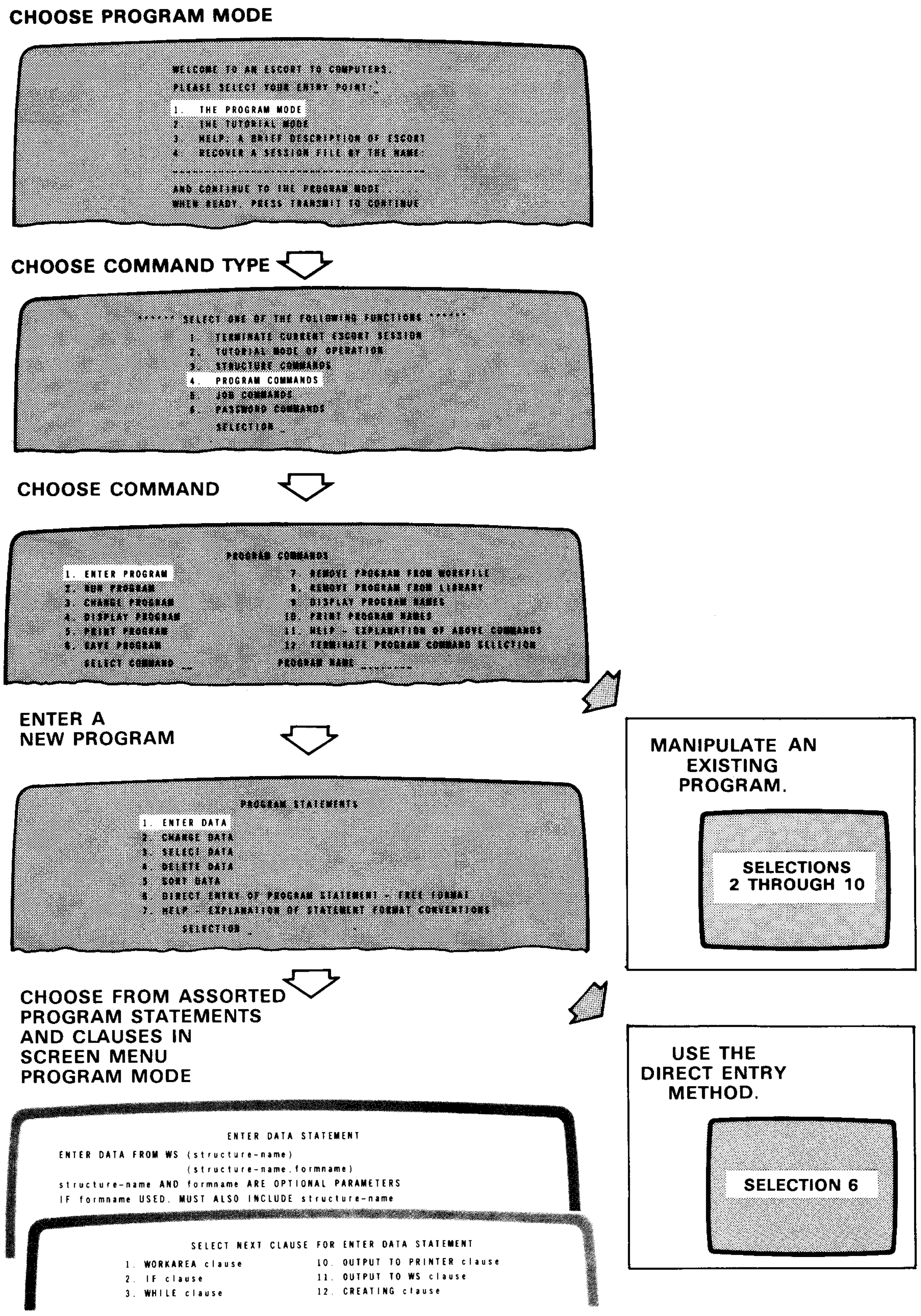
```
****** SELECT ONE OF THE FOLLOWING FUNCTIONS ******
       1. TERMINATE CURRENT ESCORT SESSION
       2. TUTORIAL MODE OF OPERATION
       3. STRUCTURE COMMANDS
       4. PROGRAM COMMANDS
       5. JOB COMMANDS
       6. PASSWORD COMMANDS
          SELECTION__
```

```
                        STRUCTURE COMMANDS
1.  ENTER STRUCTURE                 7.  REMOVE STRUCTURE
2.  CHANGE STRUCTURE                8.  INSERT INTO STRUCTURE
3.  SAVE STRUCTURE                  9.  PRINT STRUCTURE NAMES
4.  DISPLAY STRUCTURE              10.  HELP - EXPLANATION OF COMMANDS
5.  DISPLAY STRUCTURE NAMES        11.  HELP - EXPLANATION OF STRUCTURES
6.  PRINT STRUCTURE                12.  TERMINATE STRUCTURE PROCESSOR

       ENTER SELECTION__            STRUCTURE NAME________
```

```
           LENGTH/            EDIT     REPEAT    START
FIELDNAME  TYPE       KEYS    CODES    COUNT     POSITION
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
---------  -----      --      ----     ---       ----
```

a. Structure commands

Figure 1-1. Program Development in Program Mode (Part 1 of 3)


CHOOSE PROGRAM MODE
CHOOSE COMMAND TYPE
CHOOSE COMMAND
ENTER A NEW PROGRAM
MANIPULATE AN EXISTING PROGRAM.
SELECTIONS 2 THROUGH 10
CHOOSE FROM ASSORTED PROGRAM STATEMENTS AND CLAUSES IN SCREEN MENU PROGRAM MODE
USE THE DIRECT ENTRY METHOD.
SELECTION 6

```
WELCOME TO AN ESCORT TO COMPUTERS.
PLEASE SELECT YOUR ENTRY POINT:_
1.  THE PROGRAM MODE
2.  THE TUTORIAL MODE
3.  HELP: A BRIEF DESCRIPTION OF ESCORT
4.  RECOVER A SESSION FILE BY THE NAME:
----------------------------------------
AND CONTINUE TO THE PROGRAM MODE .....
WHEN READY, PRESS TRANSMIT TO CONTINUE
```

```
***** SELECT ONE OF THE FOLLOWING FUNCTIONS *****
      1.  TERMINATE CURRENT ESCORT SESSION
      2.  TUTORIAL MODE OF OPERATION
      3.  STRUCTURE COMMANDS
      4.  PROGRAM COMMANDS
      5.  JOB COMMANDS
      6.  PASSWORD COMMANDS
          SELECTION _
```

```
                     PROGRAM COMMANDS
1. ENTER PROGRAM                 7. REMOVE PROGRAM FROM WORKFILE
2. RUN PROGRAM                   8. REMOVE PROGRAM FROM LIBRARY
3. CHANGE PROGRAM                9. DISPLAY PROGRAM NAMES
4. DISPLAY PROGRAM              10. PRINT PROGRAM NAMES
5. PRINT PROGRAM                11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE PROGRAM                 12. TERMINATE PROGRAM COMMAND SELECTION
   SELECT COMMAND __            PROGRAM NAME ________
```

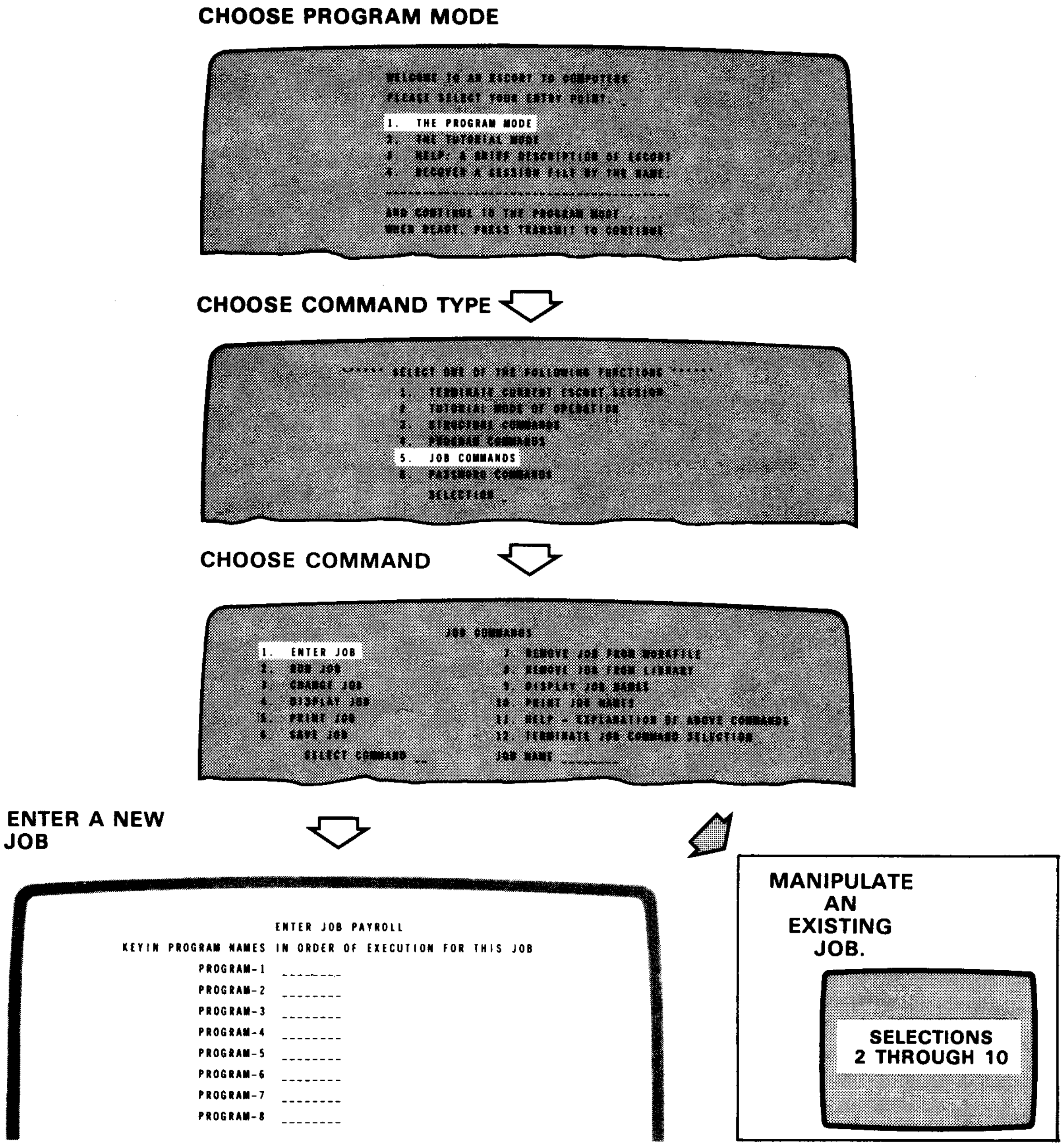
```
              PROGRAM STATEMENTS
1. ENTER DATA
2. CHANGE DATA
3. SELECT DATA
4. DELETE DATA
5. SORT DATA
6. DIRECT ENTRY OF PROGRAM STATEMENT - FREE FORMAT
7. HELP - EXPLANATION OF STATEMENT FORMAT CONVENTIONS
     SELECTION _
```

```
                     ENTER DATA STATEMENT
ENTER DATA FROM WS (structure-name)
                   (structure-name.formname)
structure-name AND formname ARE OPTIONAL PARAMETERS
IF formname USED, MUST ALSO INCLUDE structure-name
```

```
          SELECT NEXT CLAUSE FOR ENTER DATA STATEMENT
   1. WORKAREA clause          10. OUTPUT TO PRINTER clause
   2. IF clause                11. OUTPUT TO WS clause
   3. WHILE clause             12. CREATING clause
```

b. Program commands

Figure 1-1. Program Development in Program Mode (Part 2 of 3)



```
WELCOME TO AN ESCORT TO COMPUTERS
PLEASE SELECT YOUR ENTRY POINT. .

1.  THE PROGRAM MODE
2.  THE TUTORIAL MODE
3.  HELP: A BRIEF DESCRIPTION OF ESCORT
4.  RECOVER A SESSION FILE BY THE NAME:
-----------------------------------------
AND CONTINUE IN THE PROGRAM MODE . . . .
WHEN READY, PRESS TRANSMIT TO CONTINUE
```

```
****** SELECT ONE OF THE FOLLOWING FUNCTIONS ******
   1.  TERMINATE CURRENT ESCORT SESSION
   2.  TUTORIAL MODE OF OPERATION
   3.  STRUCTURE COMMANDS
   4.  PROGRAM COMMANDS
   5.  JOB COMMANDS
   6.  PASSWORD COMMANDS
       SELECTION _
```

```
                      JOB COMMANDS
1.  ENTER JOB                  7.  REMOVE JOB FROM WORKFILE
2.  RUN JOB                    8.  REMOVE JOB FROM LIBRARY
3.  CHANGE JOB                 9.  DISPLAY JOB NAMES
4.  DISPLAY JOB               10.  PRINT JOB NAMES
5.  PRINT JOB                 11.  HELP - EXPLANATION OF ABOVE COMMANDS
6.  SAVE JOB                  12.  TERMINATE JOB COMMAND SELECTION
      SELECT COMMAND __       JOB NAME ________
```

```
                    ENTER JOB PAYROLL
KEYIN PROGRAM NAMES IN ORDER OF EXECUTION FOR THIS JOB
          PROGRAM-1  ________
          PROGRAM-2  ________
          PROGRAM-3  ________
          PROGRAM-4  ________
          PROGRAM-5  ________
          PROGRAM-6  ________
          PROGRAM-7  ________
          PROGRAM-8  ________
```

c. Job commands

Figure 1-1. Program Development in Program Mode (Part 3 of 3)

Once you choose program mode, you must choose a command.

Depending on what command type you choose, you can process structures, programs, jobs, or data file passwords.

Result of choosing structure commands

When you choose structure commands, you have the choice of entering a new structure by using structure statements or manipulating an existing structure with one of the other commands.

Result of choosing program commands

When you choose program commands, you enter a new program or manipulate an existing program. If you choose to enter a new program, you get the PROGRAM STATEMENTS screen and choose the program statement that fits the program you want to write. Then the appropriate sequence of program mode screens appears to complete your program.

From this PROGRAM STATEMENTS screen, you can also choose to enter program statements directly onto a blank screen.

Result of choosing job commands

When you choose job commands, you can enter a new job or manipulate an existing job by using one of the other commands.

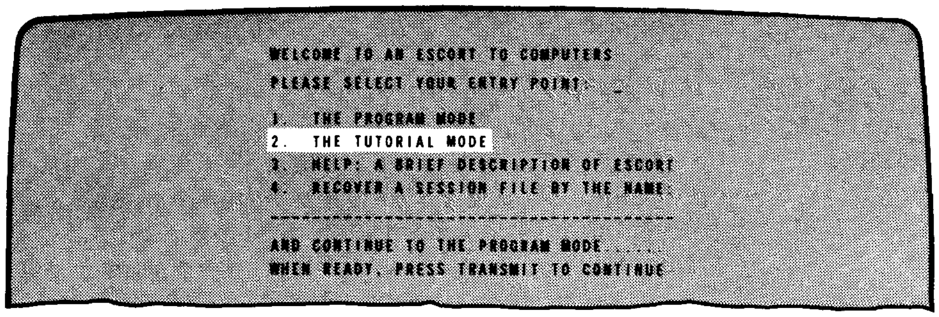
Result of choosing password commands

When you choose password commands, you may enter any or all of the passwords required for the data files named in the programs you plan to execute during this session. You may also manipulate passwords previously entered during the session. Password processing by ESCORT is session oriented. Nothing relating to passwords is retained at the end of the session.

Tutorial Mode

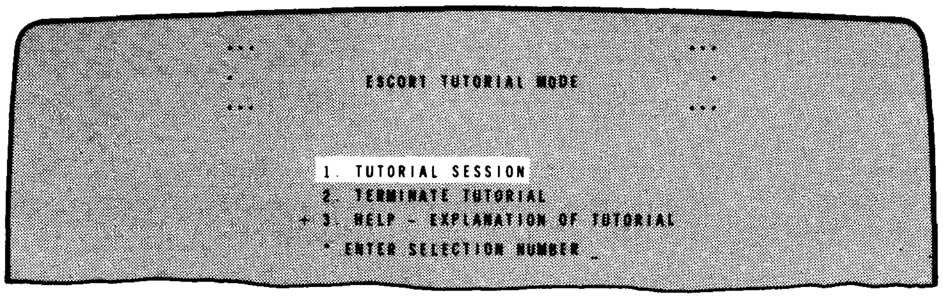
Figure 1-2 shows what happens when you choose tutorial mode.

CHOOSE TUTORIAL MODE



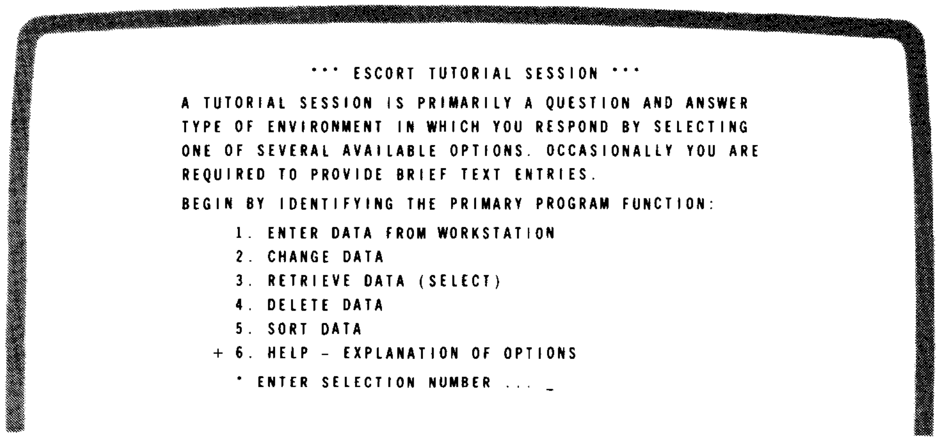
```
WELCOME TO AN ESCORT TO COMPUTERS
PLEASE SELECT YOUR ENTRY POINT:  _
1.  THE PROGRAM MODE
2.  THE TUTORIAL MODE
3.  HELP: A BRIEF DESCRIPTION OF ESCORT
4.  RECOVER A SESSION FILE BY THE NAME:
-----------------------------------------
AND CONTINUE TO THE PROGRAM MODE.....
WHEN READY, PRESS TRANSMIT TO CONTINUE
```

PICK TUTORIAL



```
***                                         ***
*           ESCORT TUTORIAL MODE              *
***                                         ***

              1. TUTORIAL SESSION
              2. TERMINATE TUTORIAL
            + 3. HELP - EXPLANATION OF TUTORIAL
              * ENTER SELECTION NUMBER _
```

PICK FROM ASSORTED TUTORIAL CHOICES



```
              *** ESCORT TUTORIAL SESSION ***
A TUTORIAL SESSION IS PRIMARILY A QUESTION AND ANSWER
TYPE OF ENVIRONMENT IN WHICH YOU RESPOND BY SELECTING
ONE OF SEVERAL AVAILABLE OPTIONS. OCCASIONALLY YOU ARE
REQUIRED TO PROVIDE BRIEF TEXT ENTRIES.
BEGIN BY IDENTIFYING THE PRIMARY PROGRAM FUNCTION:
      1. ENTER DATA FROM WORKSTATION
      2. CHANGE DATA
      3. RETRIEVE DATA (SELECT)
      4. DELETE DATA
      5. SORT DATA
    + 6. HELP - EXPLANATION OF OPTIONS
      * ENTER SELECTION NUMBER ... _
```

Figure 1-2. Program Development in Tutorial Mode

Obviously, the process is much simpler.

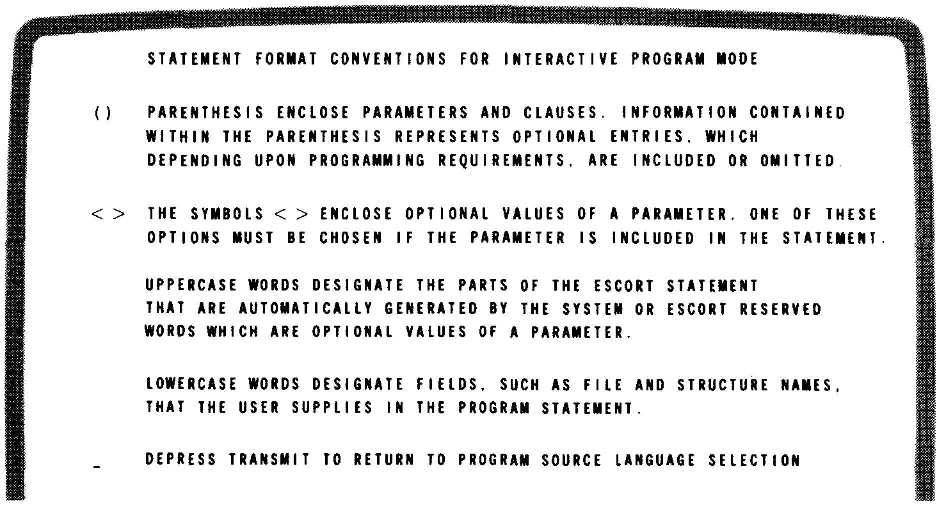
Result of choosing tutorial mode

You get the first screen of tutorial mode. From there, you can proceed with a session and build a program, terminate tutorial, or ask for help, which provides an explanation of tutorial mode.

HELP

In both Figures 1-1 and 1-2, you probably noticed a HELP option on some of the screens. This is a feature built into ESCORT to help you understand the function of the screen from which you've requested help. Here is a sample program mode HELP screen:

Sample HELP screen



```
          STATEMENT FORMAT CONVENTIONS FOR INTERACTIVE PROGRAM MODE

( )       PARENTHESIS ENCLOSE PARAMETERS AND CLAUSES. INFORMATION CONTAINED
          WITHIN THE PARENTHESIS REPRESENTS OPTIONAL ENTRIES, WHICH
          DEPENDING UPON PROGRAMMING REQUIREMENTS, ARE INCLUDED OR OMITTED.

< >       THE SYMBOLS < > ENCLOSE OPTIONAL VALUES OF A PARAMETER. ONE OF THESE
          OPTIONS MUST BE CHOSEN IF THE PARAMETER IS INCLUDED IN THE STATEMENT.

          UPPERCASE WORDS DESIGNATE THE PARTS OF THE ESCORT STATEMENT
          THAT ARE AUTOMATICALLY GENERATED BY THE SYSTEM OR ESCORT RESERVED
          WORDS WHICH ARE OPTIONAL VALUES OF A PARAMETER.

          LOWERCASE WORDS DESIGNATE FIELDS, SUCH AS FILE AND STRUCTURE NAMES,
          THAT THE USER SUPPLIES IN THE PROGRAM STATEMENT.

_         DEPRESS TRANSMIT TO RETURN TO PROGRAM SOURCE LANGUAGE SELECTION
```

1.5. STATEMENT FORMAT CONVENTIONS FOR SCREENS

The conventions for program statement formats on all ESCORT screens are:

Conventions used in presenting screens

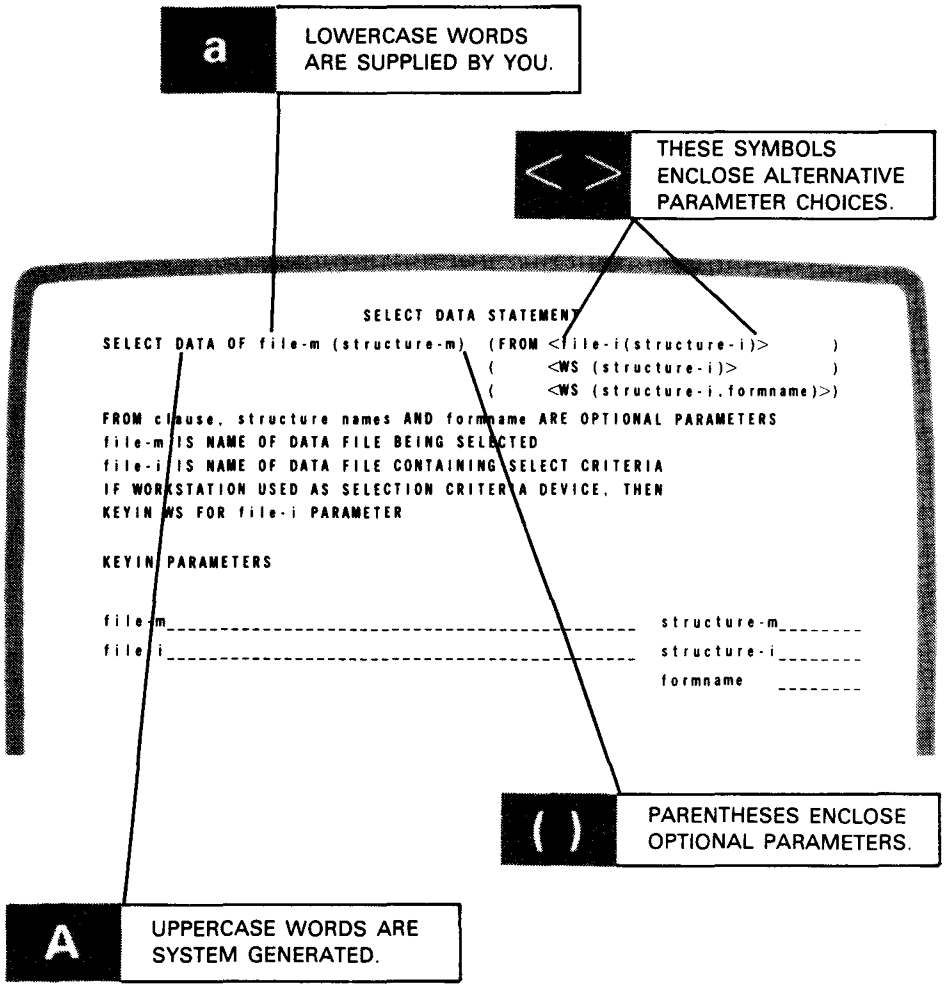
STATEMENT FORMAT CONVENTIONS FOR SCREENS	
A	Words in capital letters (uppercase) are generated for you by the system or are alternate parameter choices that are ESCORT reserved words.
a	Words with lowercase letters and embedded hyphens designate names (such as file and structure names) that you supply.
( )	Optional parameters and clauses are enclosed by parentheses.
< >	Alternative parameter choices are enclosed by the symbols < >. When you specify the parameter, you must pick one of the choices.

NOTE:

These conventions apply only to ESCORT screens. Conventions used to present statement formats in this manual are shown in 7.2.

Look at the following screen to see how each of these conventions is used:

Example of screen convention usage


a
LOWERCASE WORDS ARE SUPPLIED BY YOU.
< >
THESE SYMBOLS ENCLOSE ALTERNATIVE PARAMETER CHOICES.
( )
PARENTHESES ENCLOSE OPTIONAL PARAMETERS.
A
UPPERCASE WORDS ARE SYSTEM GENERATED.

```
                         SELECT DATA STATEMENT
SELECT DATA OF file-m (structure-m)  (FROM <file-i(structure-i)>       )
                                     (      <WS (structure-i)>         )
                                     (      <WS (structure-i.formname)>)

FROM clause, structure names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING SELECTED
file-i IS NAME OF DATA FILE CONTAINING SELECT CRITERIA
IF WORKSTATION USED AS SELECTION CRITERIA DEVICE, THEN
KEYIN WS FOR file-i PARAMETER

KEYIN PARAMETERS

file-m_______________________________________  structure-m________
file-i_______________________________________  structure-i________
                                               formname    ________
```

1.6. WHERE TO GO FROM HERE

Now that you've read Section 1 and understand what ESCORT is and how it works, you're ready to move on. This section and Section 2 both try to give you enough information so you can start up a session.

The rest of this manual is grouped according to what you might need to know at a given time.

Section 3 deals with tutorial mode, its operation, and how your program is built and executed. It includes program examples for many tutorial functions.

Section 4 deals with both the screen menu and direct entry methods of program mode. It includes program examples for many screen menu options.

Sections 5 through 9 describe, in more depth, the ESCORT concepts mentioned in the previous sections.

Inexperienced user

After reading this section and Section 2, if you're inexperienced with computers, begin with Section 3. Then, when you feel comfortable with tutorial mode, move into Section 4 (program mode).

Experienced user

If you're used to programming, you may want to skip Section 3 and move right into learning program mode.

2. Logging on the System and Calling ESCORT

2.1. PREREQUISITES

Purpose of section

This section tells you how to log on (gain entry into) your system and call ESCORT.

Data access method used by ESCORT

It's important to note that ESCORT works only under consolidated data management.

System 80 user

Series 90 user

If you're using System 80, you automatically have consolidated data management; however, for Series 90 systems, there are two types of data management possible. The type of data management was set at system generation by your system administrator. For more information on data management and system generation, see the consolidated data management concepts and facilities manual and the system installation manual. For more information on ESCORT system generation and hardware requirements, see Appendix D.

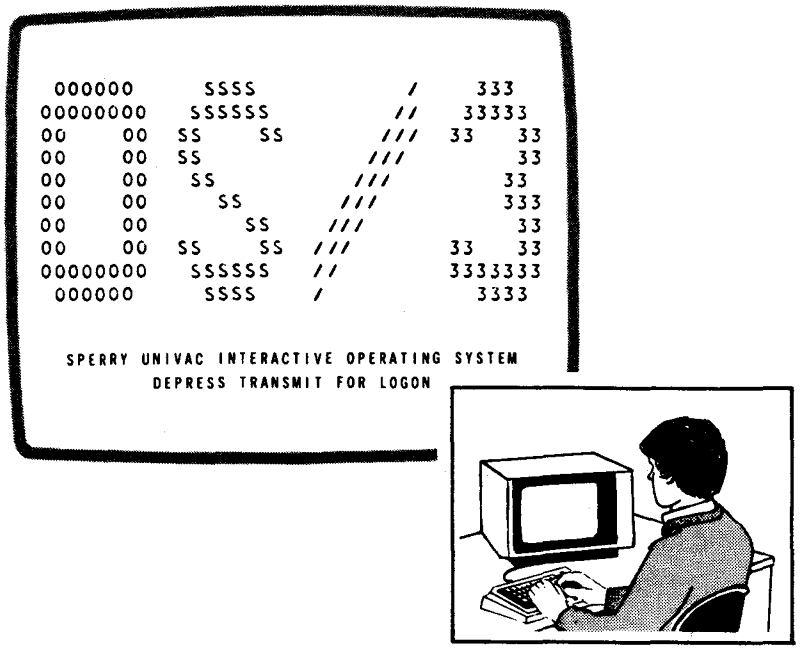
System initialization

If you haven't initialized your system, do so now. For information on this procedure, see the operations handbook.

2.2. LOGGING ON THE SYSTEM

Workstation display, idle system

For our purposes, we'll assume your system is active and your workstations are sitting idle and displaying this screen, which tells you to log on:



```
 000000     SSSS            /    333
00000000  SSSSSS           //   33333
00    00  SS     SS       ///  33    33
00    00  SS             ///          33
00    00   SS           ///          33
00    00     SS        ///           333
00    00       SS     ///             33
00    00  SS     SS  ///       33    33
00000000  SSSSSS     //        3333333
 000000     SSSS     /           3333

   SPERRY UNIVAC INTERACTIVE OPERATING SYSTEM
          DEPRESS TRANSMIT FOR LOGON
```

When to log on

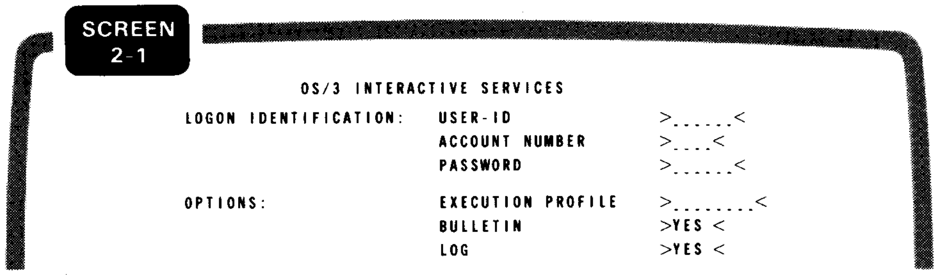
You must log on every time you use the workstation.

Purpose

Logging on tells the system you're a legitimate user, and begins an accounting of the time you use. It also provides a bulletin containing current information about system operation. For more detail on the LOGON command, see the interactive services commands and facilities user guide/programmer reference.

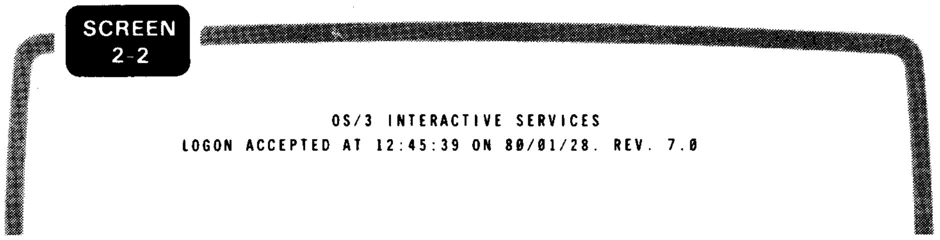
How to log on

When you press the **XMIT** key to log on, screen 2-1 appears:


SCREEN 2-1

```
                    OS/3 INTERACTIVE SERVICES
LOGON IDENTIFICATION:   USER-ID              >......<
                        ACCOUNT NUMBER       >....<
                        PASSWORD             >......<

OPTIONS:                EXECUTION PROFILE    >........<
                        BULLETIN             >YES <
                        LOG                  >YES <
```

Fill in the blanks with the required data and press the **XMIT** key. The system returns a message accepting your logon (screen 2-2):


SCREEN 2-2

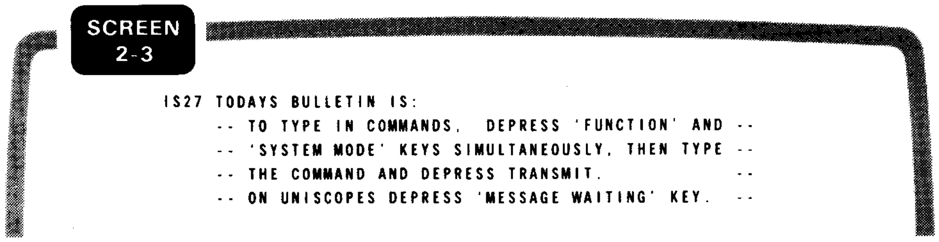
```
                OS/3 INTERACTIVE SERVICES
LOGON ACCEPTED AT 12:45:39 ON 80/01/28. REV. 7.0
```

System bulletin contents

The system then displays the system bulletin, which gives information about your system and the procedure for keying in a command.

Default system bulletin

The default system bulletin looks like this (screen 2-3):


SCREEN 2-3

```
IS27 TODAYS BULLETIN IS:
     -- TO TYPE IN COMMANDS, DEPRESS 'FUNCTION' AND --
     -- 'SYSTEM MODE' KEYS SIMULTANEOUSLY, THEN TYPE --
     -- THE COMMAND AND DEPRESS TRANSMIT.            --
     -- ON UNISCOPES DEPRESS 'MESSAGE WAITING' KEY.  --
```

2.3. HOW TO CALL ESCORT

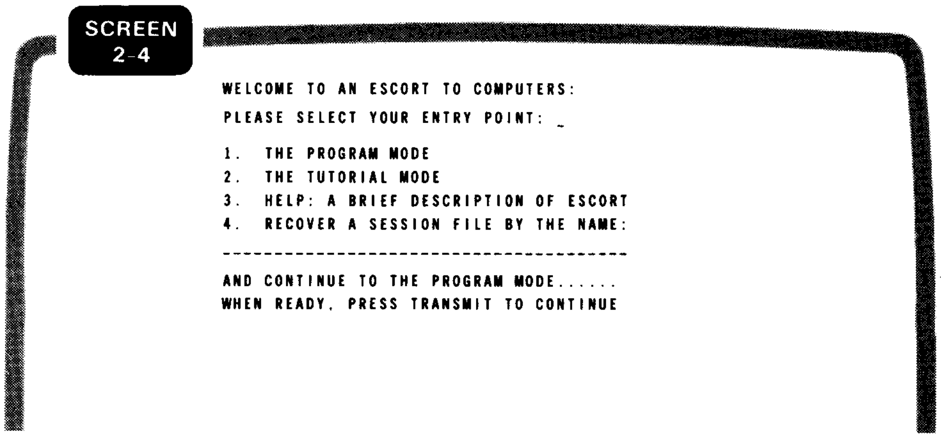
Calling ESCORT

TO CALL ESCORT

1 Press and hold down the **FUNCTION** key.	4 Key in the word ESCORT.
2 Press the **SYS MODE** key.	5 Press the **XMIT** key.
3 Release these keys.	

Starting a session

The next screen that appears (screen 2-4) is the first screen of ESCORT and looks like this:


SCREEN 2-4

```
WELCOME TO AN ESCORT TO COMPUTERS:
PLEASE SELECT YOUR ENTRY POINT: _

1.  THE PROGRAM MODE
2.  THE TUTORIAL MODE
3.  HELP: A BRIEF DESCRIPTION OF ESCORT
4.  RECOVER A SESSION FILE BY THE NAME:
--------------------------------------
AND CONTINUE TO THE PROGRAM MODE......
WHEN READY, PRESS TRANSMIT TO CONTINUE
```

Menu choices

Now you enter an ESCORT operating mode by making a menu choice. When you choose:

- 1

  Entering program mode

  You enter program mode at the command selection menu (see Figure 1-1). Here you choose the command type for the operation (program, structure, or job) you want to perform. You'll learn more about program mode in Section 4.

- 2

  Entering tutorial mode

  You enter tutorial mode. This mode guides you step by step through simple data operations and is strictly a learning tool. You'll learn all about tutorial mode in Section 3.

- 3

  HELP

  A screen that explains the other three choices appears.

- 4

  Session file recovery

  You have an opportunity to start where you left off at the end of a previous ESCORT session if you saved your session file when you terminated that session. A session file contains everything you enter or change during a session. Key in the name of the session file you want to recover.

  Password processing

  If the file you name is password protected, the following message appears at the bottom of the screen:

```
ESC117 RECOVERY/SAVE REQUIRES PASSWORD
       ENTER READ OR WRITE PASSWORD______
```

  Enter the read password for the file.

  Terminating an ESCORT session and saving the session file are described in 5.39.

Now that you've learned how to call ESCORT and initiate a session, choose number 2 on screen 2-4 to enter tutorial mode. Of course, if you're moving right into program mode, pick choice 1 and proceed to Section 4 of this manual.

PART
2

CONCEPTS OF TUTORIAL MODE

3. Interactive Tutorial Mode

3.1. WHY USE TUTORIAL MODE?

Purpose of section

This section takes you through the major tutorial mode functions and shows how they work to build a program.

Purpose of tutorial mode

Intended user

Tutorial mode is a question-and-answer method that lets you perform simple data operations while learning ESCORT programming. It is limited in scope, so use it only while you're learning ESCORT or are unfamiliar with computer programming.

In keeping with its question-and-answer format, tutorial mode consists of a series of screens that ask you:

Purpose of screens

1. To choose functions by selecting a number from a menu

2. To key in small pieces of text, such as field names, file names, or data

Effect of selection

Your responses to these screen displays determine how your ESCORT program is set up.

Mistakeproof

You can't make a mistake in statement syntax since ESCORT formats the program statements according to your menu choices.

HELP option

To help you further, most screen menus include an option called HELP.

Explanation of menu options

When you choose HELP, you receive one or more screens of information about the options presented on that tutorial menu. After viewing the contents of the HELP screens, press the **XMIT** key to return to the menu where you requested help.

Other tutorial mode topics

Resuming processing

In addition, you may select the HELP option at any point in tutorial mode by pressing the **FUNCTION** and **F2** keys. This displays a menu listing all tutorial mode topics that have HELP screens. Make a selection from the menu and view the selected HELP screens. Then, the menu reappears and you can request additional HELP screens. To resume tutorial processing at the point you left off, press the **FUNCTION** and **F2** keys.

3.2. THINGS YOU NEED TO DO BEFORE ENTERING TUTORIAL MODE

In tutorial mode, there are two steps you must take before you can create a data file. This is important because you must perform step 1 before entering ESCORT. For our purposes, we'll assume your system is active and you've logged on. You must:

Step 1: Make Sure the Files You Need Are Cataloged

Handling uncataloged files

Files are cataloged using OS/3 job control statements. In most cases, the data files you use are already allocated and cataloged. However, if you're in an ESCORT session and try to create a file that isn't cataloged, you'll get an error message. To catalog a file, press the **FUNCTION** and **SYS MODE** keys. This puts you into system mode and you can run the job control stream to catalog the file. To get back to your ESCORT session, press the **FUNCTION** and **WS MODE** keys. For more information on cataloging files, see 6.3.

Step 2: Enter ESCORT

Entering ESCORT

You enter ESCORT by keying in the word ESCORT and pressing the **XMIT** key. Then the introductory ESCORT screen appears, and you choose the mode you want. If you need a review of this process, return to 2.3.

3.3. ENTERING TUTORIAL MODE

How to enter tutorial mode

To enter tutorial mode, pick choice 2, The TUTORIAL mode, on the introductory ESCORT menu by keying in a 2 and pressing the **XMIT** key.

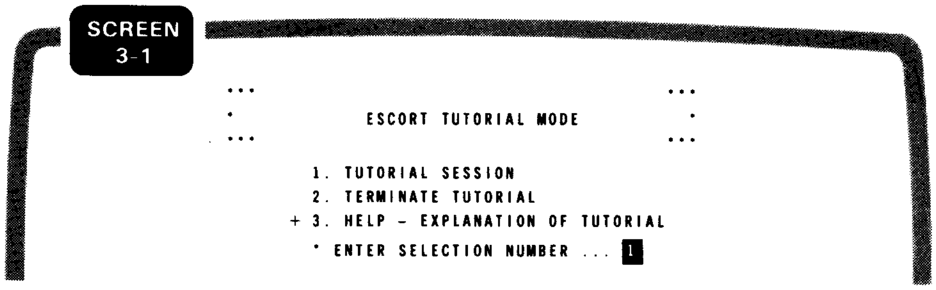
NOTE:

Standard operating procedure

Unless told otherwise, press the XMIT key after every entry to send your responses to the system.

The system then displays the first menu of tutorial mode (screen 3-1):

Tutorial mode: initial menu

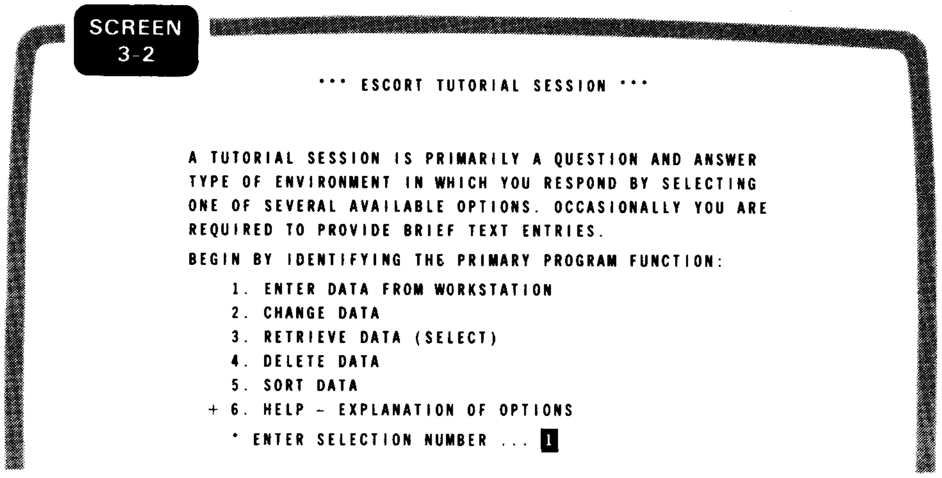


```
...                                        ...
 .          ESCORT TUTORIAL MODE            .
...                                        ...

            1. TUTORIAL SESSION
            2. TERMINATE TUTORIAL
          + 3. HELP - EXPLANATION OF TUTORIAL
          * ENTER SELECTION NUMBER ... 1
```

The plus sign (+) on tutorial mode menus indicates the default selection. You need not key in a selection number to get the default selection; just press the **XMIT** key.

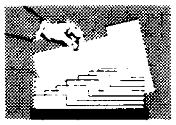
Because you want to start a session, pick 1, TUTORIAL SESSION. This displays screen 3-2, which asks what function you want to perform:

Function selection



```
            *** ESCORT TUTORIAL SESSION ***


A TUTORIAL SESSION IS PRIMARILY A QUESTION AND ANSWER
TYPE OF ENVIRONMENT IN WHICH YOU RESPOND BY SELECTING
ONE OF SEVERAL AVAILABLE OPTIONS. OCCASIONALLY YOU ARE
REQUIRED TO PROVIDE BRIEF TEXT ENTRIES.
BEGIN BY IDENTIFYING THE PRIMARY PROGRAM FUNCTION:
    1. ENTER DATA FROM WORKSTATION
    2. CHANGE DATA
    3. RETRIEVE DATA (SELECT)
    4. DELETE DATA
    5. SORT DATA
  + 6. HELP - EXPLANATION OF OPTIONS
    * ENTER SELECTION NUMBER ... 1
```

The following subsections describe how these choices work.



3.4. ENTERING DATA

Purpose

Since this is your first session, the first thing you'll want to do is enter data, so pick selection 1, ENTER DATA FROM WORKSTATION, on screen 3-2. The next screen that appears (screen 3-3) is the first screen of the data entry function, which gives you two options for entering data:

Options

1. Create a new data file

2. Extend an existing data file

Option selected

Because this is your first session, you'll want to create a data file and insert data into it. Once your file is created, you can go back at another time and insert more data by using the extending option.

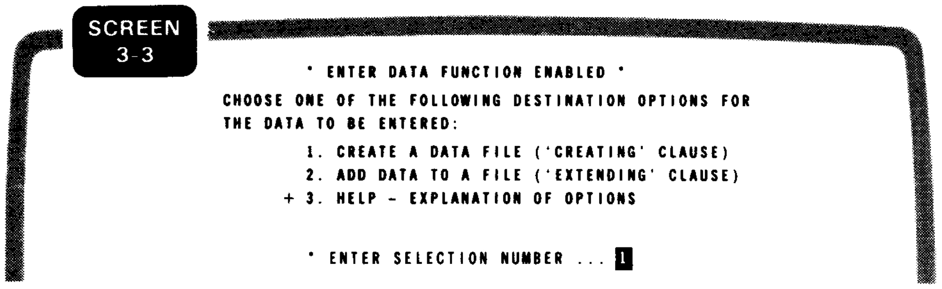
3.5. CREATING A DATA FILE

Prerequisite

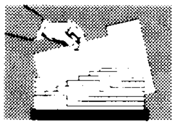
As you recall from 3.2, there is one step you must complete before you can create a data file: you must catalog your file.

Selection

Assuming you have completed this step, key in choice number 1, CREATE A DATA FILE (screen 3-3):

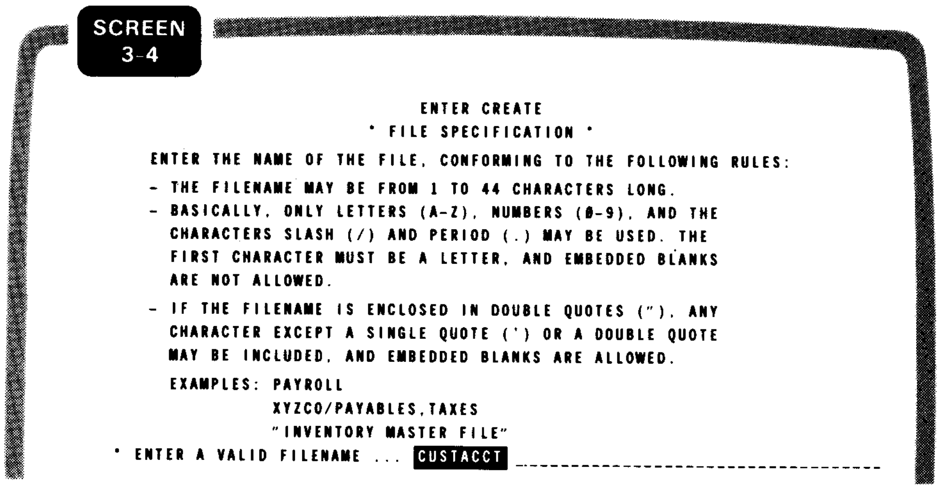


```
            * ENTER DATA FUNCTION ENABLED *
CHOOSE ONE OF THE FOLLOWING DESTINATION OPTIONS FOR
THE DATA TO BE ENTERED:
            1. CREATE A DATA FILE ('CREATING' CLAUSE)
            2. ADD DATA TO A FILE ('EXTENDING' CLAUSE)
          + 3. HELP - EXPLANATION OF OPTIONS

            * ENTER SELECTION NUMBER ... 1
```



The next screen (screen 3-4) asks you to name your file:

Specifying file name



```
                    ENTER CREATE
                 * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
 - THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
 - BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
   CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
   FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
   ARE NOT ALLOWED.
 - IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
   CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
   MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
   EXAMPLES: PAYROLL
             XYZCO/PAYABLES.TAXES
             "INVENTORY MASTER FILE"
* ENTER A VALID FILENAME ... CUSTACCT ________________________________
```

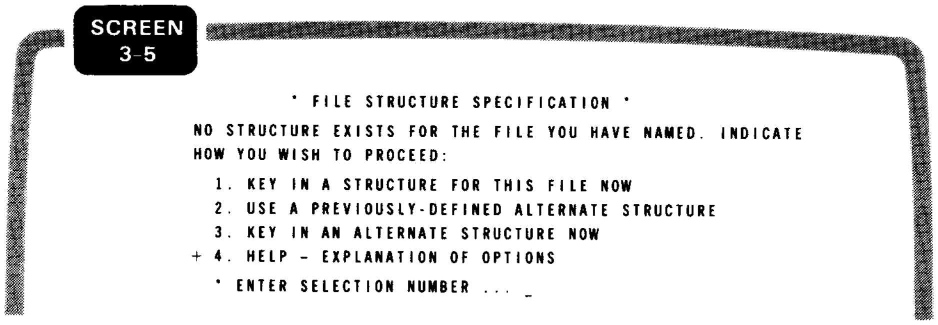
File is named CUSTACCT

You name your file by following the rules listed on the screen (we named our file CUSTACCT).

ESCORT search

At this point, ESCORT searches the session file and the library file for a structure with the same name as the first eight characters of your file name. Because we have not yet created a structure, ESCORT doesn't find one, and screen 3-5 appears:

Result of search



```
           * FILE STRUCTURE SPECIFICATION *
NO STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
HOW YOU WISH TO PROCEED:
  1. KEY IN A STRUCTURE FOR THIS FILE NOW
  2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
  3. KEY IN AN ALTERNATE STRUCTURE NOW
+ 4. HELP - EXPLANATION OF OPTIONS
  * ENTER SELECTION NUMBER ... _
```

Using alternate structures

If you choose number 2, another screen appears, asking the name of the structure you want to use. When you key in an acceptable name, you can continue with your program.

Using a new structure

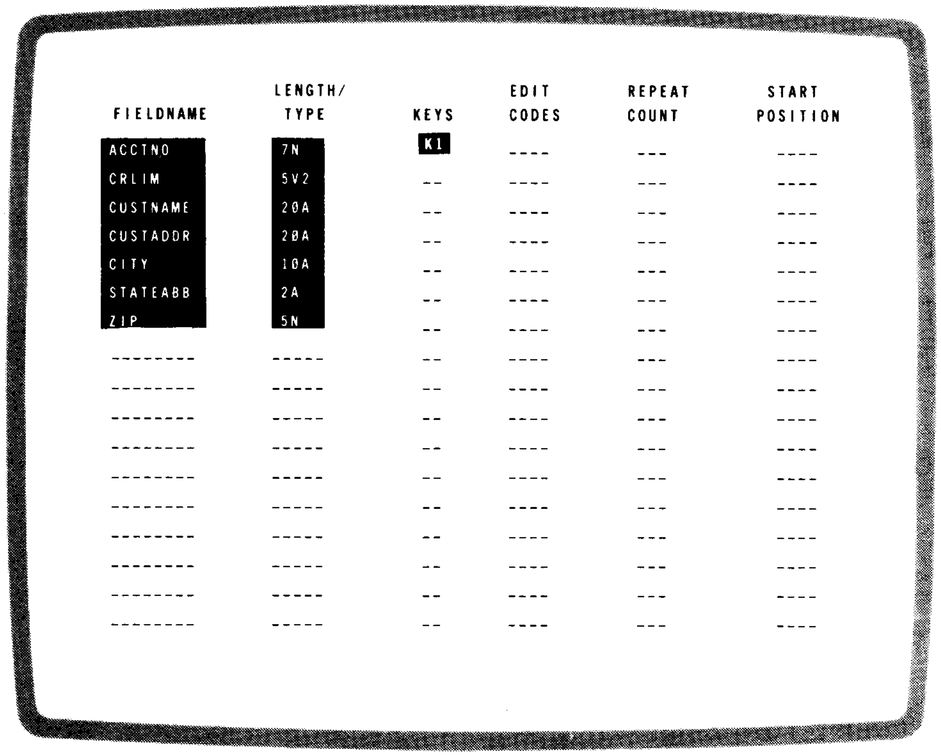
If you choose number 1 or number 3, a screen appears allowing you to enter a structure. It's the same screen you use to enter a structure in program mode. When you complete your structure entry, you can continue with your program.

ENTERING DATA

Structure definition

Because we haven't created a structure, choose number 1 on screen 3-5 and key in the structure (which ESCORT calls CUSTACCT, the same as your file name) on the following screen:

STRUCTURE OF CUSTACCT

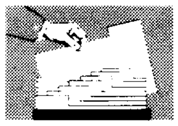


```
                  LENGTH/              EDIT     REPEAT    START
FIELDNAME         TYPE       KEYS      CODES    COUNT     POSITION
ACCTNO            7N         K1        ----     ---       ----
CRLIM             5V2        --        ----     ---       ----
CUSTNAME          20A        --        ----     ---       ----
CUSTADDR          20A        --        ----     ---       ----
CITY              10A        --        ----     ---       ----
STATEABB          2A         --        ----     ---       ----
ZIP               5N         --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
--------          -----      --        ----     ---       ----
```

Corresponding structure entries

FIELD NAMES	EXPLANATION OF FIELD NAMES
ACCTNO	*Account number*
CRLIM	*Credit limit*
CUSTNAME	*Customer name*
CUSTADDR	*Customer address*
CITY	*City*
STATEABB	*State abbreviation*
ZIP	*Zip code*

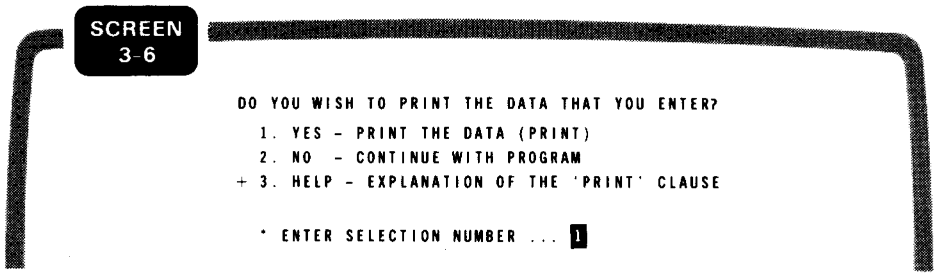
The N, V, and A designations in the structure tell ESCORT what type of data will be contained in that field. For an explanation of these and other data type designations, see 6.6.



The next screen (screen 3-6) asks if you want the data you enter to be printed.

Printing data

SCREEN 3-6



```
DO YOU WISH TO PRINT THE DATA THAT YOU ENTER?
  1. YES - PRINT THE DATA (PRINT)
  2. NO  - CONTINUE WITH PROGRAM
+ 3. HELP - EXPLANATION OF THE 'PRINT' CLAUSE

  * ENTER SELECTION NUMBER ... 1
```

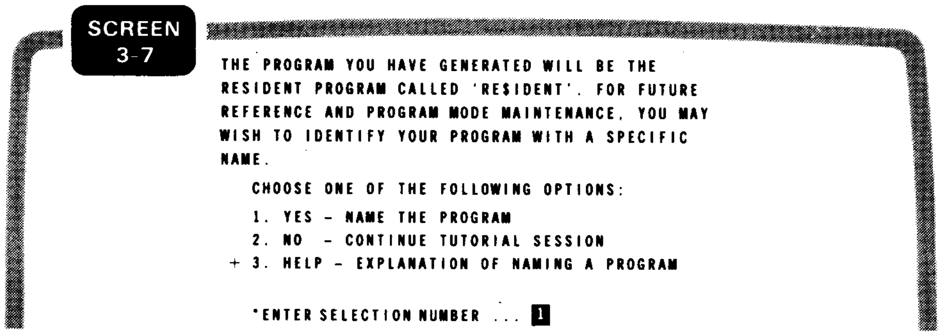
PRINT selected

This is your option, but for our purposes, we'll choose to print our data by selecting 1, YES - PRINT THE DATA (PRINT).

This option prints every record you enter on the printer. This is helpful to get a report of what you entered so you can check for accuracy or just keep track of what's been entered and what hasn't.

If you want to name your program, designate this on screen 3-7.

SCREEN 3-7

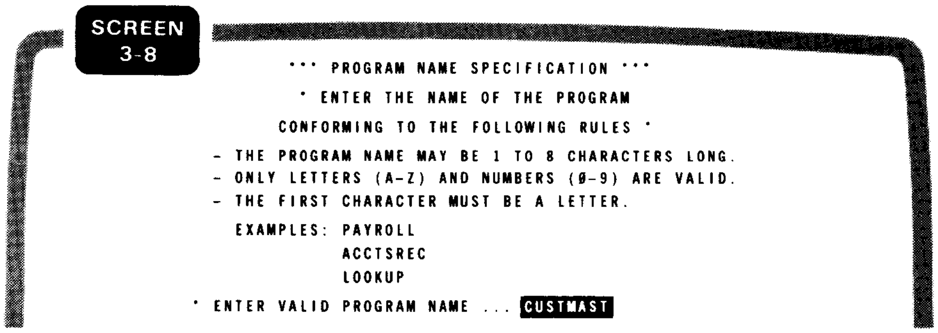


```
THE PROGRAM YOU HAVE GENERATED WILL BE THE
RESIDENT PROGRAM CALLED 'RESIDENT'. FOR FUTURE
REFERENCE AND PROGRAM MODE MAINTENANCE, YOU MAY
WISH TO IDENTIFY YOUR PROGRAM WITH A SPECIFIC
NAME.
   CHOOSE ONE OF THE FOLLOWING OPTIONS:
   1. YES - NAME THE PROGRAM
   2. NO  - CONTINUE TUTORIAL SESSION
 + 3. HELP - EXPLANATION OF NAMING A PROGRAM

   *ENTER SELECTION NUMBER ... 1
```

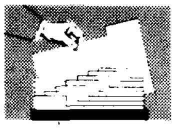
Assigning program name

The next screen (screen 3-8) asks you to name your program:

SCREEN 3-8



```
            *** PROGRAM NAME SPECIFICATION ***
             * ENTER THE NAME OF THE PROGRAM
           CONFORMING TO THE FOLLOWING RULES *
- THE PROGRAM NAME MAY BE 1 TO 8 CHARACTERS LONG.
- ONLY LETTERS (A-Z) AND NUMBERS (0-9) ARE VALID.
- THE FIRST CHARACTER MUST BE A LETTER.
  EXAMPLES: PAYROLL
            ACCTSREC
            LOOKUP
* ENTER VALID PROGRAM NAME ... CUSTMAST
```

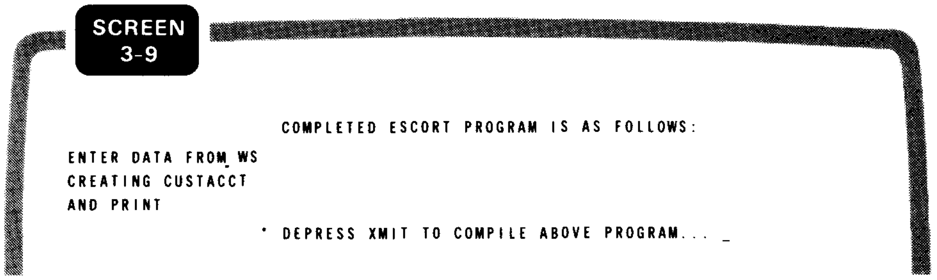

ENTERING DATA

Enter program name

Key in your program name (we named it CUSTMAST), conforming to the rules on the screen.

ESCORT program listing

Then screen 3-9 appears, listing your ESCORT program:


SCREEN 3-9

```
                COMPLETED ESCORT PROGRAM IS AS FOLLOWS:
ENTER DATA FROM WS
CREATING CUSTACCT
AND PRINT
              * DEPRESS XMIT TO COMPILE ABOVE PROGRAM... _
```

Compilation process

Press the **XMIT** key to compile your program. Compile means to check the syntax and statement order so the program can execute.

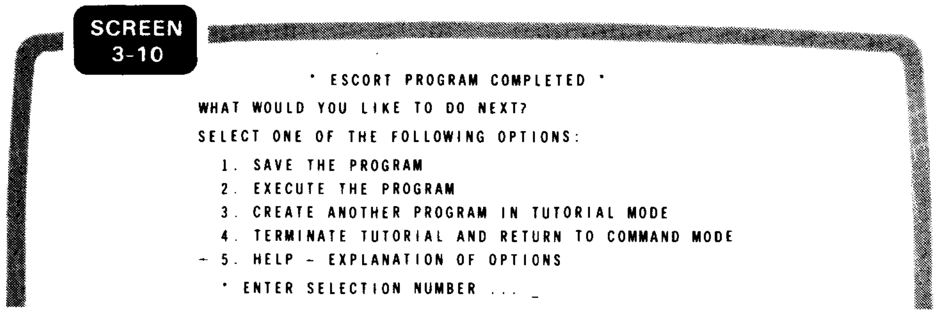
At this point, let's review what you've done. From the point where you entered the tutorial session, you:

Summary of functions completed

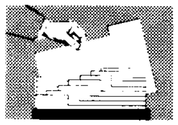
1. Chose to enter your data from the workstation
2. Chose to create a new data file
3. Named your file (CUSTACCT)
4. Chose to print the data you enter
5. Named your program (CUSTMAST)

Program disposition

The next step is to decide what you want to do with this program. Do you want to save it? Do you want to execute it? You decide this on the following "decision" screen (screen 3-10):


SCREEN 3-10

```
          * ESCORT PROGRAM COMPLETED *
WHAT WOULD YOU LIKE TO DO NEXT?
SELECT ONE OF THE FOLLOWING OPTIONS:
  1. SAVE THE PROGRAM
  2. EXECUTE THE PROGRAM
  3. CREATE ANOTHER PROGRAM IN TUTORIAL MODE
  4. TERMINATE TUTORIAL AND RETURN TO COMMAND MODE
→ 5. HELP - EXPLANATION OF OPTIONS
  * ENTER SELECTION NUMBER ... _
```



Here are your options:

Saving the program

- Selection number 1, SAVE THE PROGRAM, places your program permanently on the ESCORT library file so you can execute or change it in a later session.

Executing the program

- Selection number 2, EXECUTE THE PROGRAM, immediately executes your program.

Creating another program

- Selection number 3, CREATE ANOTHER PROGRAM IN TUTORIAL MODE, allows you to start tutorial mode again and create another program. Before you make this selection, be sure you saved the program just created if you plan to reuse it in another session. If you plan to reuse it in the same session, all you have to do is name it (screens 3-7 and 3-8). If you don't save it, ESCORT overwrites the program in the session file when you create another program.

Terminating tutorial mode

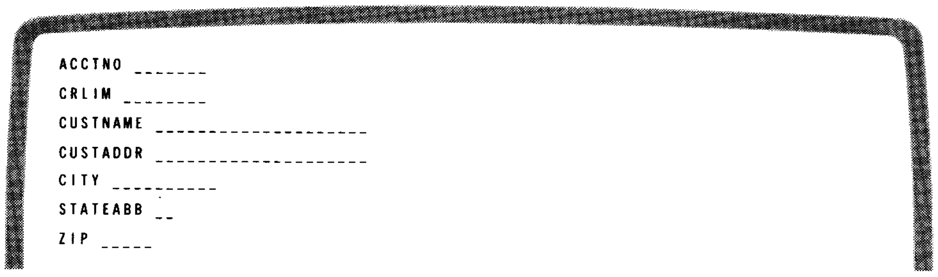
- Selection number 4, TERMINATE TUTORIAL AND RETURN TO COMMAND MODE, terminates tutorial mode and takes you to the command selection menu, which gives you three options:

Options after terminating tutorial mode

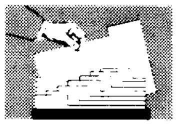
1. Enter program mode by selecting one of the command types (structure, program, or job)
2. Enter tutorial mode
3. Terminate your session

For this example, you want to execute your program, so choose number 2, EXECUTE PROGRAM, on screen 3-10. If you followed the example, the program you created (CUSTMAST) produces the following screen template (which parallels your structure) so you can enter data into your file:

Entering data



```
ACCTNO _______
CRLIM ________
CUSTNAME ____________________
CUSTADDR ____________________
CITY __________
STATEABB __
ZIP _____
```

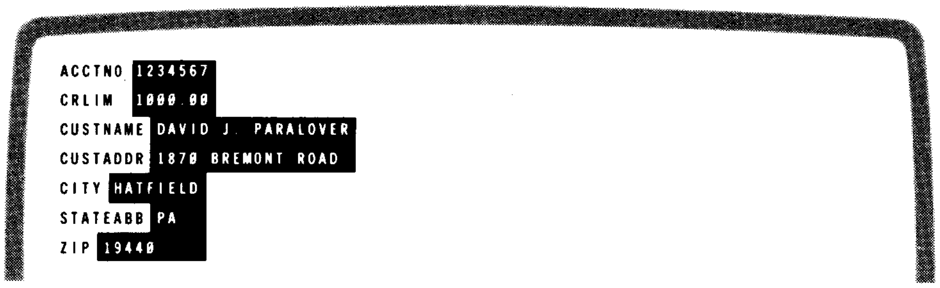


Sending data to the system

Fill in each field with data and when you complete the last field, press the **XMIT** key.

This screen shows some random data we keyed in.

Example of data entry



```
ACCTNO 1234567
CRLIM 1000.00
CUSTNAME DAVID J. PARALOVER
CUSTADDR 1870 BREMONT ROAD
CITY HATFIELD
STATEABB PA
ZIP 19440
```

Entering additional data

The data you keyed in is stored as a record, and the screen template reappears so you can key in another data record. This example uses five records. (For purposes of brevity, assume we've entered random data into our fields. If you want, you can enter your own data.)

Moving from field to field

When you key data into a field and it takes up the entire amount of space allotted for that field (e.g., a 7-digit account number into a 7N field), the cursor automatically moves to the next field. If, however, you key in data that doesn't fill the space allotted (e.g., a 15-character name in a 20A field), you must move the cursor to the next field by pressing the **TAB FORWARD** key.

Terminating data entry

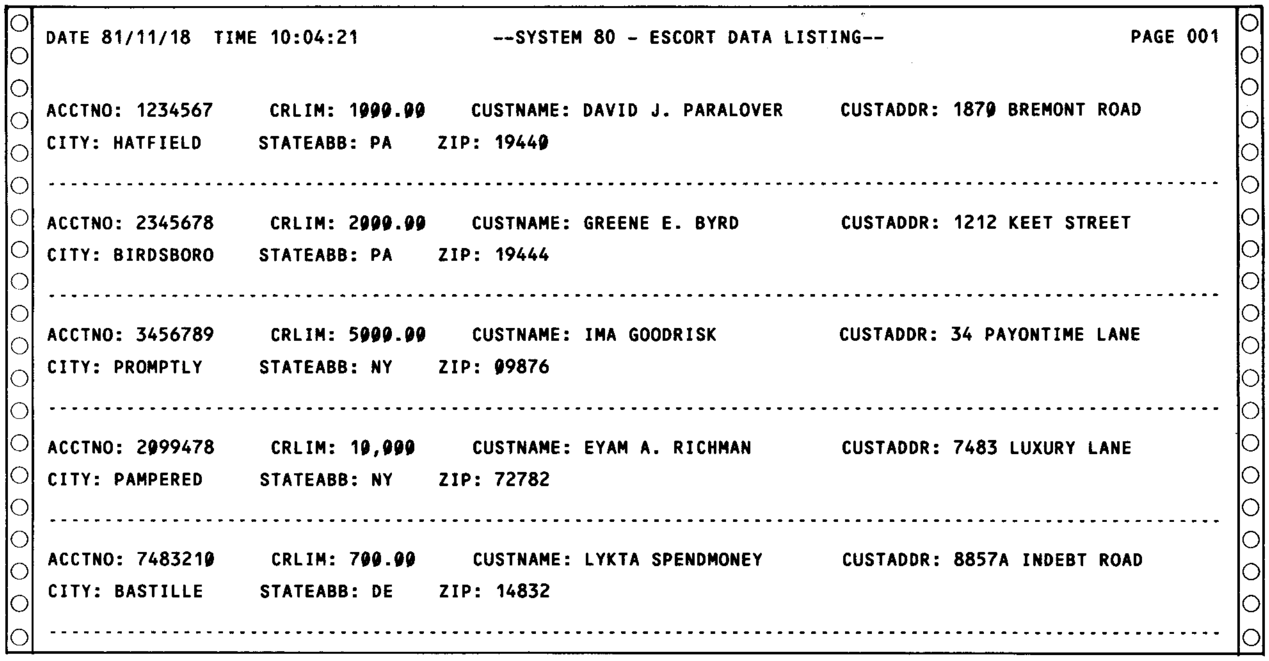
When you enter all the records you want in your file, press the **FUNCTION** and **F4** keys to signify the end of input to the system.

At this point, because you chose to print your data (screen 3-6), ESCORT prints every data record you entered (five).

ENTERING DATA

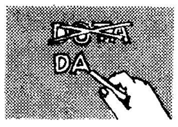
Example of data printout

Based on the data we entered, the printer output for this example looks like this:



```
DATE 81/11/18  TIME 10:04:21          --SYSTEM 80 - ESCORT DATA LISTING--                  PAGE 001

ACCTNO: 1234567     CRLIM: 1000.00    CUSTNAME: DAVID J. PARALOVER     CUSTADDR: 1870 BREMONT ROAD
CITY: HATFIELD     STATEABB: PA    ZIP: 19440
------------------------------------------------------------------------------------------------
ACCTNO: 2345678     CRLIM: 2000.00    CUSTNAME: GREENE E. BYRD         CUSTADDR: 1212 KEET STREET
CITY: BIRDSBORO    STATEABB: PA    ZIP: 19444
------------------------------------------------------------------------------------------------
ACCTNO: 3456789     CRLIM: 5000.00    CUSTNAME: IMA GOODRISK           CUSTADDR: 34 PAYONTIME LANE
CITY: PROMPTLY     STATEABB: NY    ZIP: 09876
------------------------------------------------------------------------------------------------
ACCTNO: 2099478     CRLIM: 10,000     CUSTNAME: EYAM A. RICHMAN        CUSTADDR: 7483 LUXURY LANE
CITY: PAMPERED     STATEABB: NY    ZIP: 72782
------------------------------------------------------------------------------------------------
ACCTNO: 7483210     CRLIM: 700.00     CUSTNAME: LYKTA SPENDMONEY       CUSTADDR: 8857A INDEBT ROAD
CITY: BASTILLE     STATEABB: DE    ZIP: 14832
------------------------------------------------------------------------------------------------
```

Then the decision screen (screen 3-10) reappears, and you decide where to go from here.


CHANGING DATA

3.6. CHANGING DATA

Using the data file created in 3.5, change it by using one of three methods available to you:

Methods of change

1. Selecting the records to be changed and changing them from the workstation
2. Changing records based on conditions
3. Changing all records

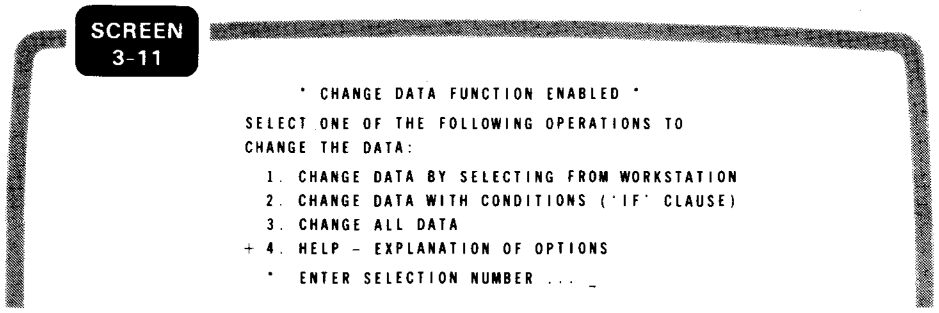
Entering tutorial mode

You enter tutorial mode by the method outlined in 3.3. When you get to the function screen (screen 3-2), choose number 2, CHANGE DATA.

Selecting a method

The next screen that appears is the CHANGE DATA menu (screen 3-11); it offers you three options and a HELP facility, which explains these options:

SCREEN 3-11



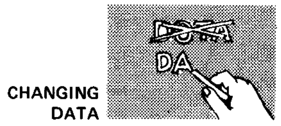
```
         * CHANGE DATA FUNCTION ENABLED *
SELECT ONE OF THE FOLLOWING OPERATIONS TO
CHANGE THE DATA:
  1. CHANGE DATA BY SELECTING FROM WORKSTATION
  2. CHANGE DATA WITH CONDITIONS ('IF' CLAUSE)
  3. CHANGE ALL DATA
+ 4. HELP - EXPLANATION OF OPTIONS
   *  ENTER SELECTION NUMBER ... _
```

3.7. CHANGING DATA FROM THE WORKSTATION

When you change data by selecting from the workstation, each record (if you have an unkeyed file) or the key fields in each record (if you have a keyed file) are displayed on the workstation. (See 6.2 and 6.7 for details on keyed and unkeyed files.) If you have an unkeyed file, you simply change the data fields you want to change by keying over the data that is displayed. If you have a keyed file, the key field prompt is displayed on the workstation, you enter the key data for the record you want to change, and that record is then displayed on the workstation and you make your changes.

Method for unkeyed files

Method for keyed files

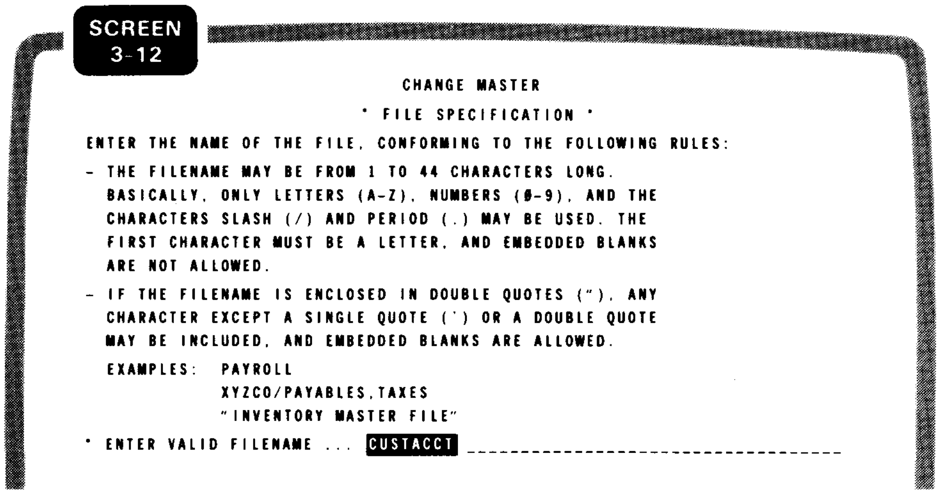


Selecting the change method

To choose CHANGE DATA BY SELECTING FROM WORKSTATION on screen 3-11, key in number 1 and press the **XMIT** key.

Name the file to be changed

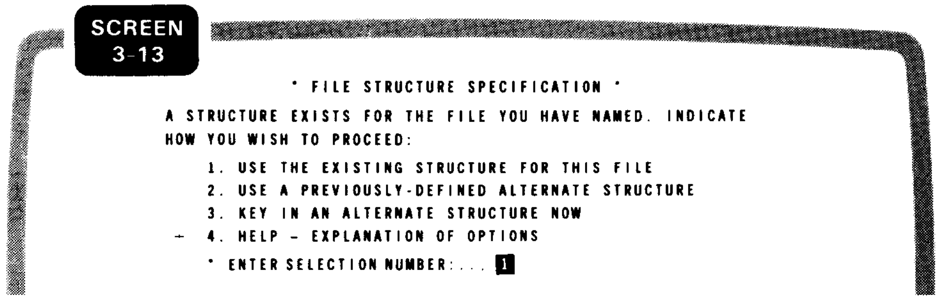
The next screen (screen 3-12) asks you to name the file you want to change. Because you're using the same keyed file you created in 3.5, key in the name CUSTACCT.



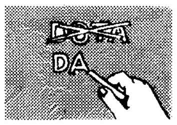
```
                              CHANGE MASTER
                          * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
  BASICALLY, ONLY LETTERS (A-Z), NUMBERS (Ø-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES:  PAYROLL
             XYZCO/PAYABLES.TAXES
             "INVENTORY MASTER FILE"
* ENTER VALID FILENAME ... CUSTACCT ________________________________
```

As previously explained (3.5), ESCORT now searches the session and library files to find the structure with the same name as the first eight characters of your file name.

Because the structure you created when you entered data is in the session file, screen 3-13 appears. Choose selection 1 to use the previously created structure.



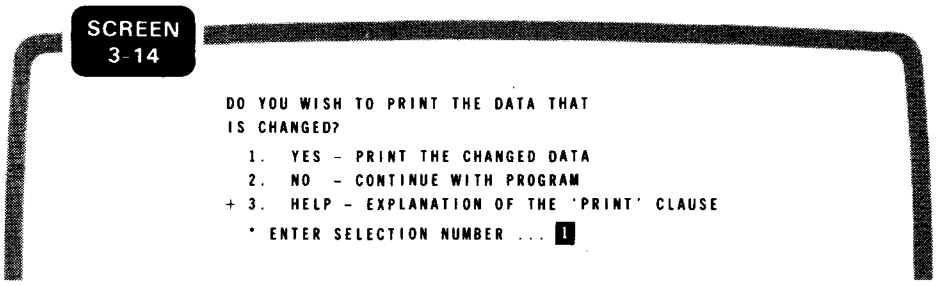
```
                    * FILE STRUCTURE SPECIFICATION *
        A STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
        HOW YOU WISH TO PROCEED:
            1. USE THE EXISTING STRUCTURE FOR THIS FILE
            2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
            3. KEY IN AN ALTERNATE STRUCTURE NOW
          - 4. HELP - EXPLANATION OF OPTIONS
            * ENTER SELECTION NUMBER:... 1
```



CHANGING DATA

Printing changed data

The next screen (screen 3-14) asks if you want the changed data printed:



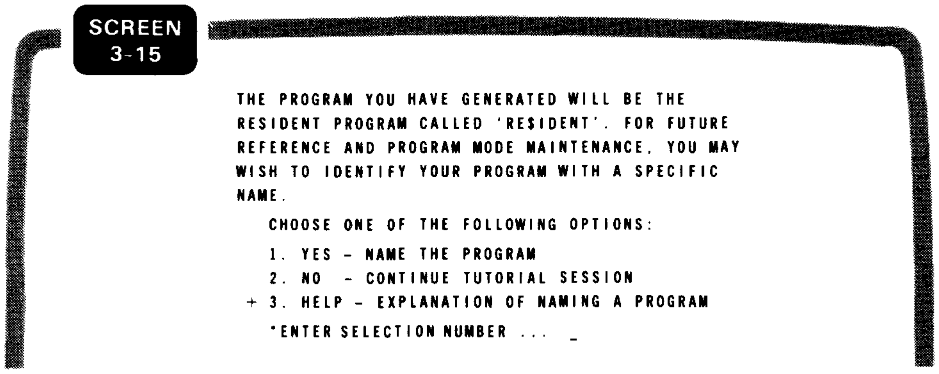
```
DO YOU WISH TO PRINT THE DATA THAT
IS CHANGED?
  1.  YES - PRINT THE CHANGED DATA
  2.  NO  - CONTINUE WITH PROGRAM
+ 3.  HELP - EXPLANATION OF THE 'PRINT' CLAUSE
  * ENTER SELECTION NUMBER ... 1
```

Printing selected

For your purposes, choose number 1, YES - PRINT THE CHANGED DATA and press the **XMIT** key.

If you wanted to name your program, you could designate this on screen 3-15:

Naming programs

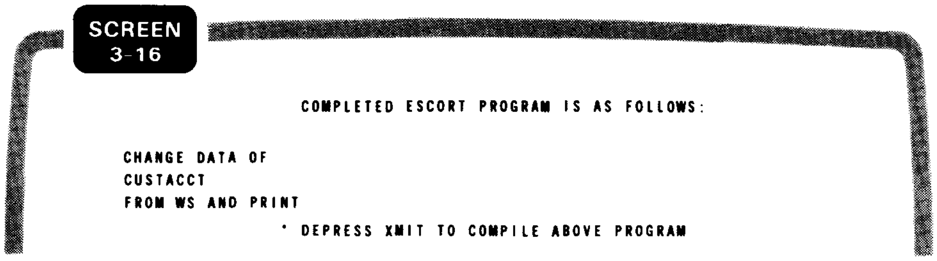


```
THE PROGRAM YOU HAVE GENERATED WILL BE THE
RESIDENT PROGRAM CALLED 'RE$IDENT'. FOR FUTURE
REFERENCE AND PROGRAM MODE MAINTENANCE, YOU MAY
WISH TO IDENTIFY YOUR PROGRAM WITH A SPECIFIC
NAME.
   CHOOSE ONE OF THE FOLLOWING OPTIONS:
   1. YES - NAME THE PROGRAM
   2. NO  - CONTINUE TUTORIAL SESSION
 + 3. HELP - EXPLANATION OF NAMING A PROGRAM
   *ENTER SELECTION NUMBER ... _
```

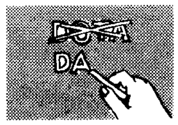
Because we don't want to name our program, ESCORT assumes we're using the RE$IDENT name.

Then screen 3-16 appears, listing your ESCORT program:

Program listing



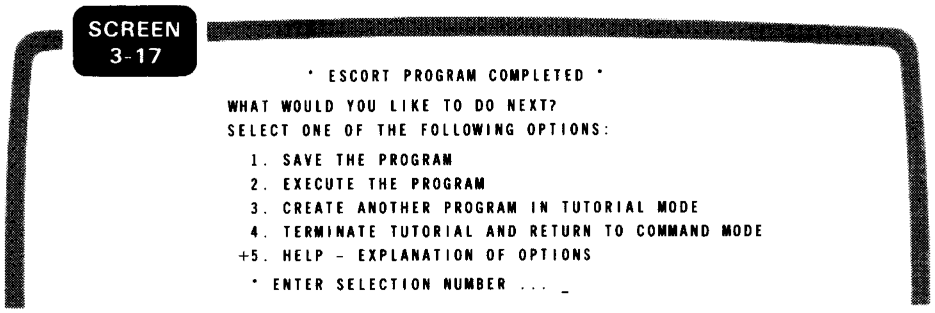
```
               COMPLETED ESCORT PROGRAM IS AS FOLLOWS:

CHANGE DATA OF
CUSTACCT
FROM WS AND PRINT
              * DEPRESS XMIT TO COMPILE ABOVE PROGRAM
```



CHANGING DATA

Compiling your program

When you press the **XMIT** key to compile your data change program, the following decision screen (3-17) appears:



SCREEN 3-17

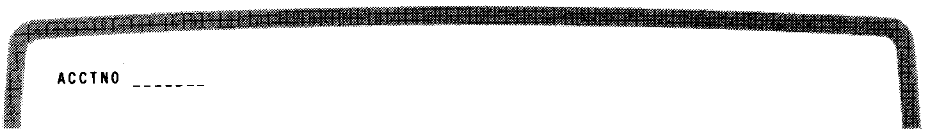
```
        * ESCORT PROGRAM COMPLETED *
WHAT WOULD YOU LIKE TO DO NEXT?
SELECT ONE OF THE FOLLOWING OPTIONS:
   1. SAVE THE PROGRAM
   2. EXECUTE THE PROGRAM
   3. CREATE ANOTHER PROGRAM IN TUTORIAL MODE
   4. TERMINATE TUTORIAL AND RETURN TO COMMAND MODE
  +5. HELP - EXPLANATION OF OPTIONS
  * ENTER SELECTION NUMBER ... _
```

Executing the program

The action initiated by each of these selections is listed under screen 3-10. For your purposes, choose number 2, EXECUTE THE PROGRAM, to execute your program.

Sample output

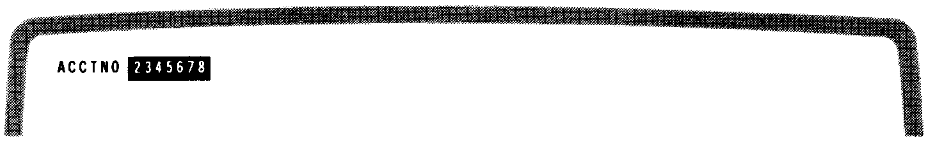
Because your file is a keyed file and its key field is ACCTNO, the following prompt is displayed on your workstation screen:



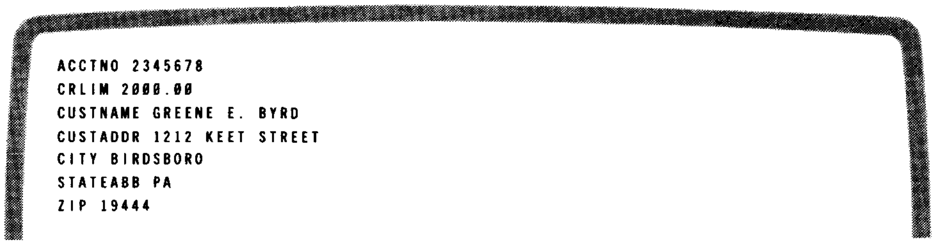
```
ACCTNO _______
```

Example

Suppose you want to change the record for customer Greene E. Byrd. The account number for that customer's record is 2345678. You key in that account number, and the rest of the record is displayed. Then you can make your changes.



```
ACCTNO 2345678
```



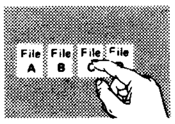
```
ACCTNO 2345678
CRLIM 2000.00
CUSTNAME GREENE E. BYRD
CUSTADDR 1212 KEET STREET
CITY BIRDSBORO
STATEABB PA
ZIP 19444
```

CHANGING DATA

After you make your changes for that record and press the **XMIT** key, the prompt reappears so you can change another record.

Terminating the change

When you finish changing your records, press the **FUNCTION** and **F4** keys to specify the end of input.



RETRIEVING DATA

3.8. RETRIEVING DATA (SELECT)

Using the same data file as in previous subsections, now select (retrieve) data, using one of three available methods:

Data selection methods

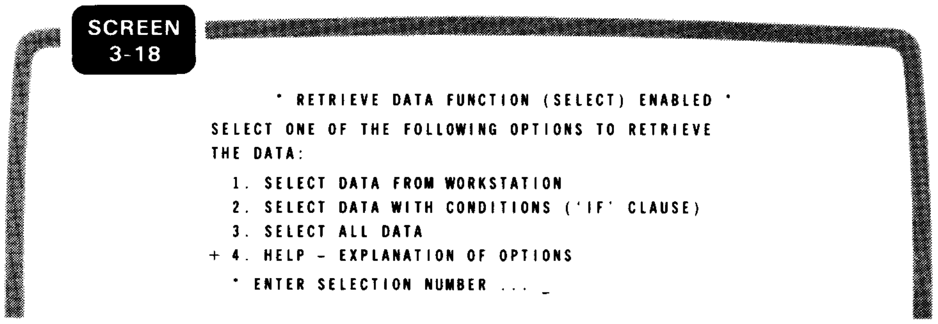
1. Select from the workstation
2. Select based on conditions
3. Select all data

Entering tutorial mode

Enter tutorial mode, using the method outlined in 3.3. When you get to the function screen (screen 3-2), choose number 3, RETRIEVE DATA (SELECT).

Data selection options

The next screen that appears is the SELECT DATA menu (screen 3-18), which offers you three options and a HELP facility explaining these options.


SCREEN 3-18

```
        * RETRIEVE DATA FUNCTION (SELECT) ENABLED *
SELECT ONE OF THE FOLLOWING OPTIONS TO RETRIEVE
THE DATA:
   1. SELECT DATA FROM WORKSTATION
   2. SELECT DATA WITH CONDITIONS ('IF' CLAUSE)
   3. SELECT ALL DATA
+  4. HELP - EXPLANATION OF OPTIONS
   * ENTER SELECTION NUMBER ... _
```

3.9. SELECTING DATA BASED ON CONDITIONS

Selecting data based on conditions means that you designate one or more conditions that must be met before the record is selected. Using the data file and structure (CUSTACCT) created in 3.5, this SELECT DATA example selects the records from CUSTACCT only if the credit limit field (CRLIM) is over 5000 dollars.

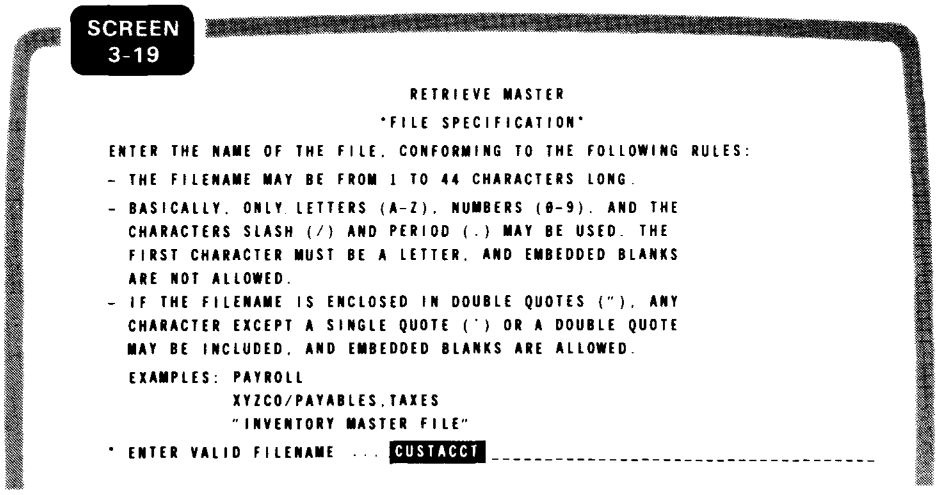
Choosing the select function

To choose SELECT DATA WITH CONDITIONS, key in number 2 on the SELECT DATA menu (screen 3-18) and press the **XMIT** key.

RETRIEVING DATA

The next screen (screen 3-19) asks you to name the file from which you want to select records. Because you're using the same file you created in 3.5, key in the name CUSTACCT.

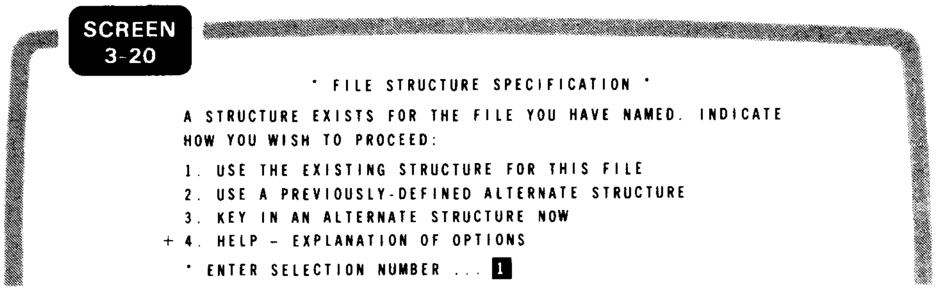
Naming the file


SCREEN 3-19

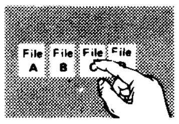
```
                         RETRIEVE MASTER
                       *FILE SPECIFICATION*
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
- BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES: PAYROLL
            XYZCO/PAYABLES,TAXES
            "INVENTORY MASTER FILE"
* ENTER VALID FILENAME ... CUSTACCT_________________________________
```

As previously explained (3.5), ESCORT now searches the session and library files to find the structure with the same name as the first eight characters of your file name.

Because a structure for your file exists, screen 3-20 appears. Again, choose selection 1 and use the previously created structure.


SCREEN 3-20

```
              * FILE STRUCTURE SPECIFICATION *
   A STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
   HOW YOU WISH TO PROCEED:
   1. USE THE EXISTING STRUCTURE FOR THIS FILE
   2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
   3. KEY IN AN ALTERNATE STRUCTURE NOW
 + 4. HELP - EXPLANATION OF OPTIONS
   * ENTER SELECTION NUMBER ... 1
```

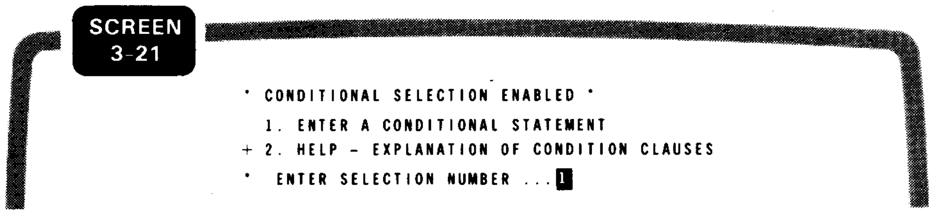


RETRIEVING DATA

At this point, ESCORT knows what file you want to access, and the next screen (screen 3-21) asks you to specify the conditions for selecting data from that file.

Condition specification

SCREEN 3-21

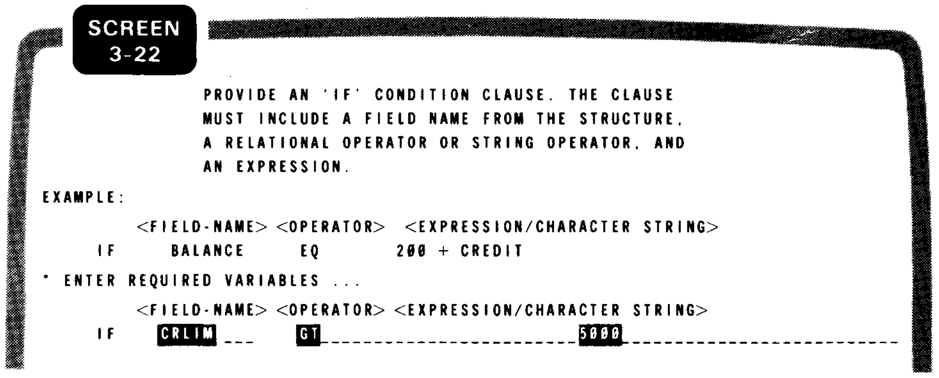


```
* CONDITIONAL SELECTION ENABLED *
  1. ENTER A CONDITIONAL STATEMENT
+ 2. HELP - EXPLANATION OF CONDITION CLAUSES
*  ENTER SELECTION NUMBER ...1
```

Example

Key in number 1. Screen 3-22 appears, and you key in your conditional statement. The record selection for this example depends on whether the value of the credit limit field (CRLIM) is more than 5000 dollars.

SCREEN 3-22

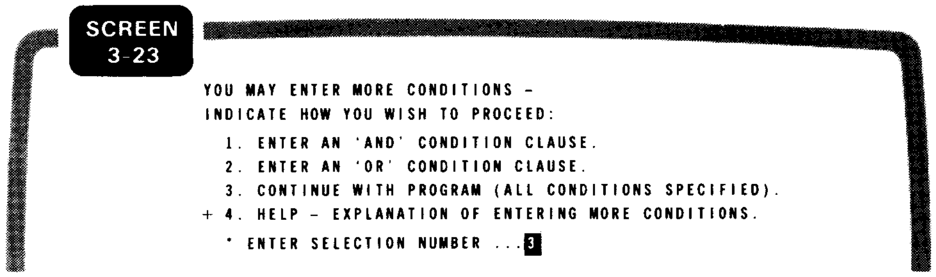


```
                PROVIDE AN 'IF' CONDITION CLAUSE. THE CLAUSE
                MUST INCLUDE A FIELD NAME FROM THE STRUCTURE,
                A RELATIONAL OPERATOR OR STRING OPERATOR, AND
                AN EXPRESSION.
EXAMPLE:
        <FIELD-NAME> <OPERATOR>  <EXPRESSION/CHARACTER STRING>
    IF     BALANCE      EQ       200 + CREDIT
* ENTER REQUIRED VARIABLES ...
        <FIELD-NAME> <OPERATOR> <EXPRESSION/CHARACTER STRING>
    IF    CRLIM___     GT______________________5000________________________
```

Additional options

The next screen (screen 3-23) asks what you want to do next. You have three options and a HELP choice:

SCREEN 3-23



```
YOU MAY ENTER MORE CONDITIONS -
INDICATE HOW YOU WISH TO PROCEED:
  1. ENTER AN 'AND' CONDITION CLAUSE.
  2. ENTER AN 'OR' CONDITION CLAUSE.
  3. CONTINUE WITH PROGRAM (ALL CONDITIONS SPECIFIED).
+ 4. HELP - EXPLANATION OF ENTERING MORE CONDITIONS.
  * ENTER SELECTION NUMBER ...3
```

RETRIEVING DATA

Because the example program only specifies one condition for selection of records, choose number 3, CONTINUE WITH PROGRAM.

NOTE:

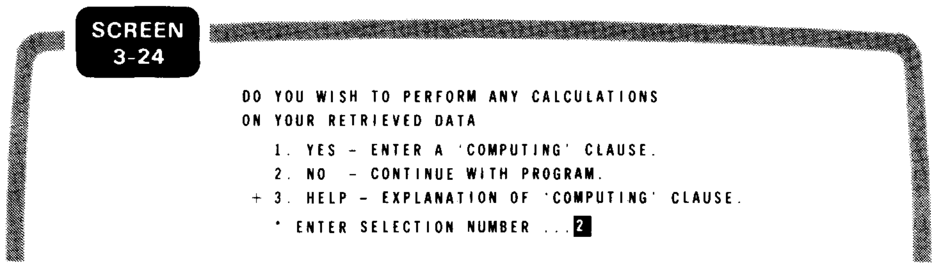
Maximum number of conditions

When you choose to enter conditions, you may enter up to four conditions in tutorial mode.

The next screen (screen 3-24) asks if you want to perform any calculations with your data:

Calculation option

SCREEN 3-24



```
DO YOU WISH TO PERFORM ANY CALCULATIONS
ON YOUR RETRIEVED DATA
     1. YES - ENTER A 'COMPUTING' CLAUSE.
     2. NO  - CONTINUE WITH PROGRAM.
   + 3. HELP - EXPLANATION OF 'COMPUTING' CLAUSE.
     * ENTER SELECTION NUMBER ...2
```

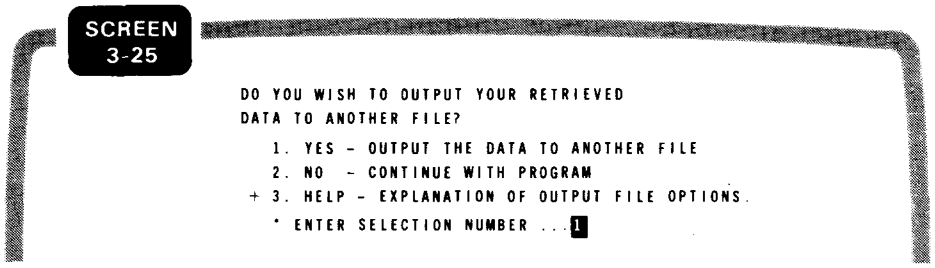
Option denied, continue

Because you don't want to perform any calculations, choose number 2, NO – CONTINUE WITH PROGRAM.

The next screen (screen 3-25) asks if you want to output your data to another file:

Output file option

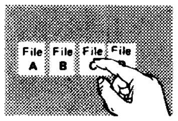
SCREEN 3-25



```
DO YOU WISH TO OUTPUT YOUR RETRIEVED
DATA TO ANOTHER FILE?
     1. YES - OUTPUT THE DATA TO ANOTHER FILE
     2. NO  - CONTINUE WITH PROGRAM
   + 3. HELP - EXPLANATION OF OUTPUT FILE OPTIONS.
     * ENTER SELECTION NUMBER ...1
```

Output to another file decision

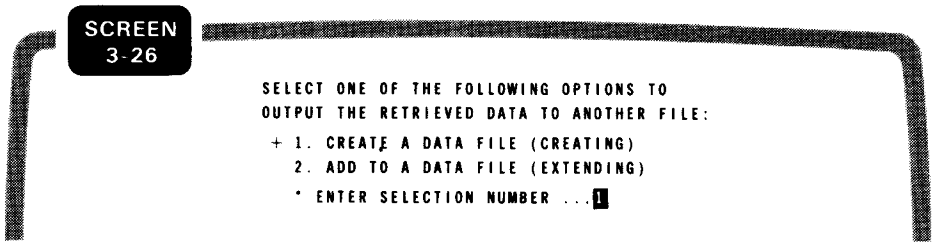
Assume you want to output the records of all customers with a credit limit greater than 5000 dollars into a separate file for special credit mailings, so choose number 1, YES – OUTPUT THE DATA TO ANOTHER FILE.



RETRIEVING DATA

Then you must specify, on the next screen (screen 3-26), whether you are creating a completely new file with this data or extending an existing one.

Type of file for output



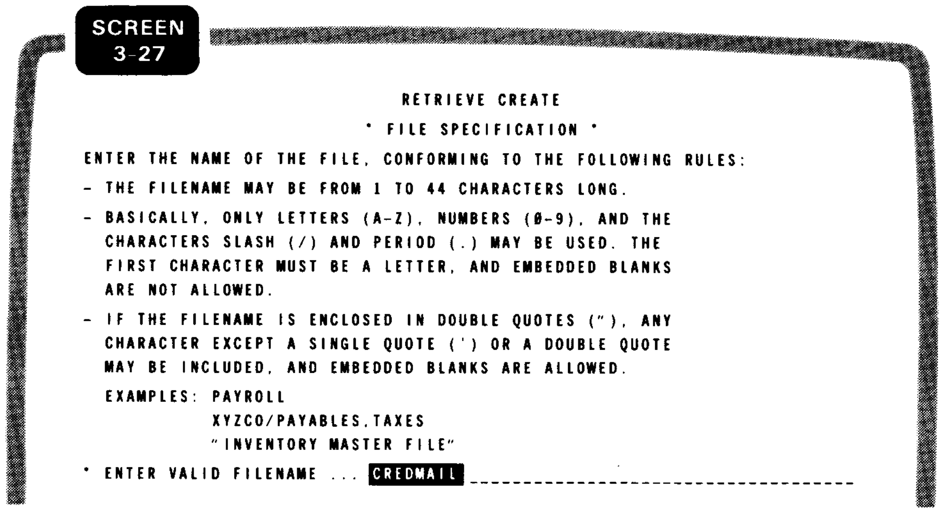
```
SELECT ONE OF THE FOLLOWING OPTIONS TO
OUTPUT THE RETRIEVED DATA TO ANOTHER FILE:
+ 1. CREATE A DATA FILE (CREATING)
  2. ADD TO A DATA FILE (EXTENDING)
  * ENTER SELECTION NUMBER ...1
```

Creating new output file

For this example, assume you're using a completely new file, so choose number 1, CREATE A DATA FILE (CREATING).

Then on screen 3-27, you must name the file you're creating. We'll name it CREDMAIL:

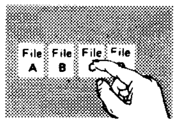
Name new output file



```
                    RETRIEVE CREATE
                  * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
- BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES: PAYROLL
            XYZCO/PAYABLES.TAXES
            "INVENTORY MASTER FILE"
* ENTER VALID FILENAME ... CREDMAIL____________________________________
```

As previously explained (3.5), ESCORT now searches the session and library files to find the structure with the same name as the first eight characters of your file name.

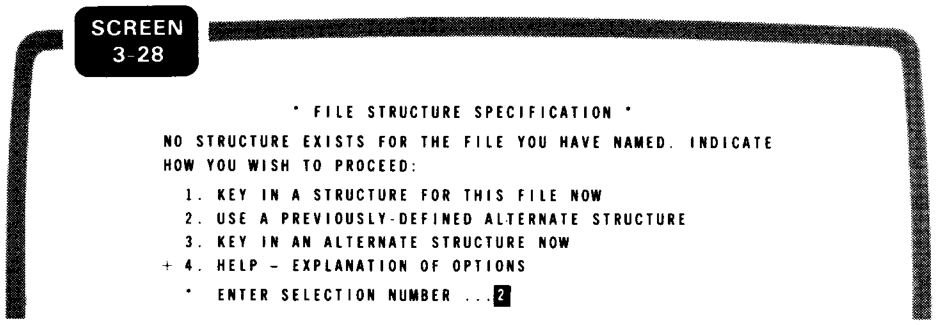
You have not created a structure for this file, so the following screen (screen 3-28) appears.



RETRIEVING DATA

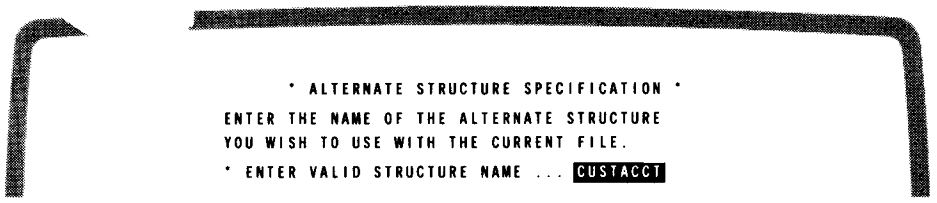
Since the layout of the records in the CUSTACCT file is the same as this file, you may use the same structure. Choose selection 2.

SCREEN 3-28



```
                 * FILE STRUCTURE SPECIFICATION *
NO STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
HOW YOU WISH TO PROCEED:
  1. KEY IN A STRUCTURE FOR THIS FILE NOW
  2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
  3. KEY IN AN ALTERNATE STRUCTURE NOW
+ 4. HELP - EXPLANATION OF OPTIONS
  *  ENTER SELECTION NUMBER ...2
```

The following screen (screen 3-29) appears. Enter the alternate structure name CUSTACCT.

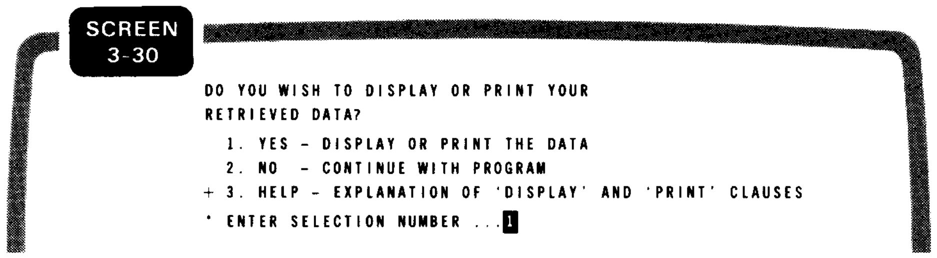


```
        * ALTERNATE STRUCTURE SPECIFICATION *
ENTER THE NAME OF THE ALTERNATE STRUCTURE
YOU WISH TO USE WITH THE CURRENT FILE.
* ENTER VALID STRUCTURE NAME ... CUSTACCT
```

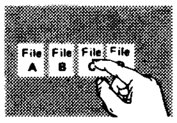
The next screen (screen 3-30) asks if you want to see the data you select:

Disposition of selected data

SCREEN 3-30



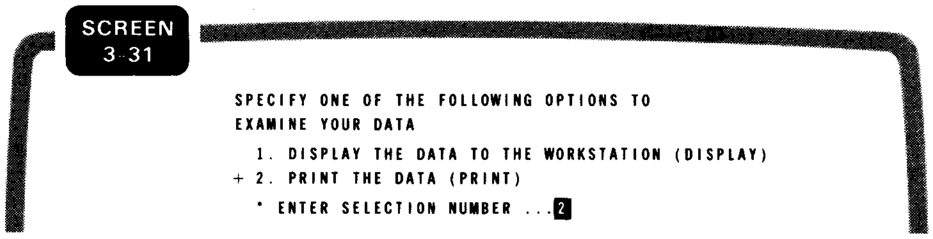
```
DO YOU WISH TO DISPLAY OR PRINT YOUR
RETRIEVED DATA?
  1. YES - DISPLAY OR PRINT THE DATA
  2. NO  - CONTINUE WITH PROGRAM
+ 3. HELP - EXPLANATION OF 'DISPLAY' AND 'PRINT' CLAUSES
* ENTER SELECTION NUMBER ...1
```



RETRIEVING DATA

Because you want to examine the data you select, choose 1, DISPLAY OR PRINT THE DATA. The next screen (screen 3-31), asks how you want to examine your data:

Viewing the data



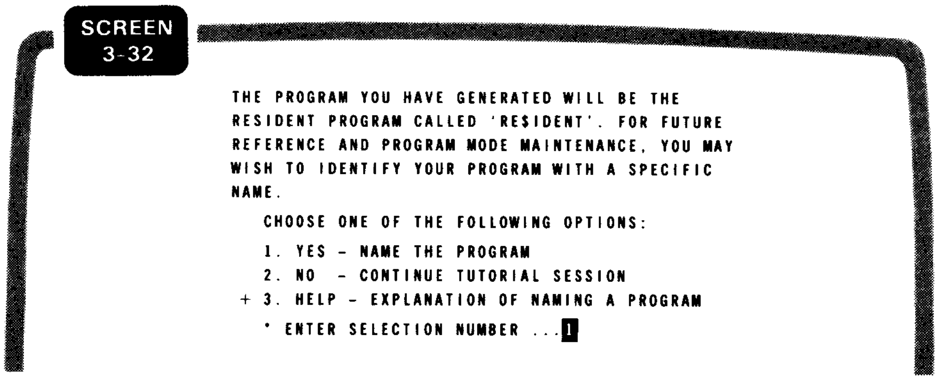
```
SPECIFY ONE OF THE FOLLOWING OPTIONS TO
EXAMINE YOUR DATA
  1. DISPLAY THE DATA TO THE WORKSTATION (DISPLAY)
+ 2. PRINT THE DATA (PRINT)
  * ENTER SELECTION NUMBER ...2
```

Choosing the PRINT option

Assume that you want a listing of all credit customers with a credit limit over 5000 dollars, so choose number 2, PRINT THE DATA.

If you want to name your program, you designate this on screen 3-32.

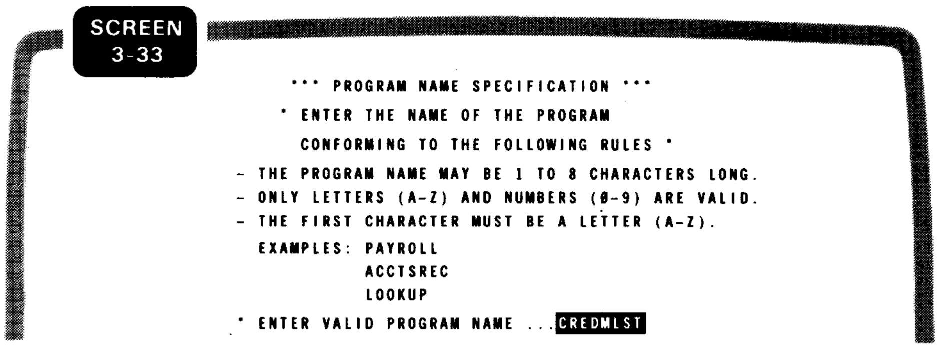
Naming your program



```
THE PROGRAM YOU HAVE GENERATED WILL BE THE
RESIDENT PROGRAM CALLED 'RE$IDENT'. FOR FUTURE
REFERENCE AND PROGRAM MODE MAINTENANCE, YOU MAY
WISH TO IDENTIFY YOUR PROGRAM WITH A SPECIFIC
NAME.
   CHOOSE ONE OF THE FOLLOWING OPTIONS:
   1. YES - NAME THE PROGRAM
   2. NO  - CONTINUE TUTORIAL SESSION
 + 3. HELP - EXPLANATION OF NAMING A PROGRAM
   * ENTER SELECTION NUMBER ...1
```

Then screen 3-33 appears, and you can name your program. For this example, we'll name our program CREDMLST:

Assign program name

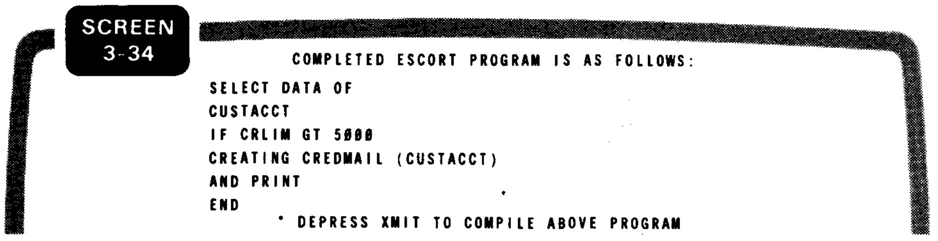


```
      *** PROGRAM NAME SPECIFICATION ***
     * ENTER THE NAME OF THE PROGRAM
       CONFORMING TO THE FOLLOWING RULES *
- THE PROGRAM NAME MAY BE 1 TO 8 CHARACTERS LONG.
- ONLY LETTERS (A-Z) AND NUMBERS (0-9) ARE VALID.
- THE FIRST CHARACTER MUST BE A LETTER (A-Z).
  EXAMPLES: PAYROLL
            ACCTSREC
            LOOKUP
* ENTER VALID PROGRAM NAME ...CREDMLST
```

RETRIEVING DATA

Program listing

The next screen (screen 3-34) lists your ESCORT program:

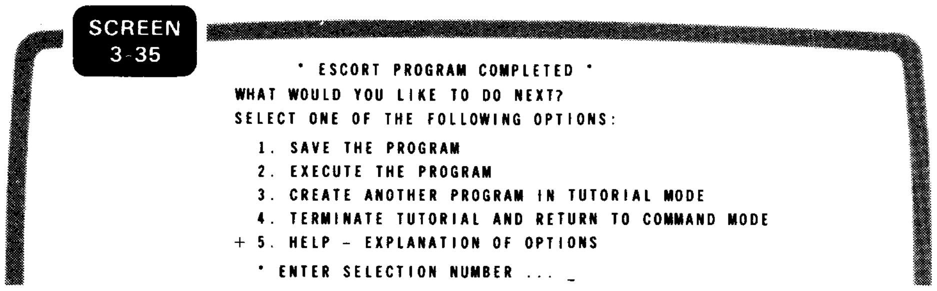


SCREEN 3-34

```
          COMPLETED ESCORT PROGRAM IS AS FOLLOWS:
SELECT DATA OF
CUSTACCT
IF CRLIM GT 5000
CREATING CREDMAIL (CUSTACCT)
AND PRINT
END
        * DEPRESS XMIT TO COMPILE ABOVE PROGRAM
```

Now your program is completed, and you must decide what to do next on the following decision screen (screen 3-35):

Program disposition



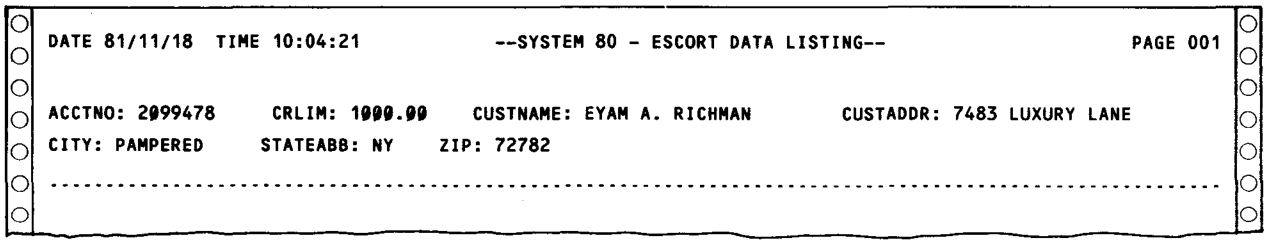
SCREEN 3-35

```
        * ESCORT PROGRAM COMPLETED *
WHAT WOULD YOU LIKE TO DO NEXT?
SELECT ONE OF THE FOLLOWING OPTIONS:
   1. SAVE THE PROGRAM
   2. EXECUTE THE PROGRAM
   3. CREATE ANOTHER PROGRAM IN TUTORIAL MODE
   4. TERMINATE TUTORIAL AND RETURN TO COMMAND MODE
 + 5. HELP - EXPLANATION OF OPTIONS
  * ENTER SELECTION NUMBER ... _
```

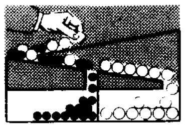
The action initiated by each of these choices is listed under screen 3-10. For your purposes, choose number 2, EXECUTE THE PROGRAM, to execute your program.

Program effects

Because there is only one record in the file CUSTACCT with a credit limit over 5000 dollars, this record is both printed and written to the file CREDMAIL:



```
DATE 81/11/18  TIME 10:04:21          --SYSTEM 80 - ESCORT DATA LISTING--                 PAGE 001

ACCTNO: 2099478    CRLIM: 1000.00    CUSTNAME: EYAM A. RICHMAN      CUSTADDR: 7483 LUXURY LANE
CITY: PAMPERED     STATEABB: NY     ZIP: 72782
----------------------------------------------------------------------------------------------
```



3.10. SORTING DATA

Purpose of SORT DATA

The SORT DATA statement lets you take an unindexed, unordered file and sort it, creating a new sorted file. (We'll explain file types and organization in 6.2.)

The SORT DATA statement gives you two ways to sort a file:

SORT DATA options

1. By using a field name that has been designated as a key in the output CREATING file

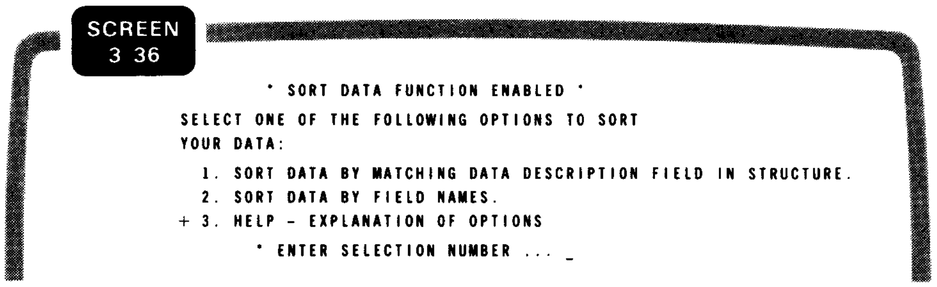
2. By specifying field names from the master file structure

Enter tutorial mode

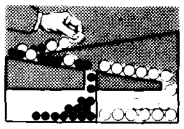
You enter tutorial mode in the method outlined in 3.3. When you get to the function screen (screen 3-2), you choose number 5, SORT DATA.

The next screen that appears (screen 3-36) is the SORT DATA screen, which offers you two options and a HELP facility that explains these options:

Option selection



```
            * SORT DATA FUNCTION ENABLED *
     SELECT ONE OF THE FOLLOWING OPTIONS TO SORT
     YOUR DATA:
        1. SORT DATA BY MATCHING DATA DESCRIPTION FIELD IN STRUCTURE.
        2. SORT DATA BY FIELD NAMES.
     + 3. HELP - EXPLANATION OF OPTIONS
           * ENTER SELECTION NUMBER ... _
```


SORTING DATA

3.11. SORTING DATA BY FIELD NAMES

For this example, use the data file CUSTACCT created in 3.5, and create a new structure CUSTSORT to use with this program statement. CUSTSORT has the same fields as the structure CUSTACCT, but eliminates the sort field designation for ACCTNO. You do this because the key designation makes CUSTACCT an indexed file using the ACCTNO field. Eliminating this key designation lets you reference CUSTACCT as a sequential file. The SORT DATA statement is used primarily to create sequential files. The new structure named CUSTSORT appears:

Structure of CUSTSORT

STRUCTURE OF CUSTSORT

ACCTNO	7N	K1
CRLIM	5V2	
CUSTNAME	20A	
CUSTADDR	20A	
CITY	10A	
STATEABB	2A	
ZIP	5N	

Example

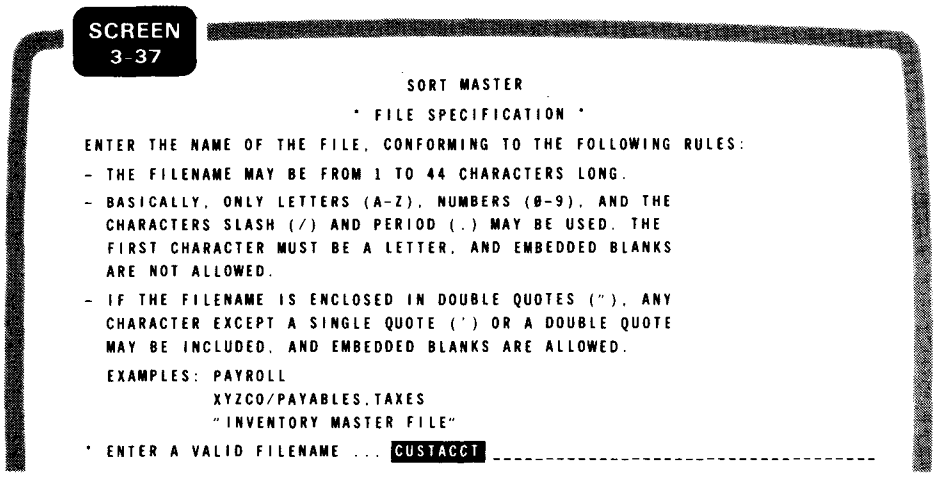
For this example, assume you want to sort the file CUSTACCT (created in 3.5) in alphabetical order by the state where the customer lives (STATEABB) and the account number (ACCTNO).

Choose an option

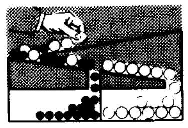
To choose SORT DATA BY FIELD NAMES, key in number 2 on the SORT DATA screen (screen 3-36) and press the **XMIT** key.

The next screen (screen 3-37) asks you the name of the file you want to sort. Because you're using the same file you created in 3.5, key in the name CUSTACCT:

Name the file

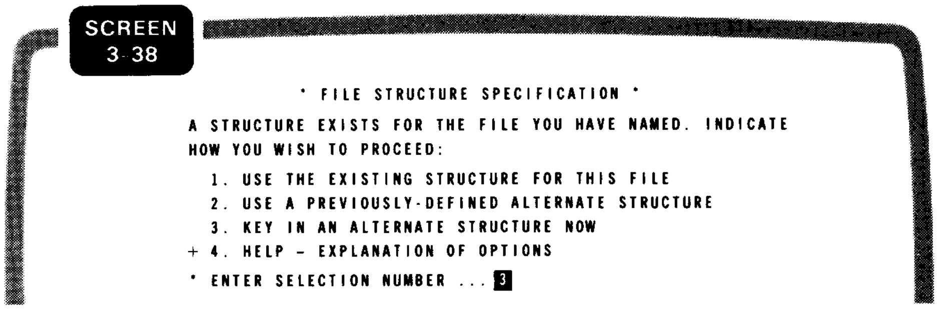

SCREEN 3-37

```
                              SORT MASTER
                        * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
- BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES: PAYROLL
            XYZCO/PAYABLES.TAXES
            "INVENTORY MASTER FILE"
* ENTER A VALID FILENAME ... CUSTACCT_____________________________
```



As previously explained (3.5), ESCORT now searches the session and library files to find the structure with the same name as the first eight characters of your file name.

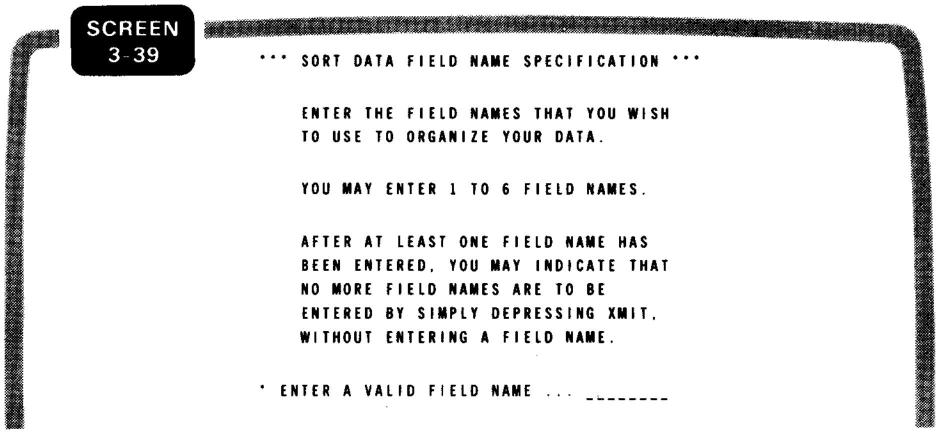
Choose selection 3 on the following screen (screen 3-38) to key in the alternate structure CUSTSORT for the CUSTACCT file.



```
                  * FILE STRUCTURE SPECIFICATION *
A STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
HOW YOU WISH TO PROCEED:
  1. USE THE EXISTING STRUCTURE FOR THIS FILE
  2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
  3. KEY IN AN ALTERNATE STRUCTURE NOW
+ 4. HELP - EXPLANATION OF OPTIONS
* ENTER SELECTION NUMBER ... 3
```

The next screen (screen 3-39) asks you to specify the field names by which you want to sort your data.

Specifying sort criteria



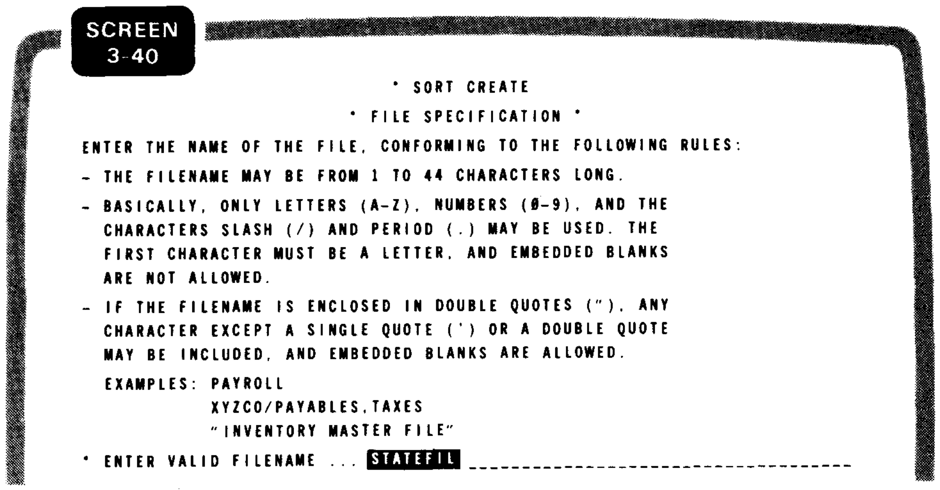
```
*** SORT DATA FIELD NAME SPECIFICATION ***

    ENTER THE FIELD NAMES THAT YOU WISH
    TO USE TO ORGANIZE YOUR DATA.

    YOU MAY ENTER 1 TO 6 FIELD NAMES.

    AFTER AT LEAST ONE FIELD NAME HAS
    BEEN ENTERED, YOU MAY INDICATE THAT
    NO MORE FIELD NAMES ARE TO BE
    ENTERED BY SIMPLY DEPRESSING XMIT,
    WITHOUT ENTERING A FIELD NAME.

* ENTER A VALID FIELD NAME ... ________
```

Entering sort criteria

Because you want to sort your data alphabetically by state and numerically by account number, key in the field name STATEABB and press the **XMIT** key. The screen then reappears and you key in ACCTNO and press the **XMIT** key. The screen reappears again. Because you have no more sort fields to designate, just press the **XMIT** key to continue with your program.

Name the output file

The next screen (screen 3-40) asks you to designate the name of the output file that is to contain the results of your sort:


SCREEN 3-40

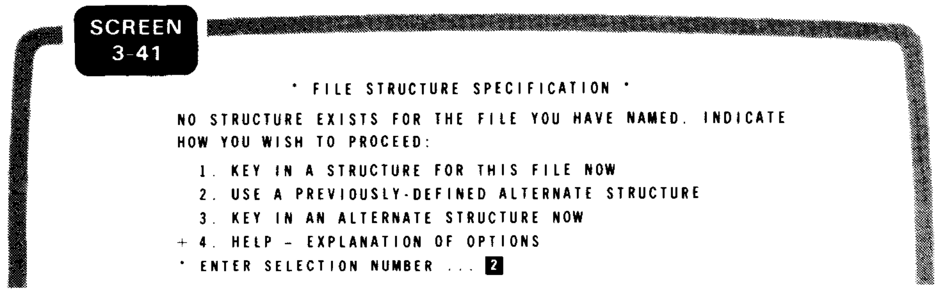
```
                    * SORT CREATE
                * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
- BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES: PAYROLL
            XYZCO/PAYABLES.TAXES
            "INVENTORY MASTER FILE"
* ENTER VALID FILENAME ... STATEFIL ___________________________________
```

Output file named STATEFIL

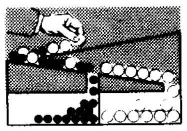
For this example, the new sorted file name is STATEFIL.

As previously explained (3.5), ESCORT now searches the session and library files to find the structure with the same name as the first eight characters of the output file name you designated.

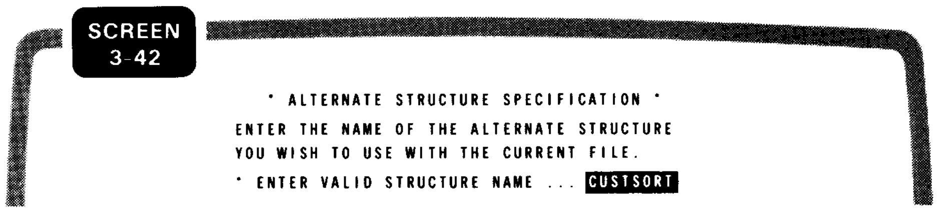
There is no structure for the file STATEFIL, but you may use the CUSTSORT structure because the layouts of the records in the files are the same. Choose selection 2 on the following screen (screen 3-41).


SCREEN 3-41

```
           * FILE STRUCTURE SPECIFICATION *
NO STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
HOW YOU WISH TO PROCEED:
  1. KEY IN A STRUCTURE FOR THIS FILE NOW
  2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
  3. KEY IN AN ALTERNATE STRUCTURE NOW
+ 4. HELP - EXPLANATION OF OPTIONS
* ENTER SELECTION NUMBER ... 2
```



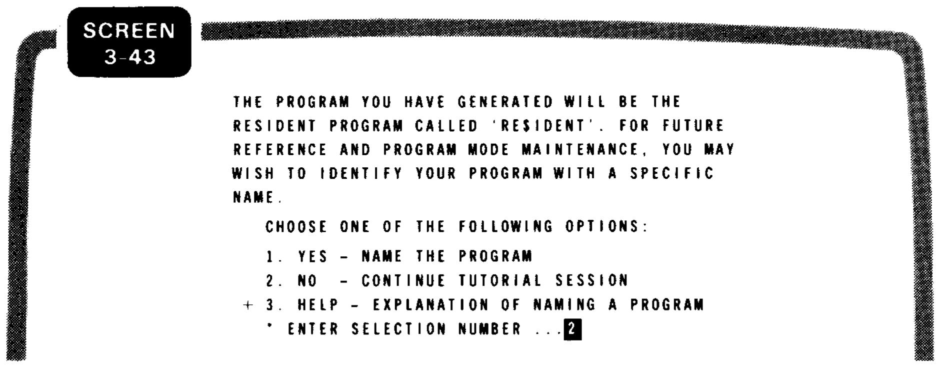
The following screen (screen 3-42) appears. Enter the alternate structure name CUSTSORT.



```
      * ALTERNATE STRUCTURE SPECIFICATION *
    ENTER THE NAME OF THE ALTERNATE STRUCTURE
    YOU WISH TO USE WITH THE CURRENT FILE.
    * ENTER VALID STRUCTURE NAME ... CUSTSORT
```

The next screen (screen 3-43) gives you the option to name the program you have completed or continue with it:

Naming your program



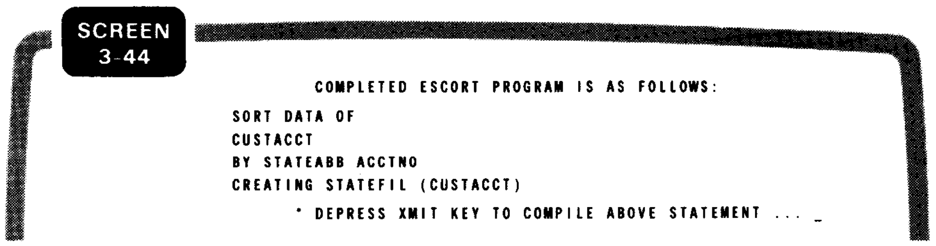
```
    THE PROGRAM YOU HAVE GENERATED WILL BE THE
    RESIDENT PROGRAM CALLED 'RESIDENT'. FOR FUTURE
    REFERENCE AND PROGRAM MODE MAINTENANCE, YOU MAY
    WISH TO IDENTIFY YOUR PROGRAM WITH A SPECIFIC
    NAME.
       CHOOSE ONE OF THE FOLLOWING OPTIONS:
       1. YES - NAME THE PROGRAM
       2. NO  - CONTINUE TUTORIAL SESSION
     + 3. HELP - EXPLANATION OF NAMING A PROGRAM
       * ENTER SELECTION NUMBER ...2
```

Program name option denied

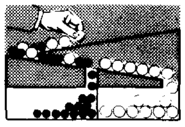
Because you probably will not want to save or reuse this program, choose 2, NO – CONTINUE TUTORIAL SESSION. This is the final tutorial screen for the SORT DATA program statement.

Your completed ESCORT program appears (screen 3-44):

Program listing

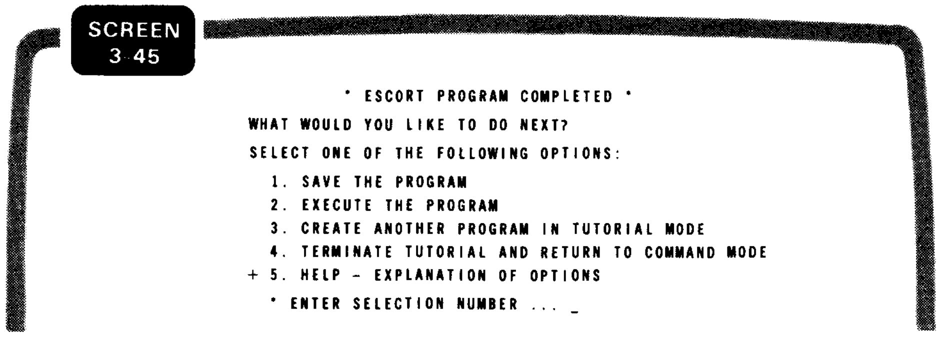


```
          COMPLETED ESCORT PROGRAM IS AS FOLLOWS:
    SORT DATA OF
    CUSTACCT
    BY STATEABB ACCTNO
    CREATING STATEFIL (CUSTACCT)
          * DEPRESS XMIT KEY TO COMPILE ABOVE STATEMENT ... _
```


SORTING DATA

When you press the **XMIT** key to compile your SORT DATA program, the following decision screen appears (screen 3-45):

Compiling your program

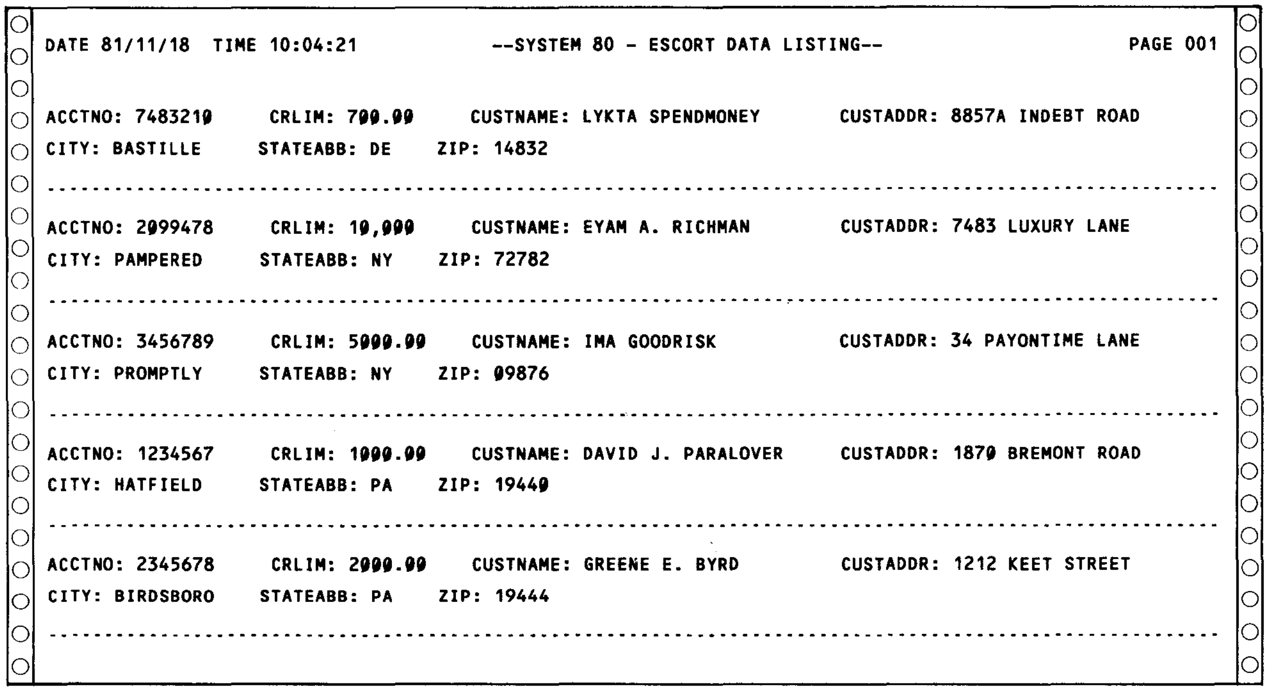

SCREEN 3-45

```
              * ESCORT PROGRAM COMPLETED *
WHAT WOULD YOU LIKE TO DO NEXT?
SELECT ONE OF THE FOLLOWING OPTIONS:
  1. SAVE THE PROGRAM
  2. EXECUTE THE PROGRAM
  3. CREATE ANOTHER PROGRAM IN TUTORIAL MODE
  4. TERMINATE TUTORIAL AND RETURN TO COMMAND MODE
+ 5. HELP - EXPLANATION OF OPTIONS
 * ENTER SELECTION NUMBER ... _
```

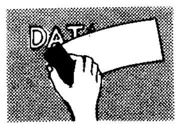
The action initiated by each of these selections is explained under screen 3-10. For your purposes, choose number 2, EXECUTE THE PROGRAM, to execute your program.

Program effects

The result of this program is:



```
DATE 81/11/18  TIME 10:04:21           --SYSTEM 80 - ESCORT DATA LISTING--                PAGE 001

ACCTNO: 7483210     CRLIM: 700.00     CUSTNAME: LYKTA SPENDMONEY       CUSTADDR: 8857A INDEBT ROAD
CITY: BASTILLE      STATEABB: DE      ZIP: 14832
-----------------------------------------------------------------------------------------------------
ACCTNO: 2099478     CRLIM: 10,000     CUSTNAME: EYAM A. RICHMAN        CUSTADDR: 7483 LUXURY LANE
CITY: PAMPERED      STATEABB: NY      ZIP: 72782
-----------------------------------------------------------------------------------------------------
ACCTNO: 3456789     CRLIM: 5000.00    CUSTNAME: IMA GOODRISK           CUSTADDR: 34 PAYONTIME LANE
CITY: PROMPTLY      STATEABB: NY      ZIP: 09876
-----------------------------------------------------------------------------------------------------
ACCTNO: 1234567     CRLIM: 1000.00    CUSTNAME: DAVID J. PARALOVER     CUSTADDR: 1870 BREMONT ROAD
CITY: HATFIELD      STATEABB: PA      ZIP: 19440
-----------------------------------------------------------------------------------------------------
ACCTNO: 2345678     CRLIM: 2000.00    CUSTNAME: GREENE E. BYRD         CUSTADDR: 1212 KEET STREET
CITY: BIRDSBORO     STATEABB: PA      ZIP: 19444
-----------------------------------------------------------------------------------------------------
```



DELETING DATA

3.12. DELETING DATA FROM THE WORKSTATION

Purpose

When you delete data from the workstation, each record (if you have an unkeyed file) or the key fields in each record (if you have a keyed file) are displayed on the workstation. For this example, we're using a keyed file.

Keyed files

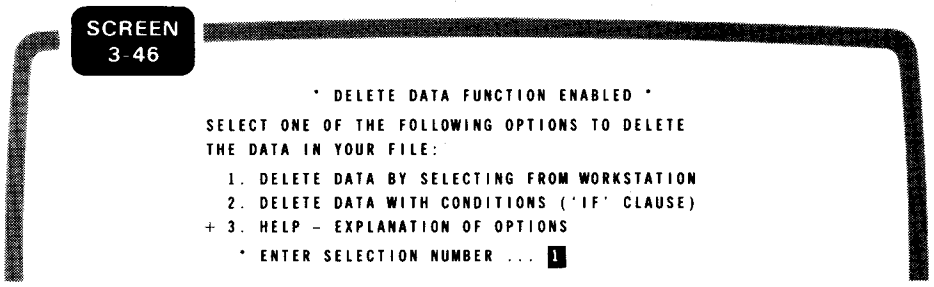
With a keyed file, the key field prompt is displayed on the workstation, you enter the key data for the record you want deleted, and the record is marked for deletion.

You enter tutorial mode in the method outlined in 3.3. When you get to the function screen (screen 3-2), you choose number 4, DELETE DATA.

To choose DELETE DATA BY SELECTING FROM WORKSTATION, key in number 1 on screen 3-46 and press the **XMIT** key:

Selecting method of deletion

SCREEN 3-46

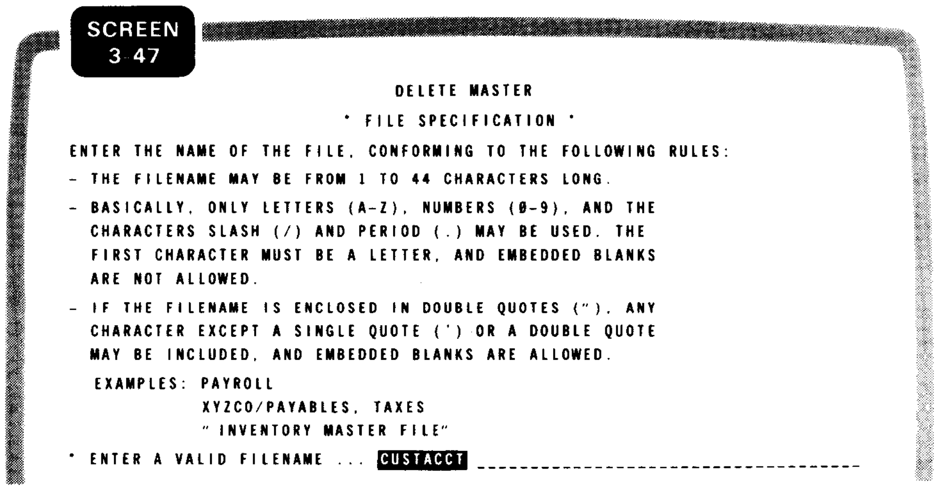


```
              * DELETE DATA FUNCTION ENABLED *
     SELECT ONE OF THE FOLLOWING OPTIONS TO DELETE
     THE DATA IN YOUR FILE:
       1. DELETE DATA BY SELECTING FROM WORKSTATION
       2. DELETE DATA WITH CONDITIONS ('IF' CLAUSE)
     + 3. HELP - EXPLANATION OF OPTIONS
        * ENTER SELECTION NUMBER ... 1
```

The next screen, screen 3-47, asks you to name the file from which you want to delete records. Because we'll use the same file we created in 3.5, key in the name CUSTACCT.

Naming the file

SCREEN 3-47



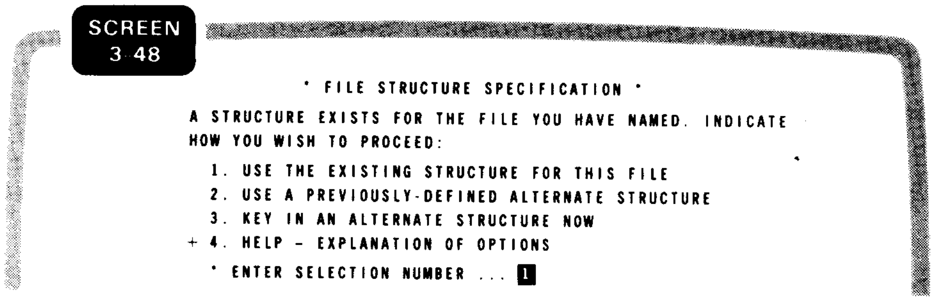
```
                            DELETE MASTER
                        * FILE SPECIFICATION *
ENTER THE NAME OF THE FILE, CONFORMING TO THE FOLLOWING RULES:
- THE FILENAME MAY BE FROM 1 TO 44 CHARACTERS LONG.
- BASICALLY, ONLY LETTERS (A-Z), NUMBERS (0-9), AND THE
  CHARACTERS SLASH (/) AND PERIOD (.) MAY BE USED. THE
  FIRST CHARACTER MUST BE A LETTER, AND EMBEDDED BLANKS
  ARE NOT ALLOWED.
- IF THE FILENAME IS ENCLOSED IN DOUBLE QUOTES ("), ANY
  CHARACTER EXCEPT A SINGLE QUOTE (') OR A DOUBLE QUOTE
  MAY BE INCLUDED, AND EMBEDDED BLANKS ARE ALLOWED.
  EXAMPLES: PAYROLL
            XYZCO/PAYABLES, TAXES
            " INVENTORY MASTER FILE"
* ENTER A VALID FILENAME ... CUSTACCT __________________________________
```

DELETING DATA

ESCORT search

At this point, ESCORT searches the session and library files to find the structure with the same name as the first eight characters of your file name.

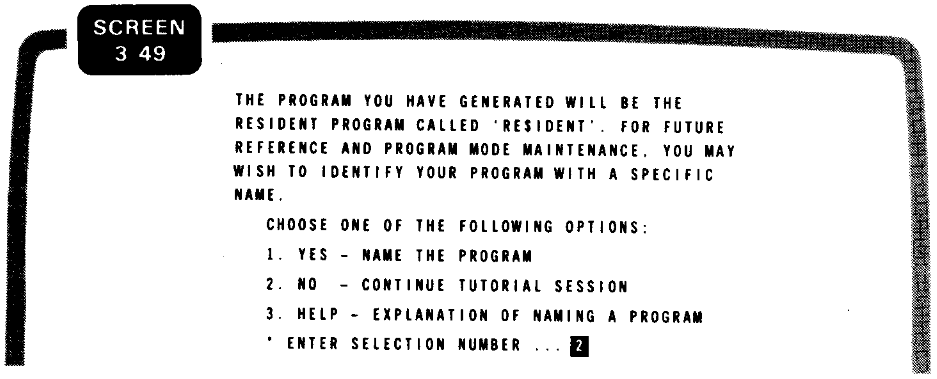
Choose selection 1 on the following screen (screen 3-48) to use the existing CUSTACCT structure.


SCREEN 3-48

```
           * FILE STRUCTURE SPECIFICATION *
A STRUCTURE EXISTS FOR THE FILE YOU HAVE NAMED. INDICATE
HOW YOU WISH TO PROCEED:
  1. USE THE EXISTING STRUCTURE FOR THIS FILE
  2. USE A PREVIOUSLY-DEFINED ALTERNATE STRUCTURE
  3. KEY IN AN ALTERNATE STRUCTURE NOW
+ 4. HELP - EXPLANATION OF OPTIONS
  * ENTER SELECTION NUMBER ... 1
```

Because we chose to delete from the workstation, there are no conditions or clauses to key in, so the next screen asks if you want to name your program. For this example we won't, so we choose 2, NO - CONTINUE TUTORIAL SESSION, on screen 3-49:

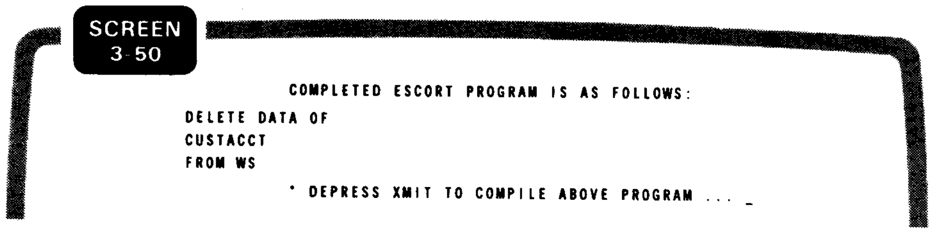
Naming the program


SCREEN 3-49

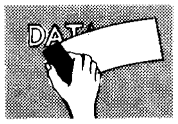
```
THE PROGRAM YOU HAVE GENERATED WILL BE THE
RESIDENT PROGRAM CALLED 'RESIDENT'. FOR FUTURE
REFERENCE AND PROGRAM MODE MAINTENANCE, YOU MAY
WISH TO IDENTIFY YOUR PROGRAM WITH A SPECIFIC
NAME.
   CHOOSE ONE OF THE FOLLOWING OPTIONS:
   1. YES - NAME THE PROGRAM
   2. NO  - CONTINUE TUTORIAL SESSION
   3. HELP - EXPLANATION OF NAMING A PROGRAM
   * ENTER SELECTION NUMBER ... 2
```

The next screen (screen 3-50) displays your completed ESCORT program:

Program compilation


SCREEN 3-50

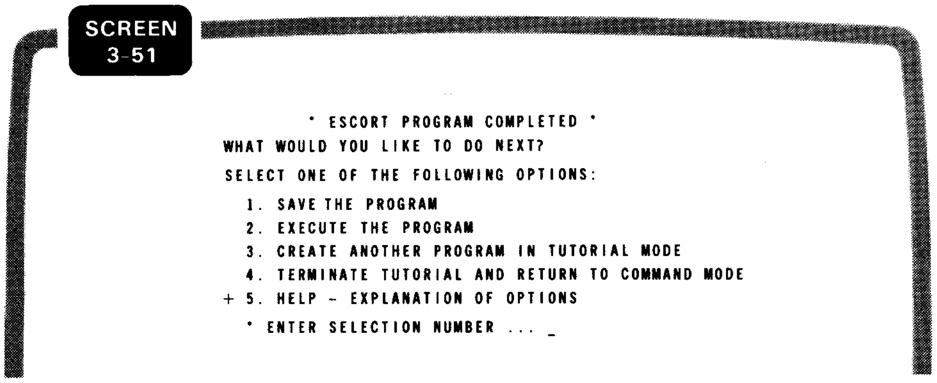
```
             COMPLETED ESCORT PROGRAM IS AS FOLLOWS:
DELETE DATA OF
CUSTACCT
FROM WS
             * DEPRESS XMIT TO COMPILE ABOVE PROGRAM ... _
```



DELETING
DATA

When you press the **XMIT** key to compile your delete data program, the following decision screen (screen 3-51) appears:

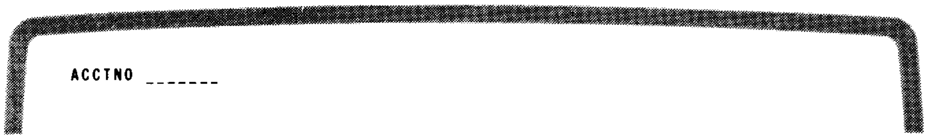
Processing options



```
          * ESCORT PROGRAM COMPLETED *
   WHAT WOULD YOU LIKE TO DO NEXT?
   SELECT ONE OF THE FOLLOWING OPTIONS:
      1. SAVE THE PROGRAM
      2. EXECUTE THE PROGRAM
      3. CREATE ANOTHER PROGRAM IN TUTORIAL MODE
      4. TERMINATE TUTORIAL AND RETURN TO COMMAND MODE
   +  5. HELP - EXPLANATION OF OPTIONS
     * ENTER SELECTION NUMBER ... _
```

Program operation

The action initiated by each of these options is listed under screen 3-10. For your purposes, choose number 2, EXECUTE THE PROGRAM, to execute your program. Because your file is a keyed file and its key field is ACCTNO, the following prompt is displayed on your workstation screen:



```
ACCTNO _______
```

Deleting the record

Suppose you want to delete the record for customer Lykta Spendmoney. The account number for that customer's record is 7483210. You key in that account number and press the **XMIT** key, and the record is marked for deletion. Then the prompt reappears, and you can key in the account number for the next record you want deleted.

Terminating deletion operation

When you've entered all the account numbers for the records you want to delete, press the **FUNCTION** and **F4** keys to specify the end of input to the system.

Now that we've gone over some of the file operations you can perform in tutorial mode, let's look at how these same operations function in program mode.

PART
3

CONCEPTS OF PROGRAM MODE

4. Interactive Program Mode

4.1. WHY USE PROGRAM MODE?

Advantages

Program mode is the heart of ESCORT, and it's the mode you'll use once you master tutorial mode. Unlike tutorial mode, which is limited in scope, program mode provides full ESCORT capability for all your problem-solving needs. It's faster than tutorial mode, and it provides you with two methods of program entry:

Entry methods

1. The screen menu method

2. The direct entry method

Screen menu method

Advantages

The screen menu method is similar to tutorial mode, but not as elementary. In tutorial mode, ESCORT creates your program; in program mode, you're in charge. ESCORT guides you in statement syntax and organization by providing the menu selections, but it's up to you to pick the program statements and clauses needed to produce the desired results in your program. The screen menu method allows full ESCORT capability, but eliminates the need to memorize statement syntax and organization because you "write" your program from a series of screen menus. The screen menu method is described in 4.3 through 4.21.

Direct entry method
Advantages

Selection

Intended user

The direct entry method has the same ESCORT capability as the screen menu method, but it's faster if you are familiar enough with ESCORT not to need menu guidance. You choose this method from the PROGRAM STATEMENTS menu (Figure 1-1b) by selecting choice number 6, DIRECT ENTRY OF PROGRAM STATEMENTS - FREE FORMAT. A blank screen then appears, and you key in program statements and clauses to meet your needs. You must be experienced with ESCORT programming before attempting this method because you're responsible for statement syntax and organization. Direct entry of program statements is discussed further in 4.22.

Summary

The following subsections discuss both methods and show you how to use them. The rest is up to you. For ease of use and minimum effort, the screen menu method is ideal. For speed, the direct entry method is best. But regardless of which method you use, ESCORT provides you with the same ease of use and versatility. Only you can determine how to use it to your best advantage.

4.2. THINGS YOU NEED TO DO BEFORE ENTERING PROGRAM MODE

Regardless of which program mode method you use, there are three steps to follow before you can create a data file. This is important now because you must perform step 1 before you enter ESCORT. For our purposes, we'll assume your system is active and you've logged on. You must perform the following steps:

Step 1: Make Sure the Files You Need Are Cataloged

Handling uncataloged files

Files are cataloged using OS/3 job control statements. In most cases, the data files you use will already be allocated and cataloged. However, if you are in an ESCORT session and try to create a file that isn't cataloged, you'll get an error message. To catalog a file, press the **FUNCTION** and **SYS MODE** keys. This puts you into system mode and you can run the job control stream to catalog the file. To get back to your ESCORT session, press the **FUNCTION** and **WS MODE** keys. For more information on cataloging files, see 6.3.

Step 2: Enter ESCORT

Entering ESCORT

Enter ESCORT by keying in the word ESCORT and pressing the **XMIT** key. When you do this, the introductory screen (Figure 1-1a) appears, welcoming you to ESCORT.

Pick the mode you want by keying in a selection number and pressing the **XMIT** key. Then, the command selection menu (Figure 1-1a) appears, and you choose the command type you need.

If you need a review on calling ESCORT, see 2.3.

Step 3: Create a Structure

All data files must have a structure. To create a file structure, you must access the ESCORT structure processor by choosing number 3, STRUCTURE COMMANDS, on the command selection menu

Structure entry

(Figure 1-1a). Then, when the STRUCTURE COMMANDS menu appears, choose number 1, ENTER STRUCTURE, and enter a structure.

If you need additional assistance or explanations, refer to Section 6.

Example

For purposes of example in this section, use the following structure, named INVNTORY:

INVNTORY structure

STRUCTURE OF INVNTORY			EXPLANATION OF FIELD NAMES
PRODNO	7N	K1	*Product number*
PRODESC	25A		*Product description*
UNITSIZE	10A		*Unit size*
UNITCOST	4V2		*Cost of unit*
MARKUP	1V2		*Retail markup*
RETPRICE	5V2		*Retail selling price*
QONHAND	5N		*Quantity on hand*
QONORDER	5N		*Quantity on order*
REORDERL	2N		*Reorder level*
QUANDISC	1V2		*Quantity discount percentage*
VENDNAME	20A		*Vendor name for reorder*
VENDADDR	35A		*Vendor address for reorder*
VENDZIP	5N		*Vendor zip code*

Saving the structure

While you're still on the STRUCTURE COMMANDS menu, it's a good idea to save the structure you just created. This ensures that it is in the ESCORT library for future use. To save a structure, choose number 3 on the STRUCTURE COMMANDS menu, name the structure you want saved (INVNTORY), and press the **XMIT** key.

Structure entry termination

When you're through creating and saving your structure, terminate the structure processor by choosing number 10, TERMINATE STRUCTURE PROCESSOR, on the STRUCTURE COMMANDS menu (Figure 1-1a).

Command selection

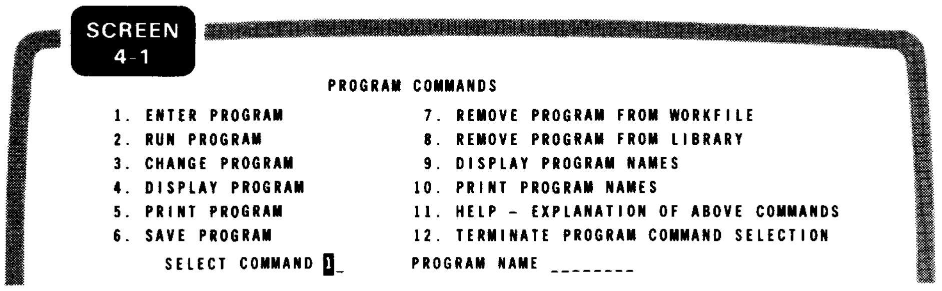
The command selection menu (Figure 1-1a) then reappears, and you can choose another command type.

4.3. PROGRAM MODE - THE SCREEN MENU METHOD

Command choice

The previous paragraph left you at the COMMANDS SELECTION menu. Since you already created a structure for your file, it's time to enter a program. Select choice number 2, PROGRAM COMMANDS, from the COMMAND SELECTION menu and press the **XMIT** key. Remember, unless you're told otherwise, you must press the **XMIT** key after every entry to send your responses to the system.

Now the following menu (screen 4-1) is displayed:



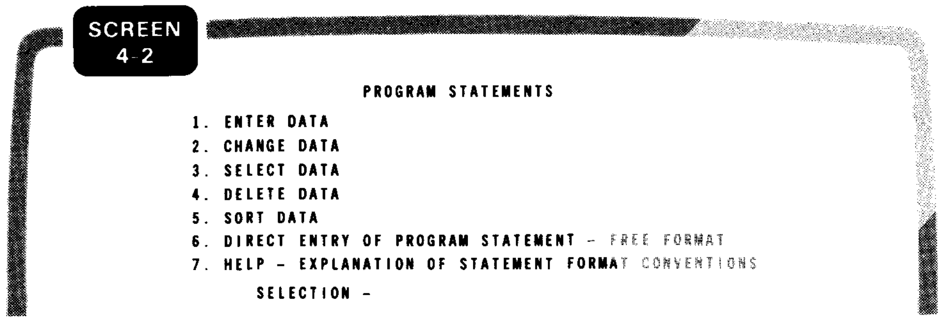
```
                    PROGRAM COMMANDS
1. ENTER PROGRAM              7. REMOVE PROGRAM FROM WORKFILE
2. RUN PROGRAM                8. REMOVE PROGRAM FROM LIBRARY
3. CHANGE PROGRAM             9. DISPLAY PROGRAM NAMES
4. DISPLAY PROGRAM           10. PRINT PROGRAM NAMES
5. PRINT PROGRAM             11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE PROGRAM              12. TERMINATE PROGRAM COMMAND SELECTION
     SELECT COMMAND 1_          PROGRAM NAME ________
```

Entering a program

Default program name

Because you want to enter a program, choose number 1, ENTER PROGRAM. Assigning a program name is not required unless you plan to save the program. If you don't name it, ESCORT assigns it the name RE$IDENT. The examples in this section use RE$IDENT except for the second CHANGE DATA example (CHANGE DATA by replacing) and the SELECT DATA program example, which you'll save.

Then the following menu (screen 4-2) appears:



```
                PROGRAM STATEMENTS
1. ENTER DATA
2. CHANGE DATA
3. SELECT DATA
4. DELETE DATA
5. SORT DATA
6. DIRECT ENTRY OF PROGRAM STATEMENT - FREE FORMAT
7. HELP - EXPLANATION OF STATEMENT FORMAT CONVENTIONS
     SELECTION -
```

Program statements selection

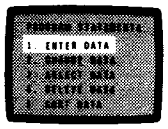
This screen lists each type of program statement, and you pick the one you want.

Purpose and use of section

At this point, your ESCORT program mode session can take many paths, depending on which program statement you choose. The specific uses and complete formats for the program statements appear in Section 7. For now, concern yourself only with the sequence of the screen menus and the choices you make to get the desired clauses and statements for your program.

Examples of each type of program statement are given in subsequent subsections, beginning with the ENTER DATA program statement.

It is important to note that any data shown as entered into files or displayed or printed as output is variable data keyed into our files for purposes of example. Obviously, your data will be different, but just assume that all the data we keyed in exists on our file and is output as such.



4.4. USING THE SCREEN MENU METHOD TO CREATE TWO KINDS OF DATA ENTRY PROGRAMS

Using the structure created in 4.2, you can write a data entry program by using one of two data entry methods:

Data entry methods

- CREATING – to create new files
- EXTENDING – to extend existing files

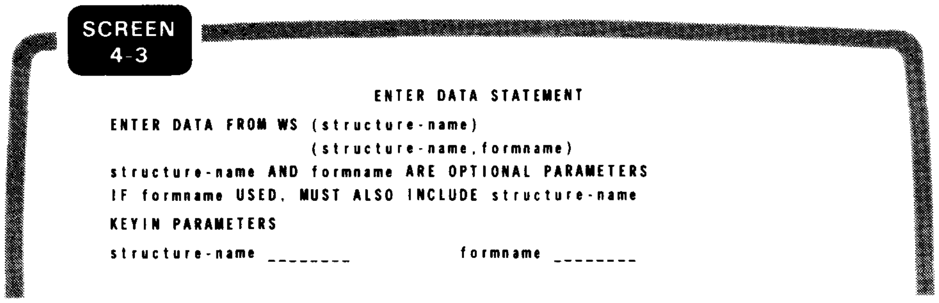
4.5. CREATING A NEW DATA FILE

Entering data

Since this is your first program mode session, you'll want to create a data file, so key in number 1, ENTER DATA, on the PROGRAM STATEMENTS menu (screen 4-2).

Since you picked ENTER DATA, the screen containing the format of the basic ENTER DATA program statement is displayed (screen 4-3). (We'll give you more information on the basic ENTER DATA statement in 7.5.)

ENTER DATA program statement


SCREEN 4-3

```
                        ENTER DATA STATEMENT
ENTER DATA FROM WS (structure-name)
                   (structure-name,formname)
structure-name AND formname ARE OPTIONAL PARAMETERS
IF formname USED, MUST ALSO INCLUDE structure-name
KEYIN PARAMETERS
structure-name ________          formname ________
```

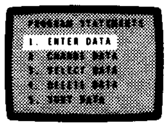
Parameter keyin

Structure-name parameter

You'll notice that both the *structure-name* and the *structure-name,formname* parameters are optional. The *structure-name* parameter is only optional, however, when the name of the structure is the same as the first eight characters of the file name. When the names are not the same, you must assign a structure name, or you'll get an error.

Structure-name,formname parameter

The *structure-name,formname* parameter is used only when you want to designate a form on which to input your data. In this case, the *structure-name* parameter is also required.

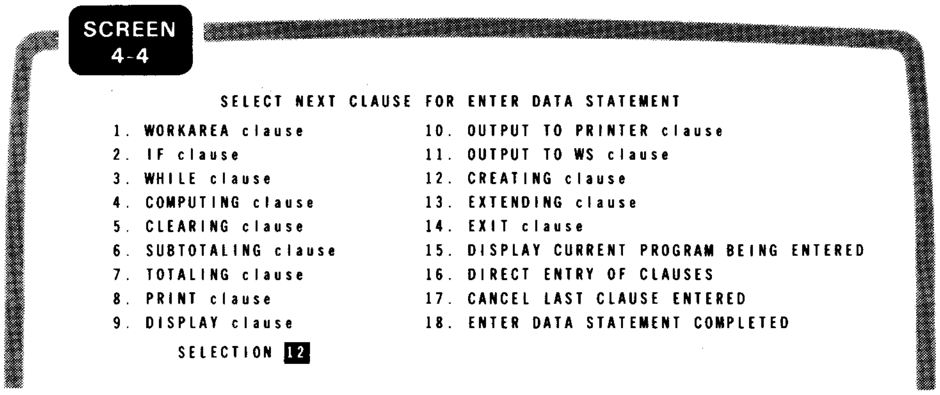


File and structure names

Because the sample structure is named INVNTORY, name your file INVNTORY also. Because they both have the same name, you don't have to specify the *structure-name* parameter. Press the **XMIT** key to move to the next screen.

Clause decisions

The next screen (screen 4–4) lists all the clauses applicable to the ENTER DATA program statement:

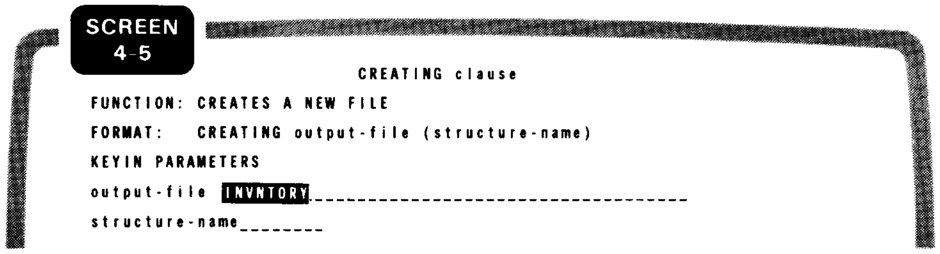

SCREEN 4-4

```
            SELECT NEXT CLAUSE FOR ENTER DATA STATEMENT
 1. WORKAREA clause            10. OUTPUT TO PRINTER clause
 2. IF clause                  11. OUTPUT TO WS clause
 3. WHILE clause               12. CREATING clause
 4. COMPUTING clause           13. EXTENDING clause
 5. CLEARING clause            14. EXIT clause
 6. SUBTOTALING clause         15. DISPLAY CURRENT PROGRAM BEING ENTERED
 7. TOTALING clause            16. DIRECT ENTRY OF CLAUSES
 8. PRINT clause               17. CANCEL LAST CLAUSE ENTERED
 9. DISPLAY clause             18. ENTER DATA STATEMENT COMPLETED
      SELECTION 12
```

You pick the clause you want, and ESCORT displays it so you can key in the necessary parameters.

CREATING clause

Because this is your first session and your first data file, choose number 12, CREATING clause. ESCORT displays the screen (screen 4–5) containing the format for the CREATING clause.

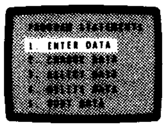

SCREEN 4-5

```
                      CREATING clause
FUNCTION: CREATES A NEW FILE
FORMAT:   CREATING output-file (structure-name)
KEYIN PARAMETERS
output-file INVNTORY___________________________________
structure-name________
```

←

Specifying output file and structure names

For this clause, the only thing you are required to do is key in the output file name, which is INVNTORY, and press the **XMIT** key. You don't need to key in the structure name because it's the same as the file name.

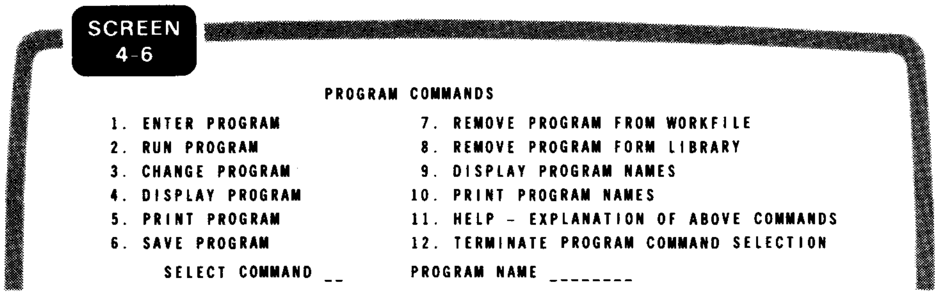


Statement completion

ESCORT keeps track of what you key in and again displays the clauses screen (screen 4-4) so you can pick another clause. Because you're creating a new data file, you don't need any more clauses, so choose number 18, ENTER DATA STATEMENT COMPLETED.

Additional file processing

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu again (screen 4-6). Now you can choose what you want to do with your program:


SCREEN 4-6

```
                         PROGRAM COMMANDS
 1. ENTER PROGRAM                  7. REMOVE PROGRAM FROM WORKFILE
 2. RUN PROGRAM                    8. REMOVE PROGRAM FORM LIBRARY
 3. CHANGE PROGRAM                 9. DISPLAY PROGRAM NAMES
 4. DISPLAY PROGRAM               10. PRINT PROGRAM NAMES
 5. PRINT PROGRAM                 11. HELP - EXPLANATION OF ABOVE COMMANDS
 6. SAVE PROGRAM                  12. TERMINATE PROGRAM COMMAND SELECTION
      SELECT COMMAND __           PROGRAM NAME ________
```

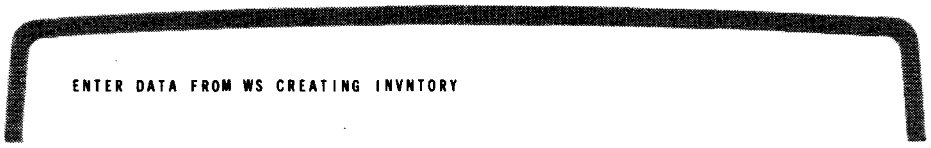
Saving a program

If you think you'll want to reuse your program (which is unlikely in the case of file creation), save it by choosing number 6, SAVE PROGRAM. ESCORT then displays the corresponding sequence of screens you must complete to save your program.

Program display

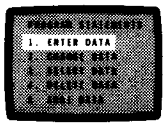
To get an idea of what your completed program looks like, choose number 4, DISPLAY PROGRAM. Because you used the system-supplied name (RE$IDENT), you don't have to key in a program name. Just pick your selection and press the **XMIT** key, and ESCORT displays your data entry program:

Example program



```
ENTER DATA FROM WS CREATING INVNTORY
```

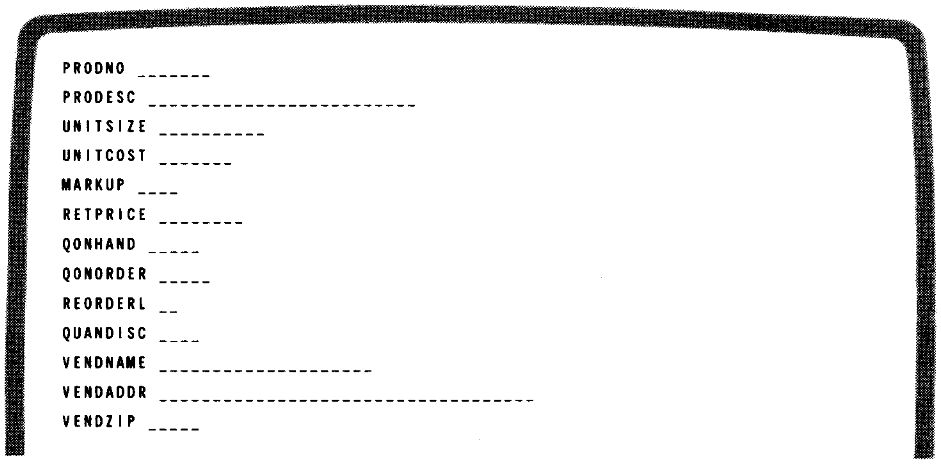
To get back to the PROGRAM COMMANDS menu (screen 4-6), press the **XMIT** key.



Program execution

Because your program is complete and you don't want to save it, choose command number 2, RUN PROGRAM, and press the **XMIT** key. Because you used the system-supplied name (RE$IDENT), you don't have to key in a program name. ESCORT executes your program and, because you did not designate a specific input form, displays the default record template corresponding to the fields in your structure. You use this template for data entry. The default record template for the structure INVNTORY appears like this:

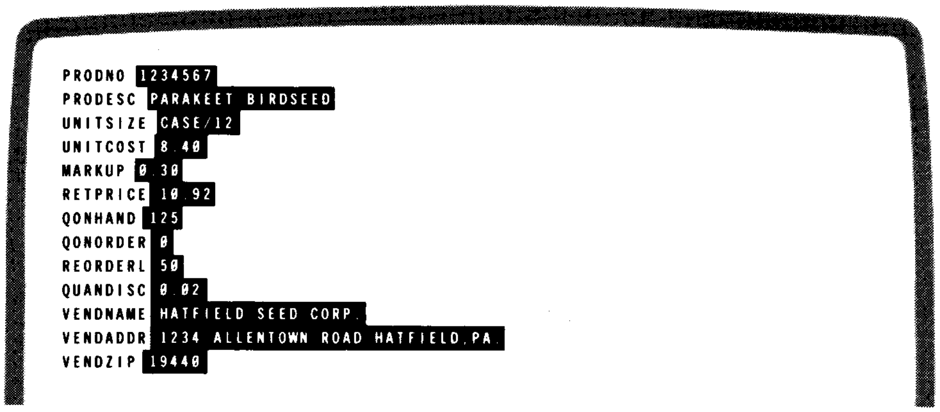
Default record format for data entry



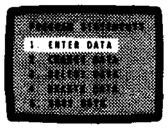
```
PRODNO _______
PRODESC _________________________
UNITSIZE __________
UNITCOST _______
MARKUP ____
RETPRICE ________
QONHAND _____
QONORDER _____
REORDERL __
QUANDISC ____
VENDNAME ____________________
VENDADDR ___________________________________
VENDZIP _____
```

Assuming you enter the following data into this template, your data record now looks like this:

Sample input



```
PRODNO 1234567
PRODESC PARAKEET BIRDSEED
UNITSIZE CASE/12
UNITCOST 8.40
MARKUP 0.30
RETPRICE 10.92
QONHAND 125
QONORDER 0
REORDERL 50
QUANDISC 0.02
VENDNAME HATFIELD SEED CORP.
VENDADDR 1234 ALLENTOWN ROAD HATFIELD,PA.
VENDZIP 19440
```



Input termination

By pressing the **XMIT** key, you can move from record to record until you finish entering data into your file. When your last record is entered, press the **FUNCTION** and **F4** keys to designate the end of the input file.

Terminate program mode

Terminate ESCORT session

At this point, you now have a data file named INVNTORY, and ESCORT is displaying the PROGRAM COMMANDS menu (screen 4-6). Your processing is complete, so choose number 12, TERMINATE PROGRAM COMMAND SELECTION. This displays the command selection menu and you choose number 1, TERMINATE CURRENT ESCORT SESSION.

Now your data entry program is complete, you have a file named INVNTORY, and you can use any of the other program statements to manipulate or change this file.

4.6. EXTENDING AN EXISTING DATA FILE

Purpose

Now that you have a data file, assume you want to add more records to it. You use the ENTER DATA program statement and the EXTENDING clause. The process is the same as in 4.5 (where you used the CREATING clause) up to the clauses screen, where you choose the EXTENDING clause rather than the CREATING clause.

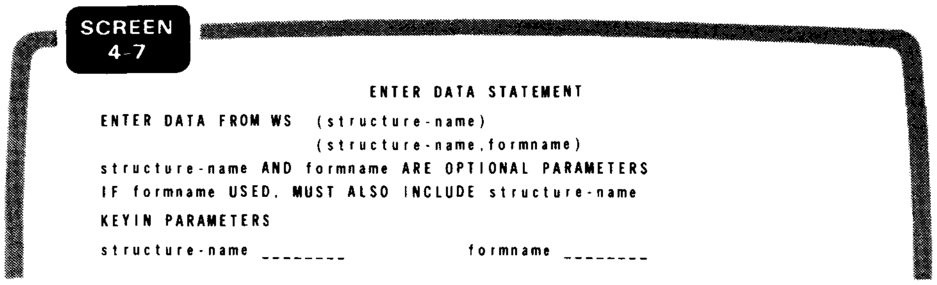
We'll assume you've chosen the ENTER PROGRAM command from the PROGRAM COMMANDS menu (screen 4-1) and are now at the PROGRAM STATEMENTS menu (screen 4-2).

Choosing ENTER DATA statement

To begin, choose number 1, ENTER DATA, on the PROGRAM STATEMENTS menu (screen 4-2). The screen containing the basic ENTER DATA statement (screen 4-7) is displayed, and you fill in the required parameters. The required parameters were outlined earlier in 4.5 (screen 4-3).

ENTER DATA statement

SCREEN 4-7



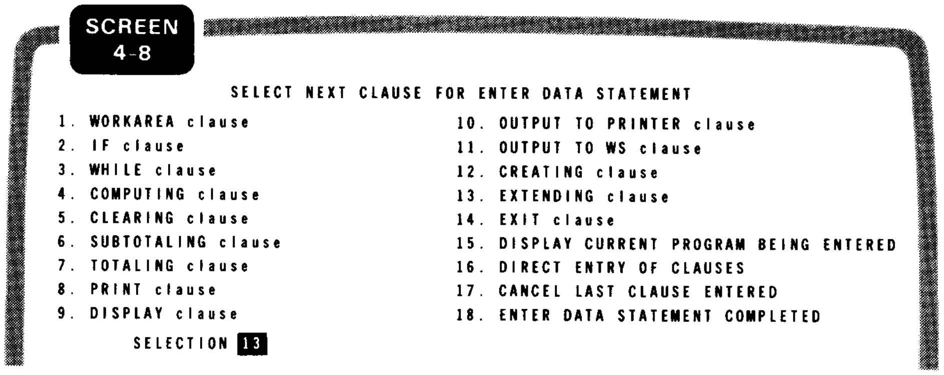
```
                              ENTER DATA STATEMENT
ENTER DATA FROM WS  (structure-name)
                    (structure-name,formname)
structure-name AND formname ARE OPTIONAL PARAMETERS
IF formname USED, MUST ALSO INCLUDE structure-name
KEYIN PARAMETERS
structure-name ________           formname ________
```

File and structure name entry

Because your file and structure name are the same, you don't need to key in any parameters on this screen (they will be named in the EXTENDING clause), so press the **XMIT** key to proceed.

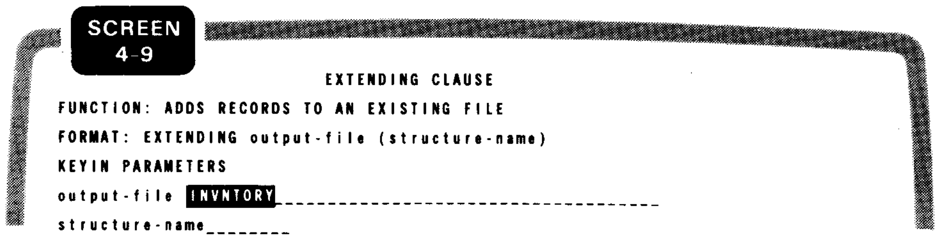
The next screen (screen 4-8) lists the clauses applicable to the ENTER DATA program statement:

Selecting the EXTENDING clause


SCREEN 4-8

```
                    SELECT NEXT CLAUSE FOR ENTER DATA STATEMENT
 1. WORKAREA clause                    10. OUTPUT TO PRINTER clause
 2. IF clause                          11. OUTPUT TO WS clause
 3. WHILE clause                       12. CREATING clause
 4. COMPUTING clause                   13. EXTENDING clause
 5. CLEARING clause                    14. EXIT clause
 6. SUBTOTALING clause                 15. DISPLAY CURRENT PROGRAM BEING ENTERED
 7. TOTALING clause                    16. DIRECT ENTRY OF CLAUSES
 8. PRINT clause                       17. CANCEL LAST CLAUSE ENTERED
 9. DISPLAY clause                     18. ENTER DATA STATEMENT COMPLETED
          SELECTION 13
```

Because you want to add records to an existing file, choose number 13, EXTENDING clause. ESCORT then displays screen 4-9, which contains the format for the EXTENDING clause:


SCREEN 4-9

```
                          EXTENDING CLAUSE
FUNCTION: ADDS RECORDS TO AN EXISTING FILE
FORMAT: EXTENDING output-file (structure-name)
KEYIN PARAMETERS
output-file INVNTORY___________________________________
structure-name________
```

Naming the output file

For this clause, you're only required to key in the output file name, which is INVNTORY, and press the **XMIT** key. You don't need to key in the structure name because it's the same as the file name.

Additional clause choices

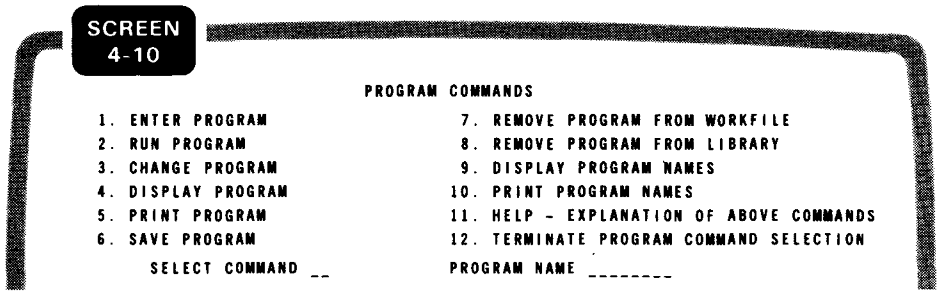
ESCORT keeps track of what you key in and redisplays the clauses screen (screen 4-8) so you can pick another clause.

Statement completion

Because you're extending an existing data file, you don't need any more clauses, so choose number 18, ENTER DATA STATEMENT COMPLETED.

Program compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4–10):



```
                         PROGRAM COMMANDS
1. ENTER PROGRAM                      7. REMOVE PROGRAM FROM WORKFILE
2. RUN PROGRAM                        8. REMOVE PROGRAM FROM LIBRARY
3. CHANGE PROGRAM                     9. DISPLAY PROGRAM NAMES
4. DISPLAY PROGRAM                   10. PRINT PROGRAM NAMES
5. PRINT PROGRAM                     11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE PROGRAM                      12. TERMINATE PROGRAM COMMAND SELECTION
     SELECT COMMAND __               PROGRAM NAME ________
```

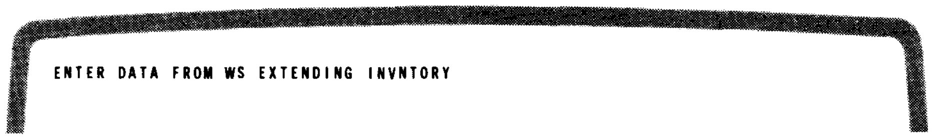
Saving a program

If you think you'll want to reuse your program, you can save it by choosing number 6, SAVE PROGRAM, and ESCORT displays the corresponding sequence of screens you must complete to save your program.

Program display

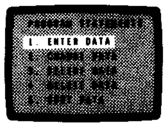
If you want to get an idea of what your completed program looks like, choose number 4, DISPLAY PROGRAM. Because you used the system-supplied name (RE$IDENT), you don't have to key in a program name. Just pick your selection and press the **XMIT** key, and ESCORT displays your file extension program:

Example program



```
ENTER DATA FROM WS EXTENDING INVNTORY
```

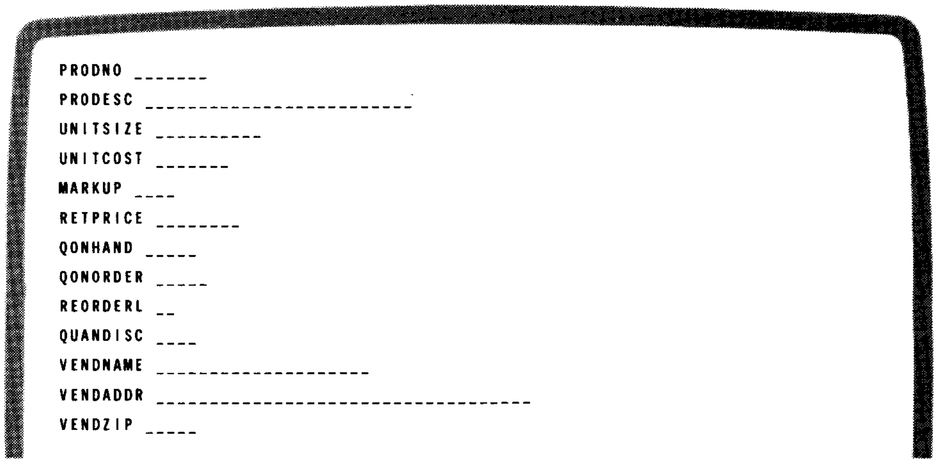
To get back to the PROGRAM COMMANDS menu (screen 4–10), press the **XMIT** key.



Program execution

Because your program is complete and you don't want to save it, choose command number 2, RUN PROGRAM. Because you used the system-supplied name, RE$IDENT, you don't need to key in a program name, so just press the **XMIT** key. ESCORT executes your program and, because you did not designate a specific input form, displays the default record template corresponding to the fields in your structure:

Default record format for data entry



```
PRODNO _______
PRODESC _________________________
UNITSIZE __________
UNITCOST _______
MARKUP ____
RETPRICE ________
QONHAND _____
QONORDER _____
REORDERL __
QUANDISC ____
VENDNAME ____________________
VENDADDR ____________________________________
VENDZIP _____
```

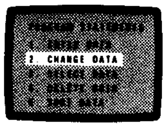
Entering data records

Then, just as when you created your file, you enter records into this template. By pressing the **XMIT** key, you can move from record to record until you finish entering data into your file. When your last record is entered, press the **FUNCTION** and **F4** keys to designate the end of the input file.

File extension complete

At this point, you have successfully added records to your data file INVNTORY, and ESCORT is displaying the PROGRAM COMMANDS menu (screen 4-10). Your processing is complete, so choose number 12, TERMINATE PROGRAM COMMAND SELECTION. This displays the command selection menu, and you choose number 1, TERMINATE CURRENT ESCORT SESSION.

Now your file extension program is complete, and you can perform other operations on your file by using the other program statements.



4.7. USING THE SCREEN MENU METHOD TO CREATE THREE KINDS OF DATA CHANGE PROGRAMS

Purpose

Now that you created a data file (INVNTORY) and entered data into it, you can write a data change program by using the CHANGE DATA program statement.

There are three ways to use the CHANGE DATA statement:

Data change methods

- Change data from the workstation
- Change data by assigning new field values
- Change data by replacing with matching file data

4.8. CHANGE DATA FROM THE WORKSTATION

Example definition

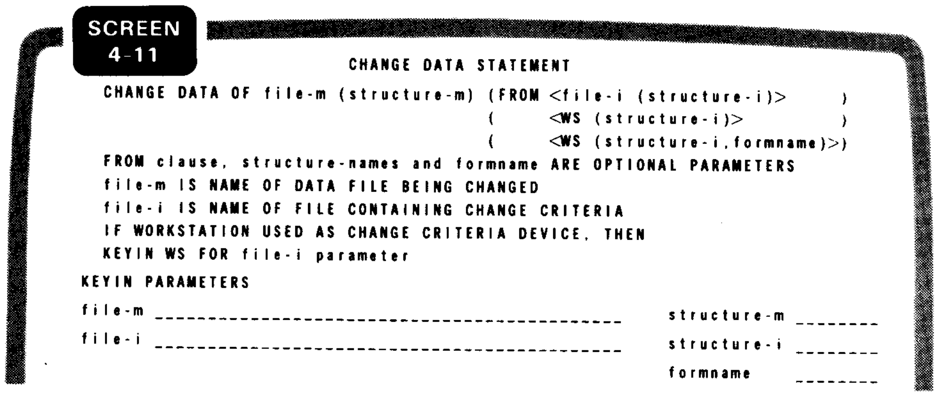
Using the data file and structure you created in 4.5, your first data change program changes the vendor address (VENDADDR) and vendor zip code (VENDZIP) fields of your INVNTORY file.

We'll assume you've chosen the ENTER PROGRAM command from the PROGRAM COMMANDS menu (screen 4-1) and are now at the PROGRAM STATEMENTS menu (screen 4-2).

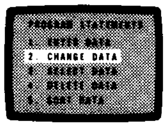
Initiation of CHANGE DATA

To get the CHANGE DATA function, choose number 2 on the PROGRAM STATEMENTS menu (screen 4-2).

This displays screen 4-11, which contains the format of the basic CHANGE DATA statement. (We'll give you more information on the basic CHANGE DATA statement in 7.8.)



```
                              CHANGE DATA STATEMENT
  CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>        )
                                      (     <WS (structure-i)>            )
                                      (     <WS (structure-i,formname)>)
  FROM clause, structure-names and formname ARE OPTIONAL PARAMETERS
  file-m IS NAME OF DATA FILE BEING CHANGED
  file-i IS NAME OF FILE CONTAINING CHANGE CRITERIA
  IF WORKSTATION USED AS CHANGE CRITERIA DEVICE, THEN
  KEYIN WS FOR file-i parameter
KEYIN PARAMETERS
file-m ________________________________________   structure-m ________
file-i ________________________________________   structure-i ________
                                                  formname    ________
```



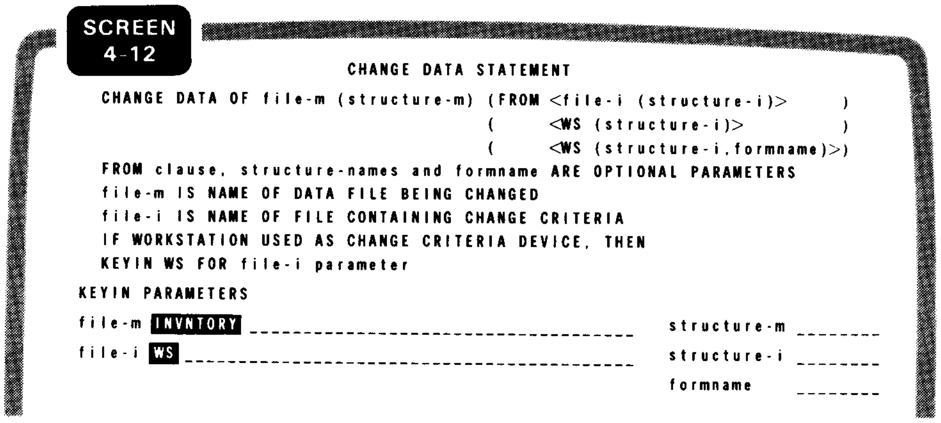
Required parameters

Notice that the only required parameter is *file-m*. This names the file you want to change (INVNTORY, for this example). Again, the *structure-name* parameter is optional. You use it only when the structure name is different from the first eight characters of the file name or when you use the *structure-name,formname* parameter to designate a customized input form.

Parameter keyin

For this example, fill in the *file-m* parameter with the file name INVNTORY and, because you're using the workstation for input, key in WS for the *file-i* parameter. Your CHANGE DATA screen now looks like screen 4-12:

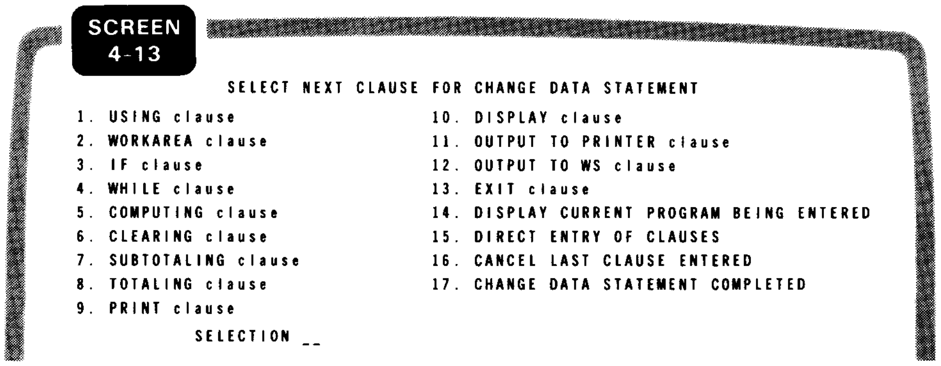
SCREEN 4-12

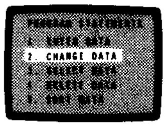

```
                         CHANGE DATA STATEMENT
  CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>       )
                                      (     <WS (structure-i)>          )
                                      (     <WS (structure-i,formname)>)
  FROM clause, structure-names and formname ARE OPTIONAL PARAMETERS
  file-m IS NAME OF DATA FILE BEING CHANGED
  file-i IS NAME OF FILE CONTAINING CHANGE CRITERIA
  IF WORKSTATION USED AS CHANGE CRITERIA DEVICE, THEN
  KEYIN WS FOR file-i parameter
KEYIN PARAMETERS
file-m INVNTORY ___________________________________   structure-m ________
file-i WS ________________________________________   structure-i ________
                                                      formname    ________
```

Clause decisions

Once your CHANGE DATA screen is completed, press the **XMIT** key to proceed. The next screen (screen 4-13) lists all the clauses applicable to the CHANGE DATA statement:

SCREEN 4-13


```
                 SELECT NEXT CLAUSE FOR CHANGE DATA STATEMENT
1. USING clause                  10. DISPLAY clause
2. WORKAREA clause               11. OUTPUT TO PRINTER clause
3. IF clause                     12. OUTPUT TO WS clause
4. WHILE clause                  13. EXIT clause
5. COMPUTING clause              14. DISPLAY CURRENT PROGRAM BEING ENTERED
6. CLEARING clause               15. DIRECT ENTRY OF CLAUSES
7. SUBTOTALING clause            16. CANCEL LAST CLAUSE ENTERED
8. TOTALING clause               17. CHANGE DATA STATEMENT COMPLETED
9. PRINT clause
          SELECTION __
```

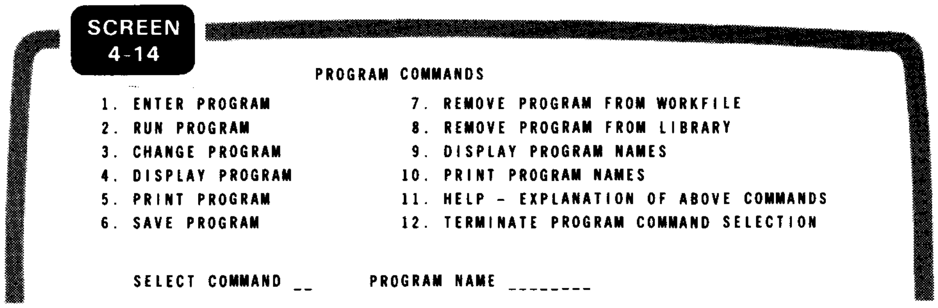


Statement completion

Because you're using the workstation for input, you really don't need any other clauses, so pick number 17, CHANGE DATA STATEMENT COMPLETED.

Statement compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4-14). Now you can choose what you want to do with your program.

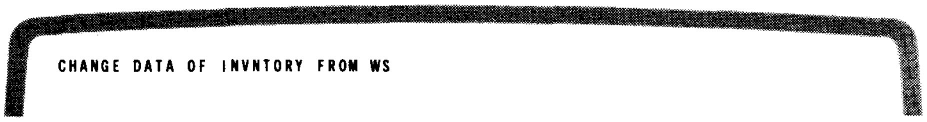

SCREEN 4-14

```
                          PROGRAM COMMANDS
  1. ENTER PROGRAM               7. REMOVE PROGRAM FROM WORKFILE
  2. RUN PROGRAM                 8. REMOVE PROGRAM FROM LIBRARY
  3. CHANGE PROGRAM              9. DISPLAY PROGRAM NAMES
  4. DISPLAY PROGRAM            10. PRINT PROGRAM NAMES
  5. PRINT PROGRAM              11. HELP - EXPLANATION OF ABOVE COMMANDS
  6. SAVE PROGRAM               12. TERMINATE PROGRAM COMMAND SELECTION

     SELECT COMMAND __      PROGRAM NAME ________
```

Again, as with all programs, you have the option to save your program. Because you're only changing addresses of specific vendors, it's unlikely you'll want to use it again, so don't save it.

Program display

To get an idea of what your finished program looks like, select number 4, DISPLAY PROGRAM, and press the **XMIT** key. Then ESCORT displays your program:

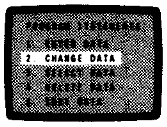


```
CHANGE DATA OF INVNTORY FROM WS
```

To get back to the PROGRAM COMMANDS menu (screen 4-14), press the **XMIT** key.

Program execution

Then choose number 2, RUN PROGRAM, on screen 4-14 to execute your program.



At this point, depending upon what type of file you're attempting to change, keyed or unkeyed, your program can take two paths. The program for this example is for a keyed file.

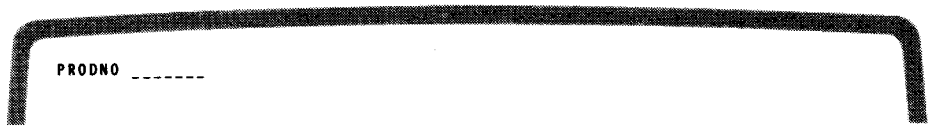
Keyed files

- If Your File Is Keyed

Example

When you use a CHANGE DATA FROM WS program on a keyed file, a prompt containing the key field name and blank spaces appears. The key field name for this example is product number (PRODNO).

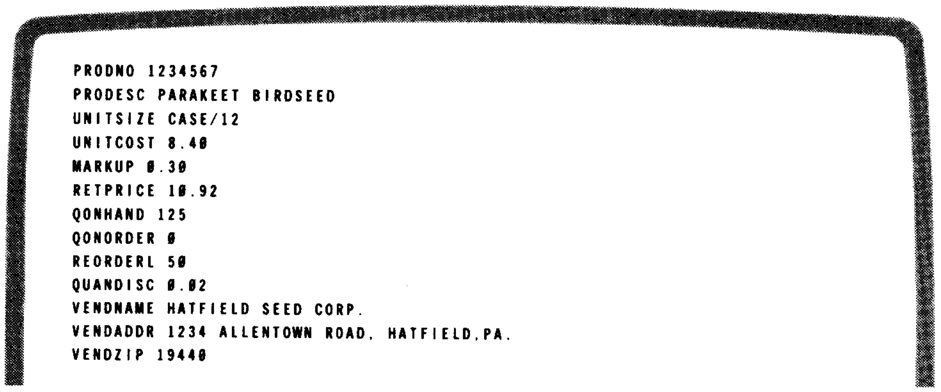
Keyed file display prompt



```
PRODNO _______
```

Key in the product number of the record you want to change (e.g., 1234567), and ESCORT displays the rest of the record. The record for this product number appears:

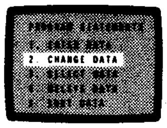
Keyed file record display



```
PRODNO 1234567
PRODESC PARAKEET BIRDSEED
UNITSIZE CASE/12
UNITCOST 8.40
MARKUP 0.30
RETPRICE 10.92
QONHAND 125
QONORDER 0
REORDERL 50
QUANDISC 0.02
VENDNAME HATFIELD SEED CORP.
VENDADDR 1234 ALLENTOWN ROAD, HATFIELD,PA.
VENDZIP 19440
```

File updated

Now you simply key in the data you're changing for the fields VENDADDR and VENDZIP and press the **XMIT** key, and your file is automatically updated.



Unkeyed files

■ If Your File Is Unkeyed

Three types

There are three types of unkeyed files:

1. Unkeyed, unordered file with no matching criteria
2. Unkeyed, unordered file with matching criteria
3. Unkeyed, sorted file with matching criteria

When you use:

– Unordered File with No Matching Criteria

ESCORT action

ESCORT displays every record on the workstation for change.

– Unordered File with Matching Criteria

ESCORT action

ESCORT displays the prompt for each field designated as a matching field on the workstation. Then you key in the data for the prompted field of the record you want to change.

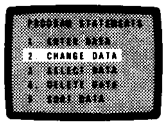
ESCORT reads the file from beginning to end and displays each record that matches the prompt field, so you can make your changes.

When you complete your changes, press the **FUNCTION** and **F4** keys to specify the end of input to your system.

– Sorted File with Matching Criteria

ESCORT action

ESCORT displays the matching field prompt on the workstation, and you key in the data for that field. ESCORT then searches the file for a matching record. Then the record is displayed on the workstation for change. After you make all your changes, press the **FUNCTION** and **F4** keys to specify the end of input to your system.



4.9. CHANGE DATA BY ASSIGNING NEW FIELD VALUES

Program objective

Using the data file and structure (INVNTORY) created in 4.5, your second data change program changes the value of the field MARKUP. Also, it computes a new retail price (RETPRICE) by multiplying the fields UNITCOST and MARKUP and adding the value of UNITCOST. It then prints a report, using the standard system formatting PRINT clause.

We'll assume you've chosen the ENTER PROGRAM command from the PROGRAM COMMANDS menu (screen 4–1) and are now at the PROGRAM STATEMENTS menu (screen 4–2).

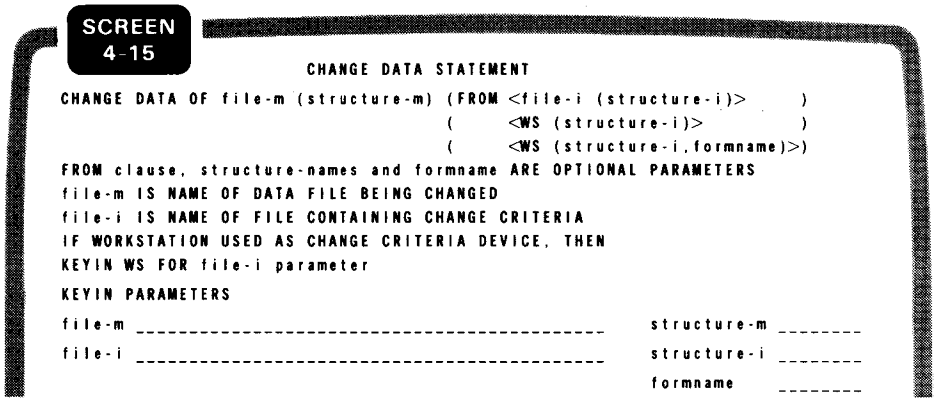
Initiation of CHANGE DATA

To get the CHANGE DATA function, choose number 2 on the PROGRAM STATEMENTS menu (screen 4–2).

This displays screen 4–15, which contains the format of the basic CHANGE DATA statement. (We'll give you more information on the basic CHANGE DATA statement in 7.8.)

CHANGE DATA statement

SCREEN 4-15



```
                          CHANGE DATA STATEMENT
CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>        )
                                    (     <WS (structure-i)>            )
                                    (     <WS (structure-i,formname)>)
FROM clause, structure-names and formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING CHANGED
file-i IS NAME OF FILE CONTAINING CHANGE CRITERIA
IF WORKSTATION USED AS CHANGE CRITERIA DEVICE, THEN
KEYIN WS FOR file-i parameter
KEYIN PARAMETERS
file-m ________________________________________    structure-m ________
file-i ________________________________________    structure-i ________
                                                   formname    ________
```

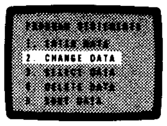
Parameter keyin

Notice that the only required parameter is the *file-m* specification. This names the file you want to change (INVNTORY, for this example).

Structure-name parameter

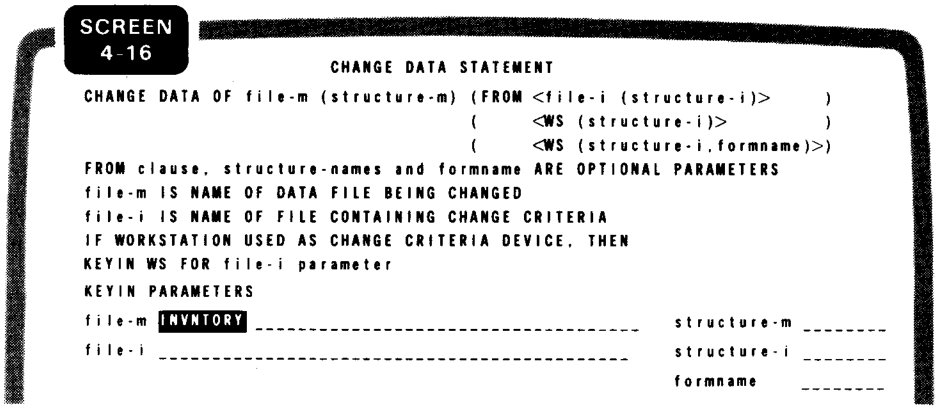
Structure-name,formname parameter

Again, the *structure-name* parameter is optional. You use it only when the structure name is different from the first eight characters of the file name or when you use the *structure-name,formname* parameter to designate a customized input form.



For this example, fill in the *file-m* parameter with the file name INVNTORY. Your CHANGE DATA screen now looks like screen 4-16.

Completed CHANGE DATA screen

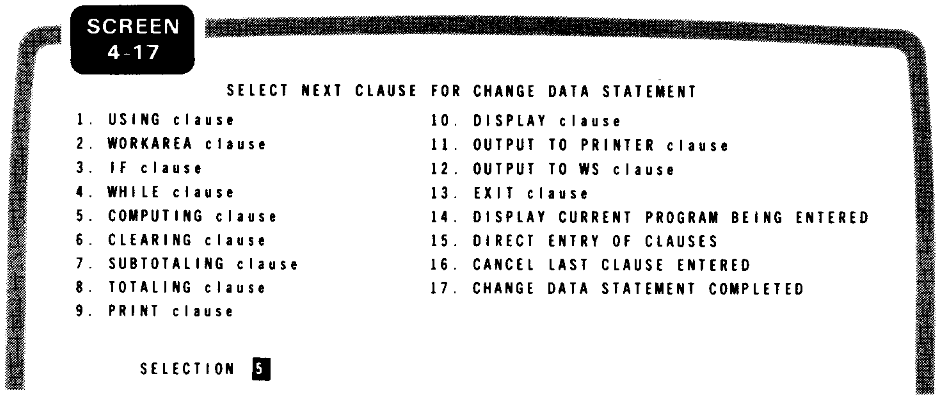

SCREEN 4-16

```
                      CHANGE DATA STATEMENT
CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>      )
                                    (     <WS (structure-i)>          )
                                    (     <WS (structure-i,formname)>)
FROM clause, structure-names and formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING CHANGED
file-i IS NAME OF FILE CONTAINING CHANGE CRITERIA
IF WORKSTATION USED AS CHANGE CRITERIA DEVICE, THEN
KEYIN WS FOR file-i parameter
KEYIN PARAMETERS
file-m INVNTORY ___________________________________   structure-m ________
file-i ____________________________________________   structure-i ________
                                                      formname    ________
```

Clause decisions

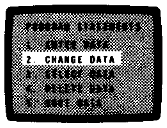
Once your CHANGE DATA screen is completed, press the **XMIT** key to proceed. The next screen (screen 4-17) lists all the clauses applicable to the CHANGE DATA statement:

Selecting the COMPUTING clause


SCREEN 4-17

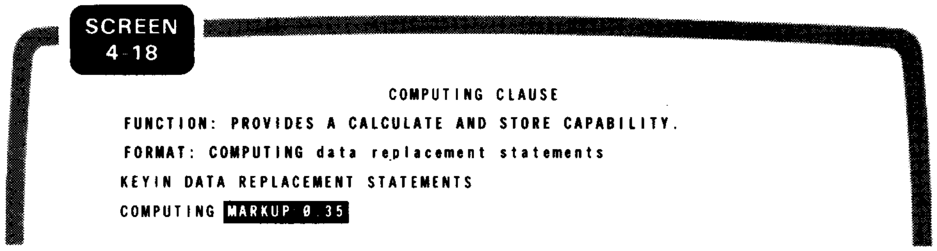
```
              SELECT NEXT CLAUSE FOR CHANGE DATA STATEMENT
1. USING clause              10. DISPLAY clause
2. WORKAREA clause           11. OUTPUT TO PRINTER clause
3. IF clause                 12. OUTPUT TO WS clause
4. WHILE clause              13. EXIT clause
5. COMPUTING clause          14. DISPLAY CURRENT PROGRAM BEING ENTERED
6. CLEARING clause           15. DIRECT ENTRY OF CLAUSES
7. SUBTOTALING clause        16. CANCEL LAST CLAUSE ENTERED
8. TOTALING clause           17. CHANGE DATA STATEMENT COMPLETED
9. PRINT clause

     SELECTION 5
```

Because you want to specify a new value for the field MARKUP, choose number 5, COMPUTING clause.



COMPUTING clauses

The screen containing the format for the COMPUTING clause appears (screen 4-18), and you key in your data replacement statement.


SCREEN 4-18

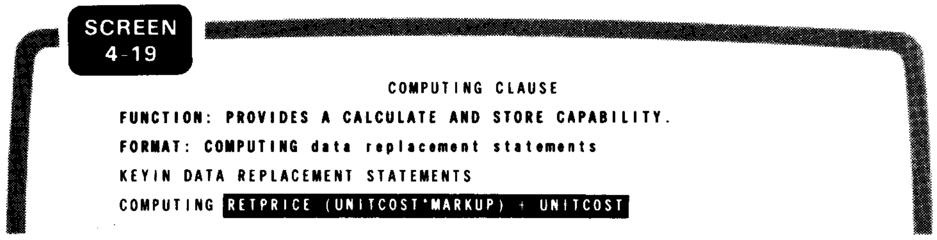
```
                              COMPUTING CLAUSE
 FUNCTION: PROVIDES A CALCULATE AND STORE CAPABILITY.
 FORMAT: COMPUTING data replacement statements
 KEYIN DATA REPLACEMENT STATEMENTS
 COMPUTING MARKUP 0.35
```

Clause selection

When you complete this clause, press the **XMIT** key. The CHANGE DATA clauses screen reappears (screen 4-17), and you select another clause for your program.

Select COMPUTING clause

Because you also want to compute a new retail price (RETPRICE), choose number 5, COMPUTING clause, again. The COMPUTING clause screen reappears (screen 4-19), and you key in another data replacement statement.


SCREEN 4-19

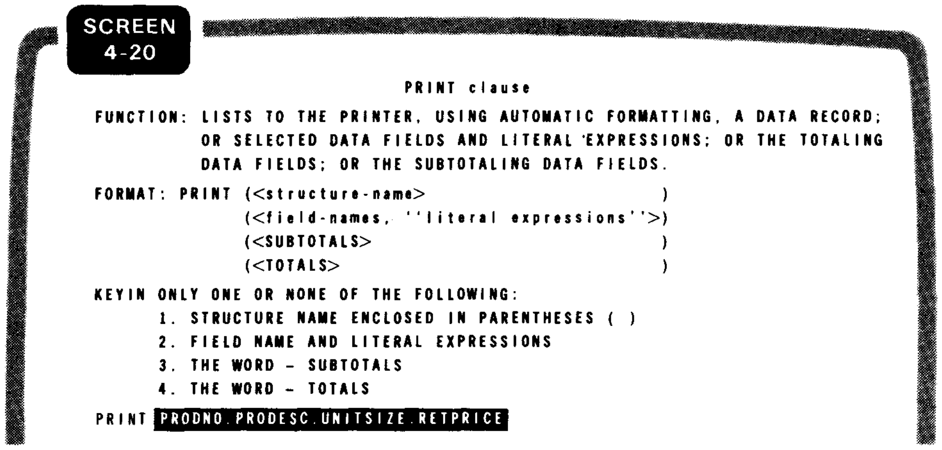
```
                              COMPUTING CLAUSE
 FUNCTION: PROVIDES A CALCULATE AND STORE CAPABILITY.
 FORMAT: COMPUTING data replacement statements
 KEYIN DATA REPLACEMENT STATEMENTS
 COMPUTING RETPRICE (UNITCOST*MARKUP) + UNITCOST
```

When you press the **XMIT** key, the CHANGE DATA clauses screen reappears (screen 4-17), and you select another clause.

PRINT clause

Because you want to print a report of the new prices, choose number 9, PRINT clause. When the PRINT clause screen appears (screen 4-20), key in the parameter you want.

SCREEN 4-20



```
                              PRINT clause
FUNCTION: LISTS TO THE PRINTER, USING AUTOMATIC FORMATTING, A DATA RECORD;
          OR SELECTED DATA FIELDS AND LITERAL EXPRESSIONS; OR THE TOTALING
          DATA FIELDS; OR THE SUBTOTALING DATA FIELDS.
FORMAT: PRINT (<structure-name>                         )
              (<field-names, ''literal expressions''>)
              (<SUBTOTALS>                              )
              (<TOTALS>                                 )
KEYIN ONLY ONE OR NONE OF THE FOLLOWING:
      1. STRUCTURE NAME ENCLOSED IN PARENTHESES ( )
      2. FIELD NAME AND LITERAL EXPRESSIONS
      3. THE WORD - SUBTOTALS
      4. THE WORD - TOTALS
PRINT PRODNO.PRODESC.UNITSIZE.RETPRICE
```

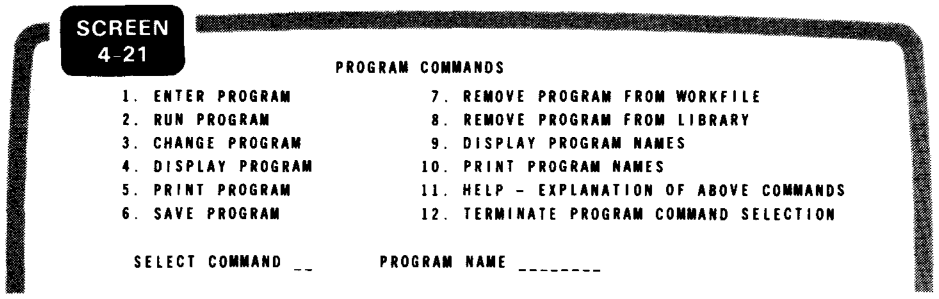
Statement completion

When you complete this clause screen, press the **XMIT** key to proceed. The CHANGE DATA clauses screen is redisplayed (screen 4-17). Because you're finished with the CHANGE DATA program keyin, pick number 17, CHANGE DATA STATEMENT COMPLETED.

Program compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4-21). Now you can choose what you want to do with your program.

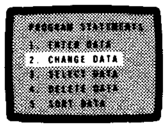
SCREEN 4-21



```
                    PROGRAM COMMANDS
   1. ENTER PROGRAM              7. REMOVE PROGRAM FROM WORKFILE
   2. RUN PROGRAM                8. REMOVE PROGRAM FROM LIBRARY
   3. CHANGE PROGRAM             9. DISPLAY PROGRAM NAMES
   4. DISPLAY PROGRAM           10. PRINT PROGRAM NAMES
   5. PRINT PROGRAM             11. HELP - EXPLANATION OF ABOVE COMMANDS
   6. SAVE PROGRAM              12. TERMINATE PROGRAM COMMAND SELECTION

    SELECT COMMAND __       PROGRAM NAME ________
```

SAVE PROGRAM option

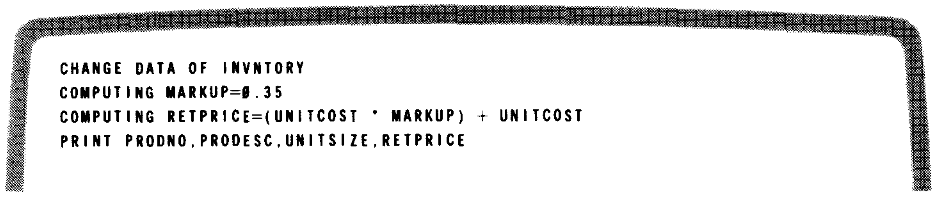
Again, as with all programs, you have the option to save your program. Because this program assigns a specific value to a field (MARKUP), it's unlikely that you'll want to use it again, so don't save it.



Program display

So you can get an idea of what your finished program looks like, select number 4, DISPLAY PROGRAM, and press the **XMIT** key. Then, ESCORT displays your program.

Example program



```
CHANGE DATA OF INVNTORY
COMPUTING MARKUP=Ø.35
COMPUTING RETPRICE=(UNITCOST * MARKUP) + UNITCOST
PRINT PRODNO,PRODESC,UNITSIZE,RETPRICE
```

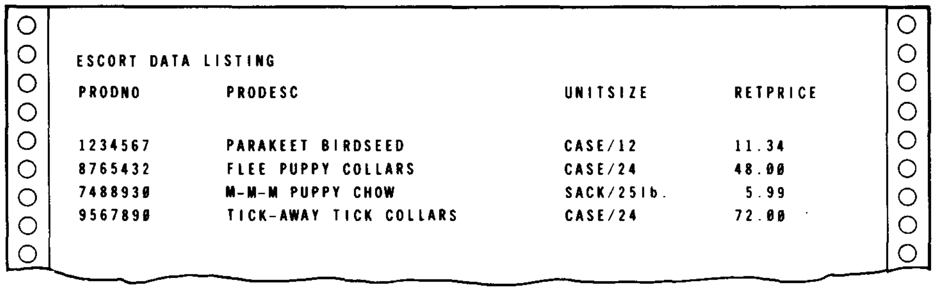
To get back to the PROGRAM COMMANDS menu (screen 4-21), press the **XMIT** key.

Program execution

Then, choose number 2, RUN PROGRAM, on screen 4-21 to execute your program.

Because you elected to output your data by using the standard system formatting PRINT clause, ESCORT outputs your data in the following manner:

Standard system output format

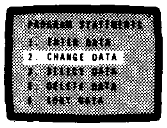


```
ESCORT DATA LISTING
PRODNO      PRODESC                      UNITSIZE      RETPRICE

1234567     PARAKEET BIRDSEED            CASE/12       11.34
8765432     FLEE PUPPY COLLARS           CASE/24       48.ØØ
748893Ø     M-M-M PUPPY CHOW             SACK/25lb.     5.99
956789Ø     TICK-AWAY TICK COLLARS       CASE/24       72.ØØ
```

4.10. CHANGE DATA OF MASTER FILE BY REPLACING WITH MATCHING FILE DATA

Creating a transactional file

Another variation of the CHANGE DATA statement is to use it to change specific fields in your record by replacing them with new data. To do this, you create a transactional file that contains the fields you want to change. The CHANGE DATA program statement lets you replace the data in the master file with the data in the transactional file.



Sample program

To create a sample transactional file, use the structure (INVNTORY) listed in 4.2. To refresh your memory, the structure of INVNTORY is:

INVNTORY structure

```
STRUCTURE OF INVNTORY

PRODNO       7N        K1
PRODESC      25A
UNITSIZE     10A
UNITCOST     4V2
MARKUP       1V2
RETPRICE     5V2
QONHAND      5N
QONORDER     5N
REORDERL     2N
QUANDISC     1V2
VENDNAME     20A
VENDADDR     35A
VENDZIP      5N
```

Using transactional files

Suppose you want to create a transactional file to change prices in your master inventory file. The first step is to create a structure containing the fields you want to change. For this example, use the structure INVTRAN. Then use an ENTER DATA program to create and input data into your transactional file. Because PRODNO is a keyed field, ESCORT uses this field as the CHANGE criteria.

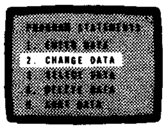
The structure of INVTRAN is:

INVTRAN structure

```
STRUCTURE OF INVTRAN

PRODNO       7N       K1
UNITCOST     4V2
```

Sample program

This example changes the UNITCOST field of selected records by matching the product number (PRODNO). When the PRODNO fields match, the UNITCOST field of INVNTORY is replaced by the UNITCOST field of INVTRAN. After each replacement is performed, a new retail price (RETPRICE) for INVNTORY is computed by multiplying the new UNITCOST by the MARKUP field and adding the value of UNITCOST.



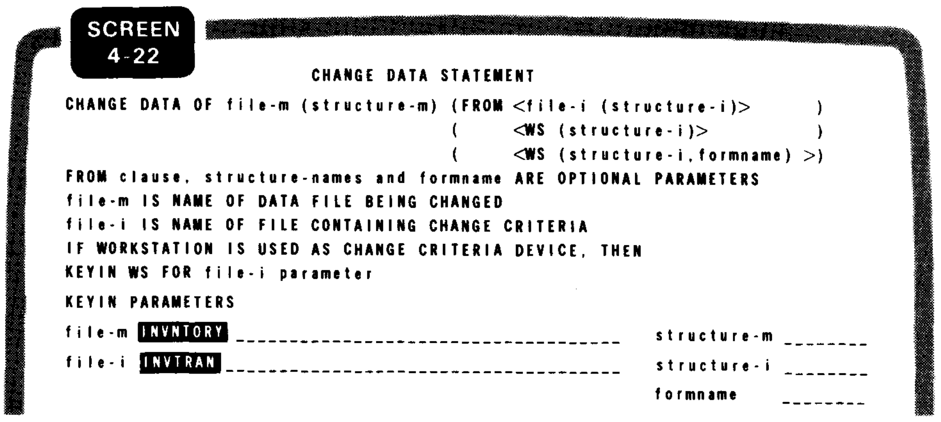
Name the program

On the PROGRAM COMMANDS menu (screen 4-21), choose number 1, ENTER PROGRAM, and name your program. For this example, name it CHDATA. Using the structures specified, you then initiate the CHANGE DATA statement. To do this, choose number 2 on the PROGRAM STATEMENTS menu (screen 4-2).

CHANGE DATA program statement

This displays the screen containing the basic format for the CHANGE DATA statement (screen 4-22). Fill in the required parameters. In this case, you need to use the FROM clause because the changed data is being input from another file.

SCREEN 4-22



```
                    CHANGE DATA STATEMENT
CHANGE DATA OF file-m (structure-m) (FROM <file-i (structure-i)>       )
                                    (     <WS (structure-i)>           )
                                    (     <WS (structure-i,formname) >)
FROM clause, structure-names and formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING CHANGED
file-i IS NAME OF FILE CONTAINING CHANGE CRITERIA
IF WORKSTATION IS USED AS CHANGE CRITERIA DEVICE, THEN
KEYIN WS FOR file-i parameter
KEYIN PARAMETERS
file-m INVNTORY ___________________________________   structure-m ________
file-i INVTRAN ____________________________________   structure-i ________
                                                      formname    ________
```

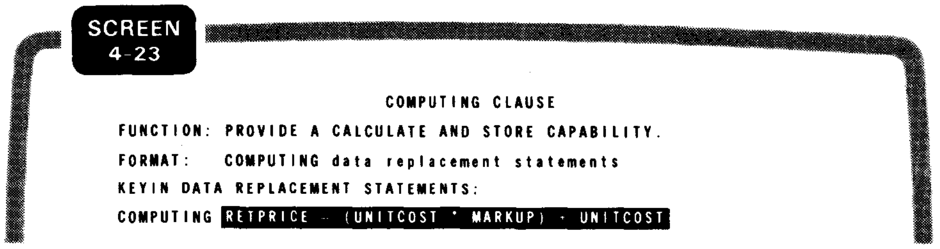
Clause decisions

Once you complete your CHANGE data screen, press the **XMIT** key to proceed. Then the clauses screen (screen 4-17) appears, listing all the clauses applicable to the CHANGE DATA statement.

Selecting the COMPUTING clause

Because you also want to compute a new retail price (RETPRICE) after the specified UNITCOST fields are changed, you must specify the COMPUTING clause, selection 5. The COMPUTING clause screen for this example appears (screen 4-23):

SCREEN 4-23



```
                          COMPUTING CLAUSE
FUNCTION: PROVIDE A CALCULATE AND STORE CAPABILITY.
FORMAT:   COMPUTING data replacement statements
KEYIN DATA REPLACEMENT STATEMENTS:
COMPUTING RETPRICE = (UNITCOST * MARKUP) + UNITCOST
```

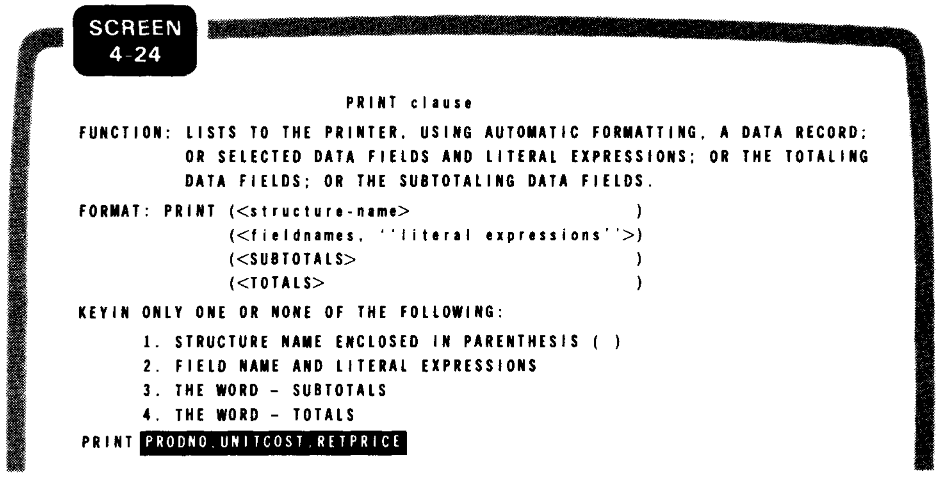
Additional clause selection

When you complete this clause, press the **XMIT** key. The CHANGE DATA clauses screen reappears (screen 4–17).

Choosing the PRINT option

Because you want to print the new data in a report form, choose the system formatting PRINT clause, number 9 on the menu. This displays the PRINT clause screen (screen 4–24), and you key in the parameters you want to use.

SCREEN 4-24



```
                          PRINT clause
FUNCTION: LISTS TO THE PRINTER, USING AUTOMATIC FORMATTING, A DATA RECORD;
          OR SELECTED DATA FIELDS AND LITERAL EXPRESSIONS; OR THE TOTALING
          DATA FIELDS; OR THE SUBTOTALING DATA FIELDS.
FORMAT: PRINT (<structure-name>                      )
              (<fieldnames, ''literal expressions''>)
              (<SUBTOTALS>                           )
              (<TOTALS>                              )
KEYIN ONLY ONE OR NONE OF THE FOLLOWING:
     1. STRUCTURE NAME ENCLOSED IN PARENTHESIS ( )
     2. FIELD NAME AND LITERAL EXPRESSIONS
     3. THE WORD - SUBTOTALS
     4. THE WORD - TOTALS
PRINT PRODNO,UNITCOST,RETPRICE
```

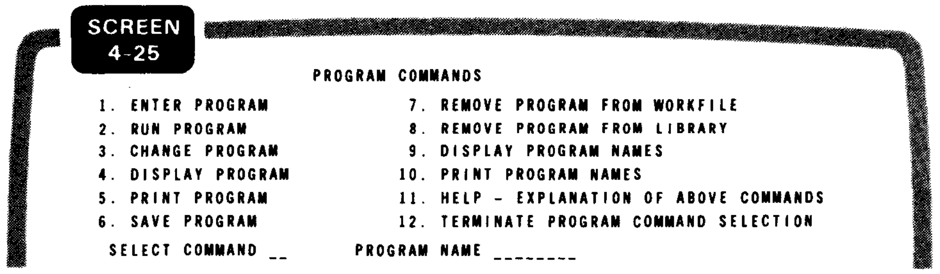
Statement completion

When you complete this clause, press the **XMIT** key to proceed. The CHANGE DATA clauses screen is redisplayed (screen 4–17). Because you're finished with the CHANGE DATA program keyin, pick number 17, CHANGE DATA STATEMENT COMPLETED.

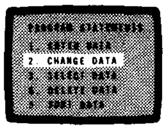
Program compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4–25). Now you can choose what you want to do with your program.

SCREEN 4-25



```
                    PROGRAM COMMANDS
 1. ENTER PROGRAM               7. REMOVE PROGRAM FROM WORKFILE
 2. RUN PROGRAM                 8. REMOVE PROGRAM FROM LIBRARY
 3. CHANGE PROGRAM              9. DISPLAY PROGRAM NAMES
 4. DISPLAY PROGRAM            10. PRINT PROGRAM NAMES
 5. PRINT PROGRAM              11. HELP - EXPLANATION OF ABOVE COMMANDS
 6. SAVE PROGRAM               12. TERMINATE PROGRAM COMMAND SELECTION
  SELECT COMMAND __        PROGRAM NAME ________
```



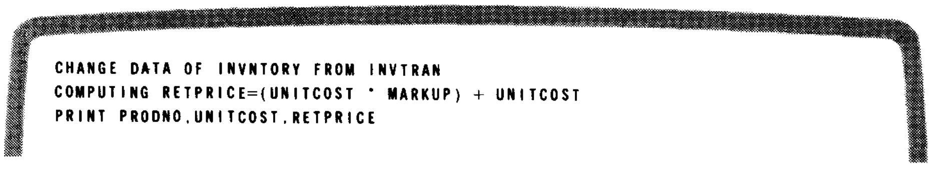
Saving the program

Again, as with all programs, you have the option to save your program. Because this program can be used over and over again when you change your prices, it would be wise to save it. We'll give information on saving programs in 5.13.

Program display

To get an idea of what your finished program looks like, choose number 4, DISPLAY PROGRAM, on screen 4-25; key in your program name (CHDATA), and press the **XMIT** key. Then ESCORT displays your program:

Program example

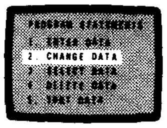


```
CHANGE DATA OF INVNTORY FROM INVTRAN
COMPUTING RETPRICE=(UNITCOST * MARKUP) + UNITCOST
PRINT PRODNO,UNITCOST,RETPRICE
```

To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

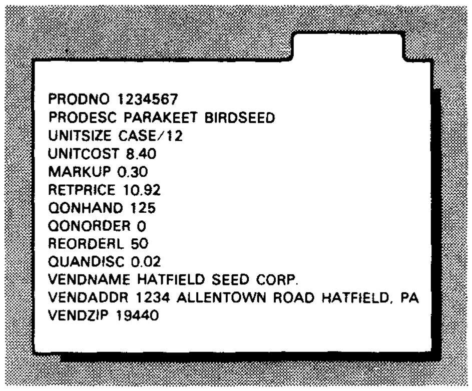
Program execution

Then choose number 2, RUN PROGRAM, on screen 4-25, to execute your program.



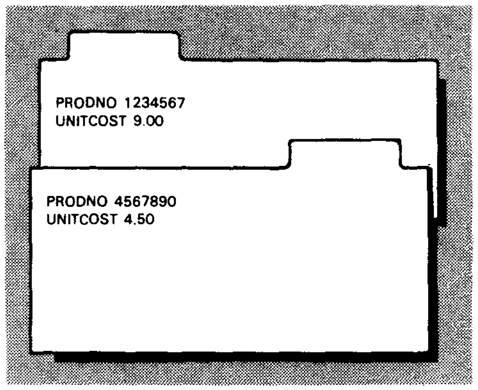
The following records from the INVNTORY and INVTRAN file show how the data change program affects the file data.

Three data records from INVNTORY:



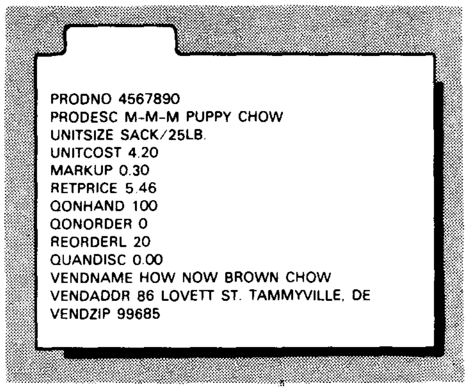
```
PRODNO 1234567
PRODESC PARAKEET BIRDSEED
UNITSIZE CASE/12
UNITCOST 8.40
MARKUP 0.30
RETPRICE 10.92
QONHAND 125
QONORDER 0
REORDERL 50
QUANDISC 0.02
VENDNAME HATFIELD SEED CORP.
VENDADDR 1234 ALLENTOWN ROAD HATFIELD, PA
VENDZIP 19440
```

Two data records from INVTRAN:

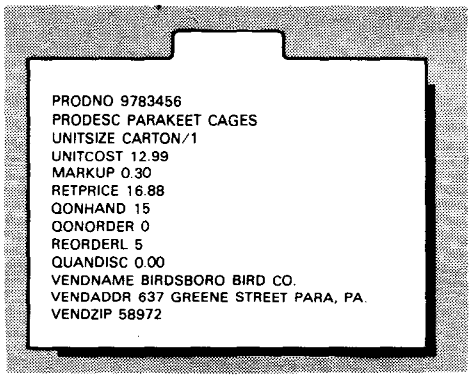


```
PRODNO 1234567
UNITCOST 9.00
```

```
PRODNO 4567890
UNITCOST 4.50
```



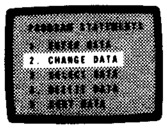
```
PRODNO 4567890
PRODESC M-M-M PUPPY CHOW
UNITSIZE SACK/25LB.
UNITCOST 4.20
MARKUP 0.30
RETPRICE 5.46
QONHAND 100
QONORDER 0
REORDERL 20
QUANDISC 0.00
VENDNAME HOW NOW BROWN CHOW
VENDADDR 86 LOVETT ST. TAMMYVILLE, DE
VENDZIP 99685
```



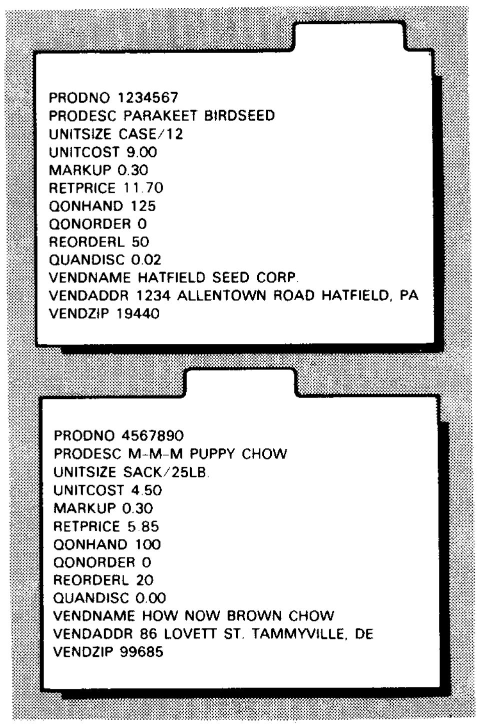
```
PRODNO 9783456
PRODESC PARAKEET CAGES
UNITSIZE CARTON/1
UNITCOST 12.99
MARKUP 0.30
RETPRICE 16.88
QONHAND 15
QONORDER 0
REORDERL 5
QUANDISC 0.00
VENDNAME BIRDSBORO BIRD CO.
VENDADDR 637 GREENE STREET PARA, PA.
VENDZIP 58972
```

Program results

Notice that only the records with the PRODNO (product number) 1234567 and 4567890 are replaced with the data from INVTRAN. The third record of INVNTORY is not changed because there is no INVTRAN record requesting that it be changed.



The changed data records now look like this:



PRODNO 1234567
PRODESC PARAKEET BIRDSEED
UNITSIZE CASE/12
UNITCOST 9.00
MARKUP 0.30
RETPRICE 11.70 ←
QONHAND 125
QONORDER 0
REORDERL 50
QUANDISC 0.02
VENDNAME HATFIELD SEED CORP.
VENDADDR 1234 ALLENTOWN ROAD HATFIELD, PA
VENDZIP 19440

PRODNO 4567890
PRODESC M-M-M PUPPY CHOW
UNITSIZE SACK/25LB.
UNITCOST 4.50
MARKUP 0.30
RETPRICE 5.85 ←
QONHAND 100
QONORDER 0
REORDERL 20
QUANDISC 0.00
VENDNAME HOW NOW BROWN CHOW
VENDADDR 86 LOVETT ST. TAMMYVILLE, DE
VENDZIP 99685

4.11. USING THE SCREEN MENU METHOD TO CREATE THREE KINDS OF SELECT DATA PROGRAMS

Purpose

Using the same data file and structure (INVNTORY) you created in 4.5 and modified in 4.6, you can use the SELECT DATA program statement to retrieve your data, using one of three methods:

Selection methods

- Select data, with or without conditions
- Select data from the workstation
- Select data of one file from another

4.12. SELECT DATA WITH CONDITIONS

Example definition

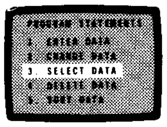
The example in this subsection shows how you use the SELECT DATA program statement to issue an inventory reorder. In this example, you retrieve data records selectively from the file INVNTORY based on the IF condition that the field QONHAND (quantity on hand) is less than or equal to the field REORDERL (reorder level). You then output the reorder records to the printer, using the structure INVREORD and the form REORDFM, both shown below.

When you do not specify conditions through either the IF or WHILE clauses, all file records are output.

Example structure

As you recall from 4.2, the structure of INVNTORY is:

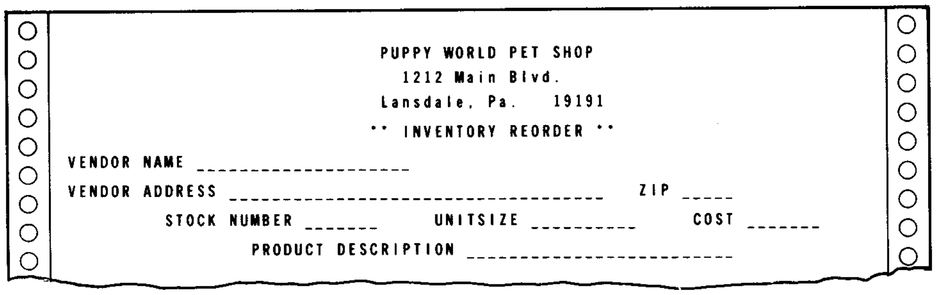
```
STRUCTURE OF INVNTORY

PRODNO        7N       K1
PRODESC       25A
UNITSIZE      10A
UNITCOST      4V2
MARKUP        1V2
RETPRICE      5V2
QONHAND       5N
QONORDER      5N
REORDERL      2N
QUANDISC      1V2
VENDNAME      20A
VENDADDR      35A
VENDZIP       5N
```



Using an output screen form

Because you're using a form for your output and you can't create forms in ESCORT, you must create the form either before you enter ESCORT or exit from your current session and create it. For this example, assume you created the form (REORDFM) before entering ESCORT and it looks like this:

Example screen form



```
                              PUPPY WORLD PET SHOP
                                 1212 Main Blvd.
                               Lansdale, Pa.   19191
                             ** INVENTORY REORDER **
VENDOR NAME ____________________
VENDOR ADDRESS ___________________________________   ZIP _____
           STOCK NUMBER _______      UNITSIZE __________    COST _______
                   PRODUCT DESCRIPTION _________________________
```

Structures and forms

Because you're using a form to output your file data, the fields you want to output must appear in the structure in the order you want them output in the form. In our INVNTORY structure, the fields do not appear in the order you want them output in the form, so you must c-eate a new structure.

Example structure for form

So you can visualize how the structure fields parallel the form fields, the following structure, named INVREORD, is shown with the form fields it parallels:

STRUCTURE OF INVREORD			FORM FIELD DESIGNATION
VENDNAME	20A		*Vendor name*
VENDADDR	35A		*Vendor address*
VENDZIP	5N		*Vendor zip code*
PRODNO	7N	K1	*Stock number*
UNITSIZE	10A		*Unit size*
UNITCOST	4V2		*Cost*
PRODESC	25A		*Product description*

Now that your structure and form specifications are complete, you're ready to build your program.

Building the program

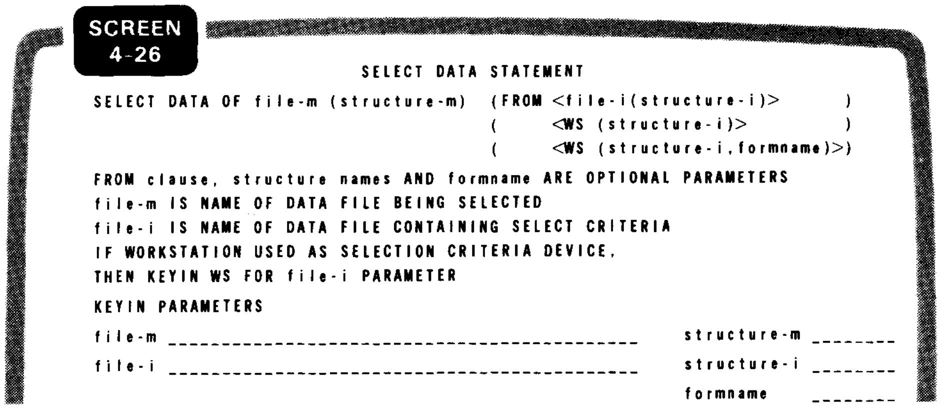
On the PROGRAM COMMANDS menu (screen 4-25), choose number 1, ENTER PROGRAM, and name your program. For this example, name it SELDATA and press the **XMIT** key. Then the PROGRAM STATEMENTS menu appears.

Initiation of SELECT DATA

To get the SELECT DATA function, choose number 3 on the PROGRAM STATEMENTS menu (screen 4-2).

This displays the screen containing the format of the basic SELECT DATA statement (screen 4-26). (We'll give you more information on the basic SELECT DATA statement in 7.11.)

SCREEN 4-26



```
                         SELECT DATA STATEMENT
SELECT DATA OF file-m (structure-m)  (FROM <file-i(structure-i)>        )
                                     (      <WS (structure-i)>           )
                                     (      <WS (structure-i,formname)>)
FROM clause, structure names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING SELECTED
file-i IS NAME OF DATA FILE CONTAINING SELECT CRITERIA
IF WORKSTATION USED AS SELECTION CRITERIA DEVICE,
THEN KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS
file-m ____________________________________________   structure-m ________
file-i ____________________________________________   structure-i ________
                                                      formname    ________
```

Required parameters

Notice that the only required parameter is the *file-m* specification. This names the file you want to retrieve (INVNTORY, for this example).

Structure-name parameter

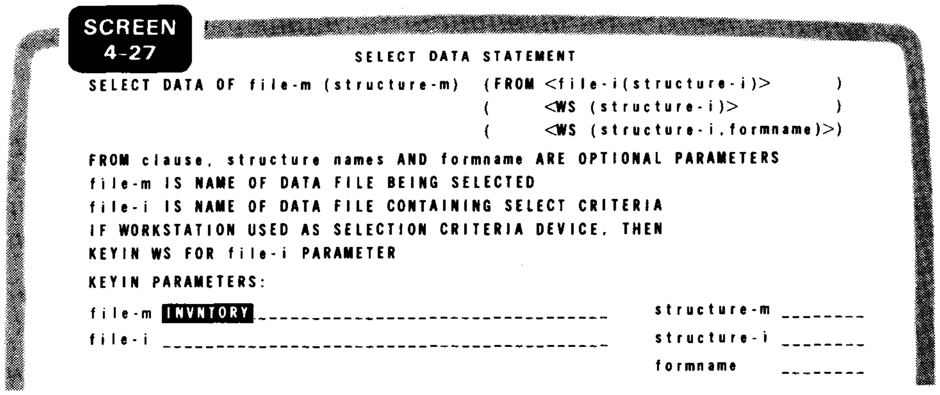
Structure-name,formname parameter

Again, the *structure-name* parameter is optional, and you need only use it when the structure name is different from the first eight characters of the file name or when you use the *structure-name,formname* parameter to designate a customized output form.

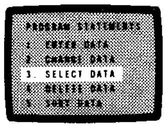
Parameter keyin

For this example, fill in the *file-m* parameter with the file name INVNTORY. You don't need a structure name because it's the same as the name of the file you're selecting. Your SELECT DATA screen now looks like screen 4-27.

SCREEN 4-27

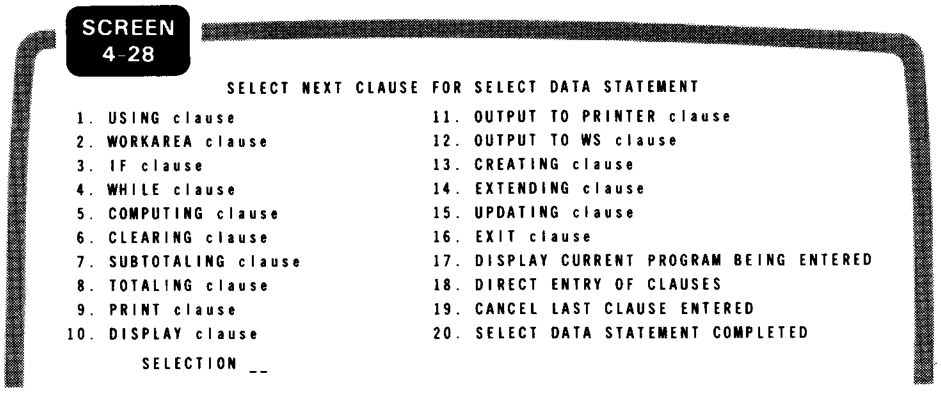


```
                         SELECT DATA STATEMENT
SELECT DATA OF file-m (structure-m)  (FROM <file-i(structure-i)>        )
                                     (      <WS (structure-i)>           )
                                     (      <WS (structure-i,formname)>)
FROM clause, structure names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING SELECTED
file-i IS NAME OF DATA FILE CONTAINING SELECT CRITERIA
IF WORKSTATION USED AS SELECTION CRITERIA DEVICE, THEN
KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS:
file-m INVNTORY__________________________________   structure-m ________
file-i ___________________________________________   structure-i ________
                                                     formname    ________
```



Clause decisions

Once your basic SELECT DATA screen is completed, press the **XMIT** key to proceed. The next screen (screen 4–28) lists all the clauses applicable to the SELECT DATA statement.


SCREEN 4-28

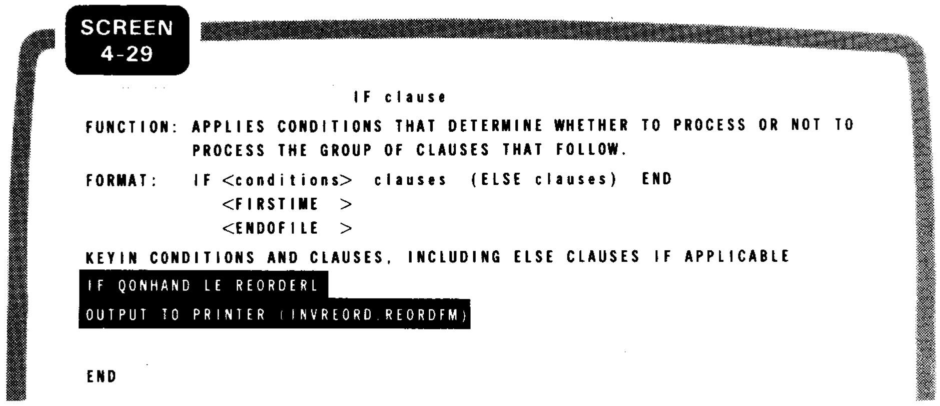
```
                    SELECT NEXT CLAUSE FOR SELECT DATA STATEMENT
 1. USING clause                  11. OUTPUT TO PRINTER clause
 2. WORKAREA clause               12. OUTPUT TO WS clause
 3. IF clause                     13. CREATING clause
 4. WHILE clause                  14. EXTENDING clause
 5. COMPUTING clause              15. UPDATING clause
 6. CLEARING clause               16. EXIT clause
 7. SUBTOTALING clause            17. DISPLAY CURRENT PROGRAM BEING ENTERED
 8. TOTALING clause               18. DIRECT ENTRY OF CLAUSES
 9. PRINT clause                  19. CANCEL LAST CLAUSE ENTERED
10. DISPLAY clause                20. SELECT DATA STATEMENT COMPLETED
        SELECTION __
```

IF clause

Because you want to set conditions on the data selection, choose number 3, IF clause.

The screen containing the format for the IF clause appears (screen 4–29); key in your conditional clause and any clauses to be executed if the conditions you specify are satisfied:

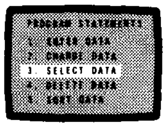
IF clause screen


SCREEN 4-29

```
                              IF clause
FUNCTION: APPLIES CONDITIONS THAT DETERMINE WHETHER TO PROCESS OR NOT TO
          PROCESS THE GROUP OF CLAUSES THAT FOLLOW.
FORMAT:   IF <conditions> clauses (ELSE clauses) END
             <FIRSTIME  >
             <ENDOFILE  >
KEYIN CONDITIONS AND CLAUSES, INCLUDING ELSE CLAUSES IF APPLICABLE
IF QONHAND LE REORDERL
OUTPUT TO PRINTER (INVREORD,REORDFM)


END
```

Using customized output

Because you want customized data output to the printer, key in the OUTPUT TO PRINTER clause with the necessary structure and form names.



PRINT clause

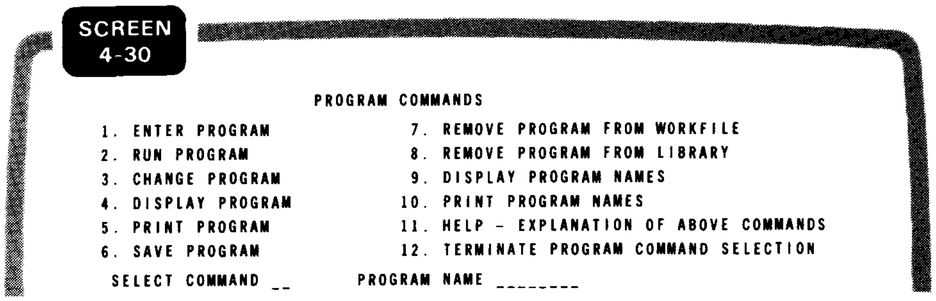
If you don't want customized output and still want a printer listing, use the PRINT clause that uses standard system formatting.

Statement completion

When you complete this screen, press the **XMIT** key to proceed. The SELECT DATA clauses screen is then redisplayed (screen 4-28). Because you're finished with the SELECT DATA program keyin, pick number 20, SELECT DATA STATEMENT COMPLETED.

Program compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4-30). Now you can choose what you want to do with your program.


SCREEN 4-30

```
                           PROGRAM COMMANDS
  1. ENTER PROGRAM                 7. REMOVE PROGRAM FROM WORKFILE
  2. RUN PROGRAM                   8. REMOVE PROGRAM FROM LIBRARY
  3. CHANGE PROGRAM                9. DISPLAY PROGRAM NAMES
  4. DISPLAY PROGRAM              10. PRINT PROGRAM NAMES
  5. PRINT PROGRAM                11. HELP - EXPLANATION OF ABOVE COMMANDS
  6. SAVE PROGRAM                 12. TERMINATE PROGRAM COMMAND SELECTION
   SELECT COMMAND __         PROGRAM NAME ________
```

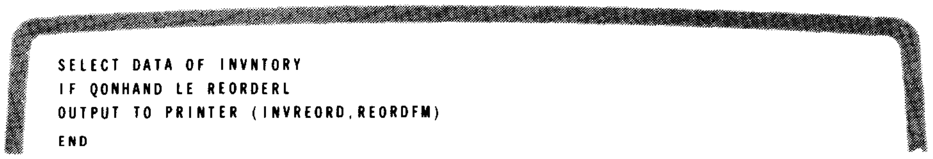
Saving the program

Again, as with all programs, you have the option to save your program. Because this program can be used over and over again for inventory reorders, it would be wise to save it. (We'll give information on saving programs in 5.13.)

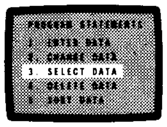
Program display

To get an idea of what your finished program looks like, choose number 4, DISPLAY PROGRAM, on screen 4-30 and key in the name of the program you want displayed (SELDATA). Then ESCORT displays your sample program:

Example program



```
SELECT DATA OF INVNTORY
IF QONHAND LE REORDERL
OUTPUT TO PRINTER (INVREORD,REORDFM)
END
```

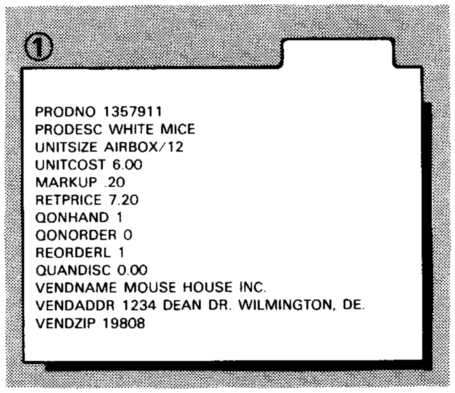


To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

Program execution

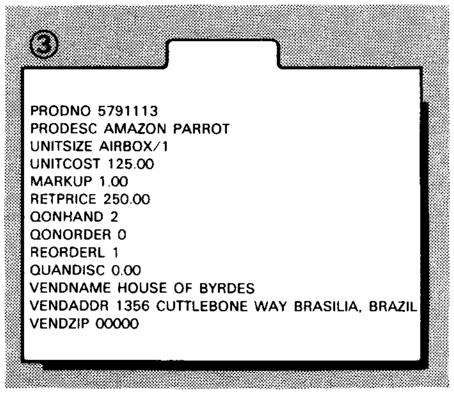
Then choose number 2, RUN PROGRAM, to execute your program.

Of the following four records from the previously created file INVNTORY, only 1, 2, and 4 meet the selection conditions.



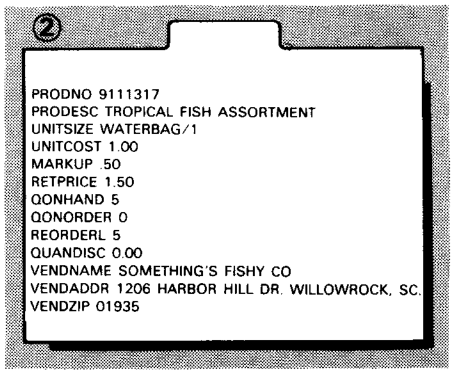
①

```
PRODNO 1357911
PRODESC WHITE MICE
UNITSIZE AIRBOX/12
UNITCOST 6.00
MARKUP .20
RETPRICE 7.20
QONHAND 1
QONORDER 0
REORDERL 1
QUANDISC 0.00
VENDNAME MOUSE HOUSE INC.
VENDADDR 1234 DEAN DR. WILMINGTON, DE.
VENDZIP 19808
```



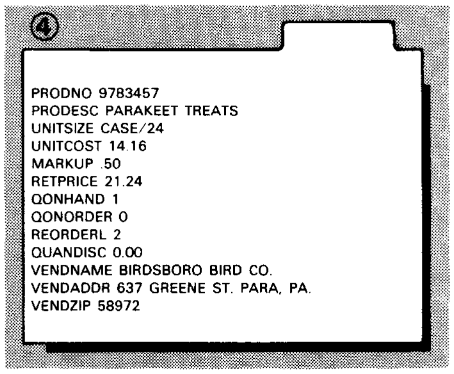
③

```
PRODNO 5791113
PRODESC AMAZON PARROT
UNITSIZE AIRBOX/1
UNITCOST 125.00
MARKUP 1.00
RETPRICE 250.00
QONHAND 2
QONORDER 0
REORDERL 1
QUANDISC 0.00
VENDNAME HOUSE OF BYRDES
VENDADDR 1356 CUTTLEBONE WAY BRASILIA, BRAZIL
VENDZIP 00000
```



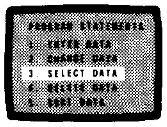
②

```
PRODNO 9111317
PRODESC TROPICAL FISH ASSORTMENT
UNITSIZE WATERBAG/1
UNITCOST 1.00
MARKUP .50
RETPRICE 1.50
QONHAND 5
QONORDER 0
REORDERL 5
QUANDISC 0.00
VENDNAME SOMETHING'S FISHY CO
VENDADDR 1206 HARBOR HILL DR. WILLOWROCK, SC.
VENDZIP 01935
```



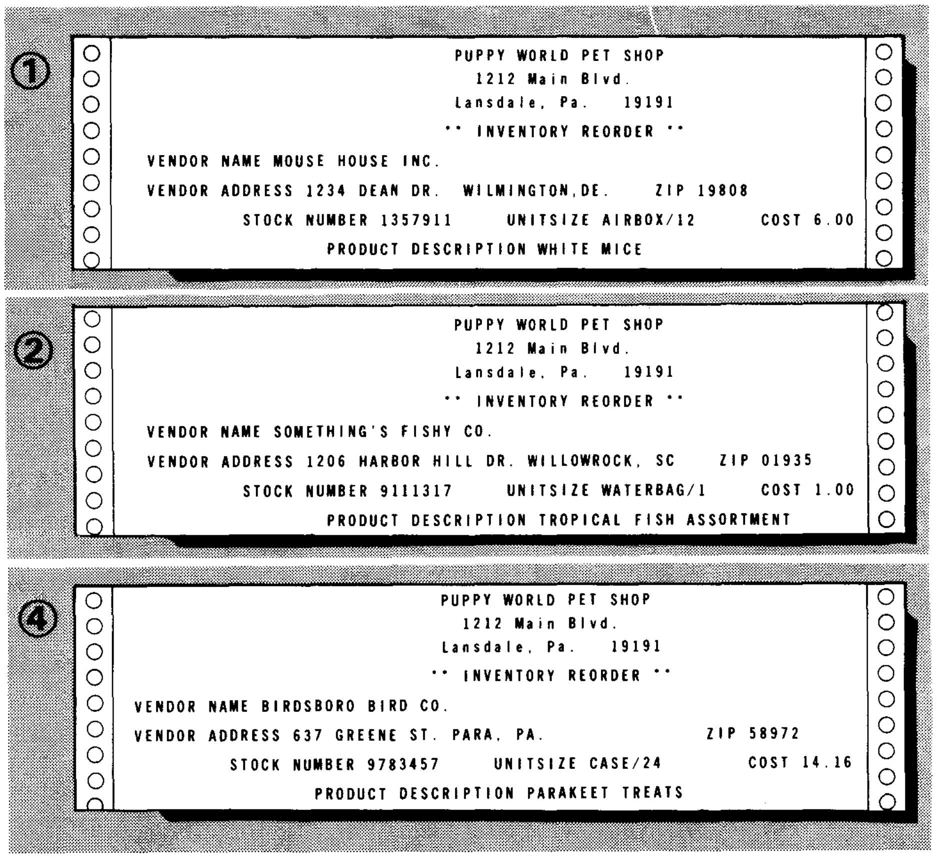
④

```
PRODNO 9783457
PRODESC PARAKEET TREATS
UNITSIZE CASE/24
UNITCOST 14.16
MARKUP .50
RETPRICE 21.24
QONHAND 1
QONORDER 0
REORDERL 2
QUANDISC 0.00
VENDNAME BIRDSBORO BIRD CO.
VENDADDR 637 GREENE ST. PARA, PA.
VENDZIP 58972
```



Program output

The three forms show the data records from INVNTORY that meet the selection criteria and are output for inventory reorder.



①

```
                         PUPPY WORLD PET SHOP
                           1212 Main Blvd.
                         Lansdale, Pa.   19191
                        ** INVENTORY REORDER **
VENDOR NAME MOUSE HOUSE INC.
VENDOR ADDRESS 1234 DEAN DR.  WILMINGTON,DE.    ZIP 19808
         STOCK NUMBER 1357911     UNITSIZE AIRBOX/12      COST 6.00
               PRODUCT DESCRIPTION WHITE MICE
```

②

```
                         PUPPY WORLD PET SHOP
                           1212 Main Blvd.
                         Lansdale, Pa.   19191
                        ** INVENTORY REORDER **
VENDOR NAME SOMETHING'S FISHY CO.
VENDOR ADDRESS 1206 HARBOR HILL DR. WILLOWROCK, SC    ZIP 01935
         STOCK NUMBER 9111317     UNITSIZE WATERBAG/1     COST 1.00
               PRODUCT DESCRIPTION TROPICAL FISH ASSORTMENT
```

④

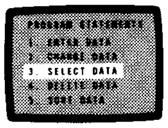
```
                         PUPPY WORLD PET SHOP
                           1212 Main Blvd.
                         Lansdale, Pa.   19191
                        ** INVENTORY REORDER **
VENDOR NAME BIRDSBORO BIRD CO.
VENDOR ADDRESS 637 GREENE ST. PARA, PA.             ZIP 58972
         STOCK NUMBER 9783457     UNITSIZE CASE/24        COST 14.16
               PRODUCT DESCRIPTION PARAKEET TREATS
```

4.13. SELECT DATA FROM THE WORKSTATION

Program objective

Again using the file and structure created in 4.5 (INVNTORY), this SELECT DATA FROM WS example performs a file inquiry operation. Because the field PRODNO (product number) in the structure INVNTORY is the key field, this field is used as your prompt field.

The program you write for this example displays the key field prompt from your file. You key in the product number (PRODNO) of the record you want selected, and ESCORT gets that record and displays it on the workstation.



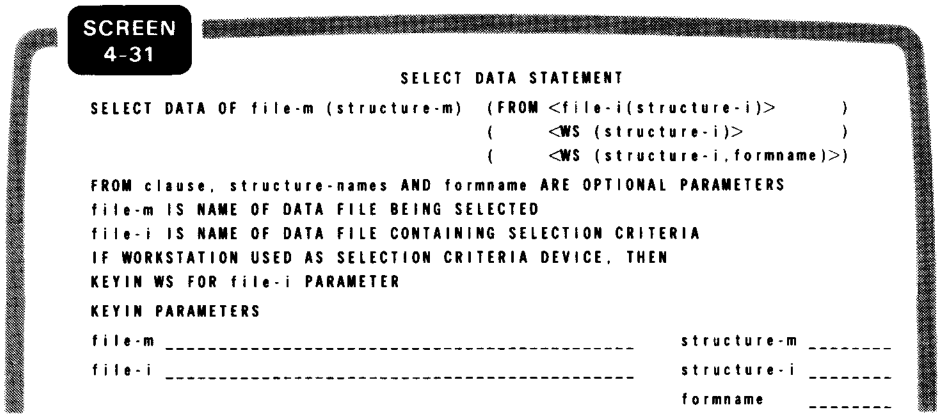
We'll assume you've chosen the ENTER PROGRAM command from the PROGRAM COMMANDS menu (screen 4-1) and are now at the PROGRAM STATEMENTS menu (screen 4-2).

Initiation of SELECT DATA

To get the SELECT DATA function, choose number 3 on the PROGRAM STATEMENTS menu (screen 4-2).

This displays the screen containing the format of the basic SELECT DATA statement (screen 4-31). (We'll give more information on the basic SELECT DATA statement in 7.11.)

SELECT DATA statement


SCREEN 4-31

```
                         SELECT DATA STATEMENT
SELECT DATA OF file-m (structure-m)  (FROM <file-i(structure-i)>        )
                                     (      <WS (structure-i)>          )
                                     (      <WS (structure-i,formname)>)
FROM clause, structure-names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING SELECTED
file-i IS NAME OF DATA FILE CONTAINING SELECTION CRITERIA
IF WORKSTATION USED AS SELECTION CRITERIA DEVICE, THEN
KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS
file-m ____________________________________________   structure-m ________
file-i ____________________________________________   structure-i ________
                                                      formname    ________
```

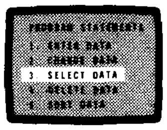
Required parameters

Notice that the only required parameter is the *file-m* specification. This names the file you want to retrieve (INVNTORY, for this example).

Structure-name parameter

Structure-name,formname parameter

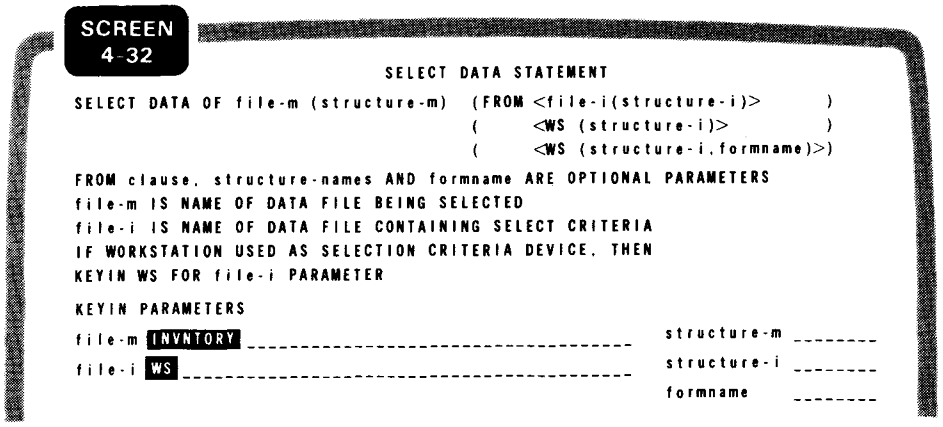
Again, the *structure-name* parameter is optional. You use it only when the structure name is different from the first eight characters of the file name or when you use the *structure-name,formname* parameter to designate a customized output form.



Parameter keyin

For this example, fill in the *file-m* parameter with the file name INVNTORY. You don't need a structure name because it's the same as the name of the file you're selecting. Then key in WS for the *file-i* parameter. Your SELECT DATA screen now looks like screen 4-32.

SCREEN 4-32

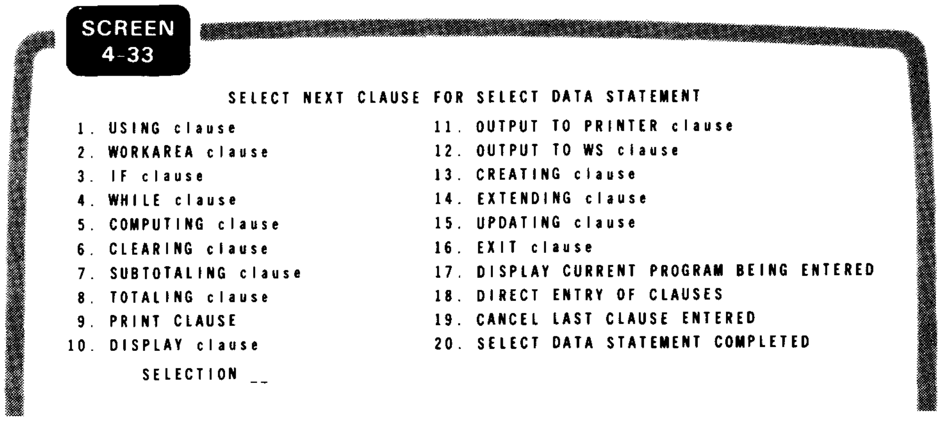


SELECT DATA statement

```
                         SELECT DATA STATEMENT
SELECT DATA OF file-m (structure-m)  (FROM <file-i(structure-i)>       )
                                     (      <WS (structure-i)>         )
                                     (      <WS (structure-i,formname)>)
FROM clause, structure-names AND formname ARE OPTIONAL PARAMETERS
file-m IS NAME OF DATA FILE BEING SELECTED
file-i IS NAME OF DATA FILE CONTAINING SELECT CRITERIA
IF WORKSTATION USED AS SELECTION CRITERIA DEVICE, THEN
KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS
file-m INVNTORY__________________________________   structure-m ________
file-i WS________________________________________   structure-i ________
                                                    formname    ________
```

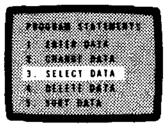
Now that your basic SELECT DATA screen is completed, press the **XMIT** key to proceed. The next screen (screen 4-33) lists all the clauses applicable to the SELECT DATA statement.

SCREEN 4-33



Clause selection

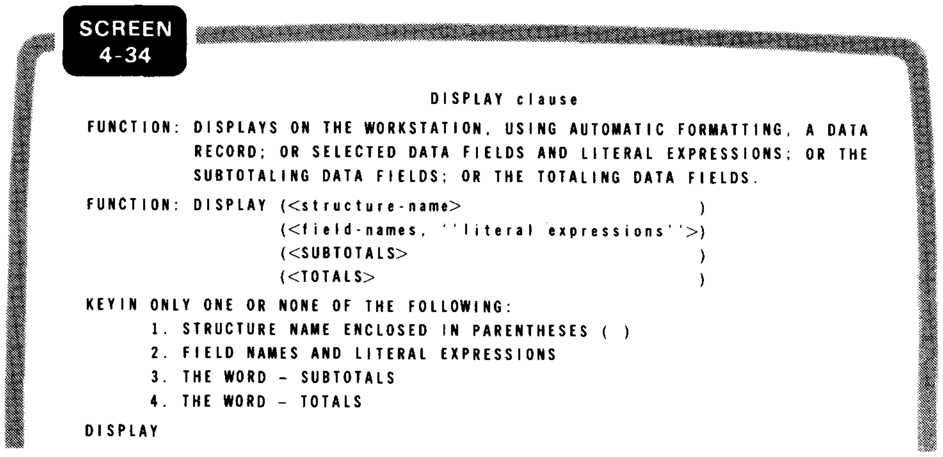
```
              SELECT NEXT CLAUSE FOR SELECT DATA STATEMENT
 1. USING clause                11. OUTPUT TO PRINTER clause
 2. WORKAREA clause             12. OUTPUT TO WS clause
 3. IF clause                   13. CREATING clause
 4. WHILE clause                14. EXTENDING clause
 5. COMPUTING clause            15. UPDATING clause
 6. CLEARING clause             16. EXIT clause
 7. SUBTOTALING clause          17. DISPLAY CURRENT PROGRAM BEING ENTERED
 8. TOTALING clause             18. DIRECT ENTRY OF CLAUSES
 9. PRINT CLAUSE                19. CANCEL LAST CLAUSE ENTERED
10. DISPLAY clause              20. SELECT DATA STATEMENT COMPLETED
       SELECTION __
```



Displaying data records

Because you want to display the data records you select on the workstation, choose number 10, DISPLAY clause. The screen containing the format for the DISPLAY clause appears (screen 4-34).

SCREEN 4-34



DISPLAY clause

```
                              DISPLAY clause
FUNCTION: DISPLAYS ON THE WORKSTATION, USING AUTOMATIC FORMATTING, A DATA
          RECORD; OR SELECTED DATA FIELDS AND LITERAL EXPRESSIONS; OR THE
          SUBTOTALING DATA FIELDS; OR THE TOTALING DATA FIELDS.
FUNCTION: DISPLAY (<structure-name>                          )
                  (<field-names, ''literal expressions''>)
                  (<SUBTOTALS>                               )
                  (<TOTALS>                                  )
KEYIN ONLY ONE OR NONE OF THE FOLLOWING:
      1. STRUCTURE NAME ENCLOSED IN PARENTHESES ( )
      2. FIELD NAMES AND LITERAL EXPRESSIONS
      3. THE WORD - SUBTOTALS
      4. THE WORD - TOTALS
DISPLAY
```

Because you want to display the whole record, you don't need to use any of the parameters. Press the **XMIT** key to continue.

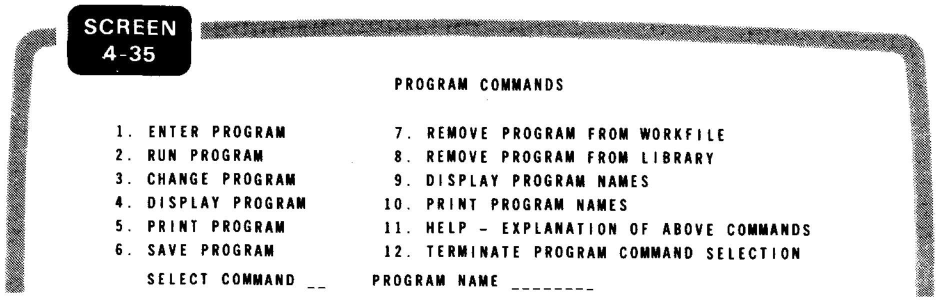
Statement completion

The SELECT DATA clauses screen reappears (screen 4-33). Because you're finished keying in your program, pick number 20, SELECT DATA STATEMENT COMPLETED.

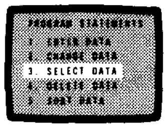
Program compilation

When ESCORT receives this statement, it compiles your program and displays the PROGRAM COMMANDS menu (screen 4-35). Now you can choose what you want to do with your program.

SCREEN 4-35



```
                           PROGRAM COMMANDS

  1. ENTER PROGRAM            7. REMOVE PROGRAM FROM WORKFILE
  2. RUN PROGRAM              8. REMOVE PROGRAM FROM LIBRARY
  3. CHANGE PROGRAM           9. DISPLAY PROGRAM NAMES
  4. DISPLAY PROGRAM         10. PRINT PROGRAM NAMES
  5. PRINT PROGRAM           11. HELP - EXPLANATION OF ABOVE COMMANDS
  6. SAVE PROGRAM            12. TERMINATE PROGRAM COMMAND SELECTION
     SELECT COMMAND __      PROGRAM NAME ________
```



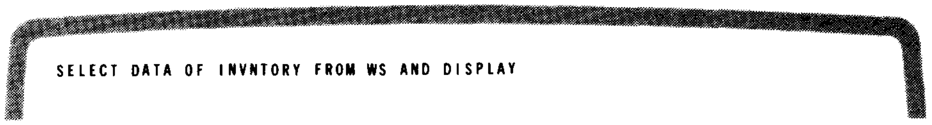
Saving the program

Again, as with all programs, you have the option to save your program. Because you probably won't use the program again, you won't need to save it.

Displaying the program

To get an idea of what your finished program looks like, choose number 4, DISPLAY PROGRAM, on screen 4-35. Because you're using the system-supplied name, RE$IDENT, you don't need to key in a program name, just press the **XMIT** key. Then ESCORT displays your sample program:

Example program



```
SELECT DATA OF INVNTORY FROM WS AND DISPLAY
```

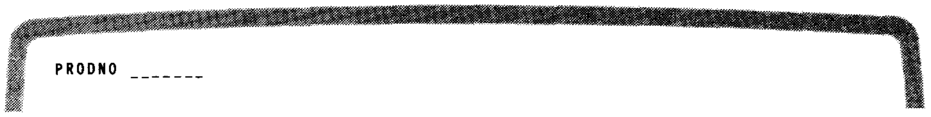
To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

Program execution

Then choose number 2, RUN PROGRAM, on screen 4-35 to execute your program.

When your program is executed, the key field prompt (PRODNO) appears on your screen:

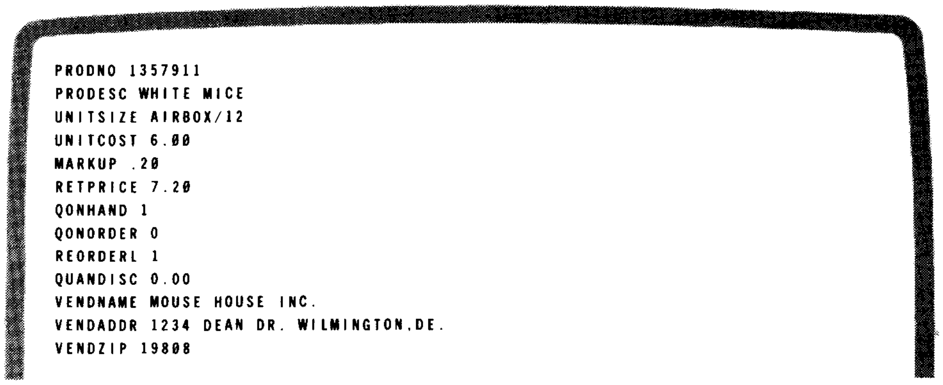
Keyed file selection prompt



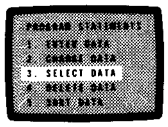
```
PRODNO _______
```

Key in the product number of the record you want to select (e.g., 1357911), and ESCORT displays the rest of the record. The record for this product number appears:

Keyed file record selected



```
PRODNO 1357911
PRODESC WHITE MICE
UNITSIZE AIRBOX/12
UNITCOST 6.00
MARKUP .20
RETPRICE 7.20
QONHAND 1
QONORDER 0
REORDERL 1
QUANDISC 0.00
VENDNAME MOUSE HOUSE INC.
VENDADDR 1234 DEAN DR. WILMINGTON.DE.
VENDZIP 19808
```



Each time you press the **XMIT** key, the prompt reappears, and you key in the product number for the next record you want to see.

To terminate input, press the **FUNCTION** and **F4** keys.

4.14. SELECT DATA FROM ANOTHER FILE

Program objective

This sample data retrieval program selects data from a file, using two other files, and creates a fourth file.

Files used

The program uses four files, three input and one output. It produces an itemized sales listing and the total daily sales.

The example uses these four structures:

Example structures

INPUT

VENDMSTR		
VNUMBER	5N	K1
VNAME	25A	
VADRESS	30A	
VCITYST	30A	
TELEPHON	9N	
BALANCE	7V2	
REMARKS	50A	

TRANFILE	
VNUMBER	5N
ITEMCODE	6A
NUMITEMS	4N

PARTFILE		
ITEMCODE	6A	K1
PARTDESC	30A	
PRICE	4V2	
COST	4V2	

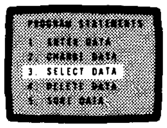
OUTPUT

DLYSALES	
VNUMBER	5N
ITEMCODE	6A
PARTDESC	30A
NUMITEMS	4N
PRICE	4V2
TOTALPRC	6V2
DATE	8A

Because this example is longer and involves more clauses than previous examples, we won't show the individual screens. Instead, we'll list the program and explain it line by line and then show the sample output.

Example setup

ESCORT programs do not have line numbers; however, the sample program is numbered for ease of explanation.

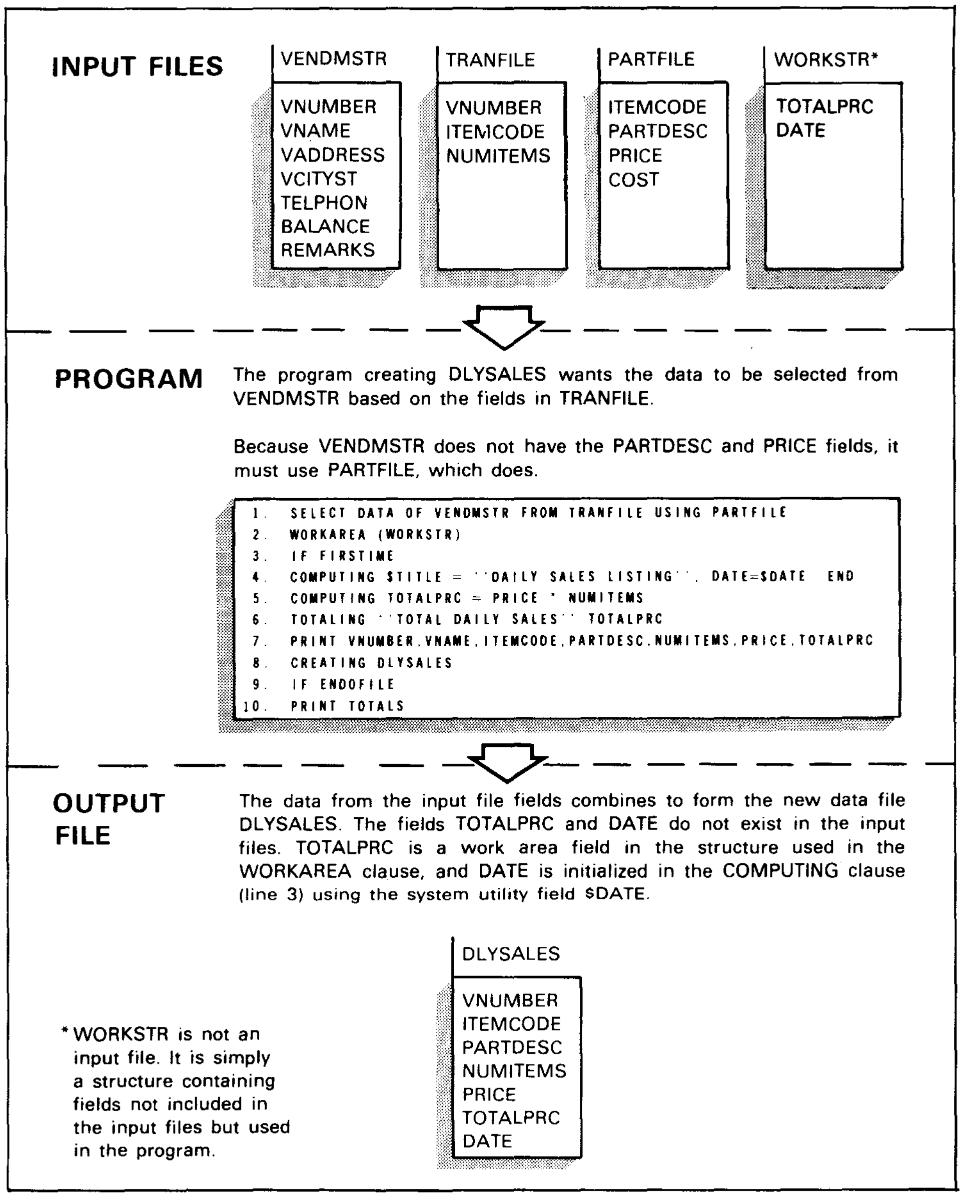


Example of selecting from another file

As shown in Figure 4-1, the program, line by line:

1. Reads a record from the unkeyed file TRANFILE. Then selects a record from the keyed file VENDMSTR, using the value of the key field VNUMBER of TRANFILE as the selection criteria.

   Then selects a record from PARTFILE, using the value of the ITEMCODE field of TRANFILE as the selection criteria.



The program creating DLYSALES wants the data to be selected from VENDMSTR based on the fields in TRANFILE.

Because VENDMSTR does not have the PARTDESC and PRICE fields, it must use PARTFILE, which does.

```
1.  SELECT DATA OF VENDMSTR FROM TRANFILE USING PARTFILE
2.  WORKAREA (WORKSTR)
3.  IF FIRSTIME
4.  COMPUTING $TITLE = ''DAILY SALES LISTING'', DATE=$DATE  END
5.  COMPUTING TOTALPRC = PRICE * NUMITEMS
6.  TOTALING ''TOTAL DAILY SALES'' TOTALPRC
7.  PRINT VNUMBER,VNAME,ITEMCODE,PARTDESC,NUMITEMS,PRICE,TOTALPRC
8.  CREATING DLYSALES
9.  IF ENDOFILE
10. PRINT TOTALS
```

The data from the input file fields combines to form the new data file DLYSALES. The fields TOTALPRC and DATE do not exist in the input files. TOTALPRC is a work area field in the structure used in the WORKAREA clause, and DATE is initialized in the COMPUTING clause (line 3) using the system utility field $DATE.

*WORKSTR is not an input file. It is simply a structure containing fields not included in the input files but used in the program.

Figure 4-1. Example of Selecting from Another File

2. Because you're only using TOTALPRC as a one-time calculation for your output, you create a structure (WORKSTR) for it. Then you designate the name of the structure in the WORKAREA clause. When your calculations are performed, ESCORT knows where to find TOTALPRC.

3. If this is the first time through the processing cycle, performs the clauses on line 4 through END.

   If it's not the first time through the processing cycle, skips to line 5.

4. Assigns values to data fields, using two data replacement statements:

   - $TITLE

     Is the system utility field that prints the title "DAILY SALES LISTING" at the top of your report.

   - DATE

     Is a field in the structure DLYSALES that you assign the value of the system utility field, $DATE. This prints the current system date whenever you print your report.

   END encloses the clauses to be performed if the FIRSTIME clause tests true.

5. Calculates the total price of the transaction.

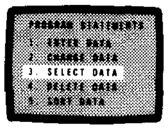
6. Contains the literal "TOTAL DAILY SALES" and accumulates a grand total for the field TOTALPRC.

7. Prints, using automatic formatting, the data fields you list as part of your report.

8. Creates a new file called DLYSALES, using the fields in the DLYSALES structure and the data selected from VENDMSTR, TRANFILE, and PARTFILE.

9. If it's the end of the input file (TRANFILE), performs the clause on line 10.

   If it's not the end of the input file, returns to line 1.

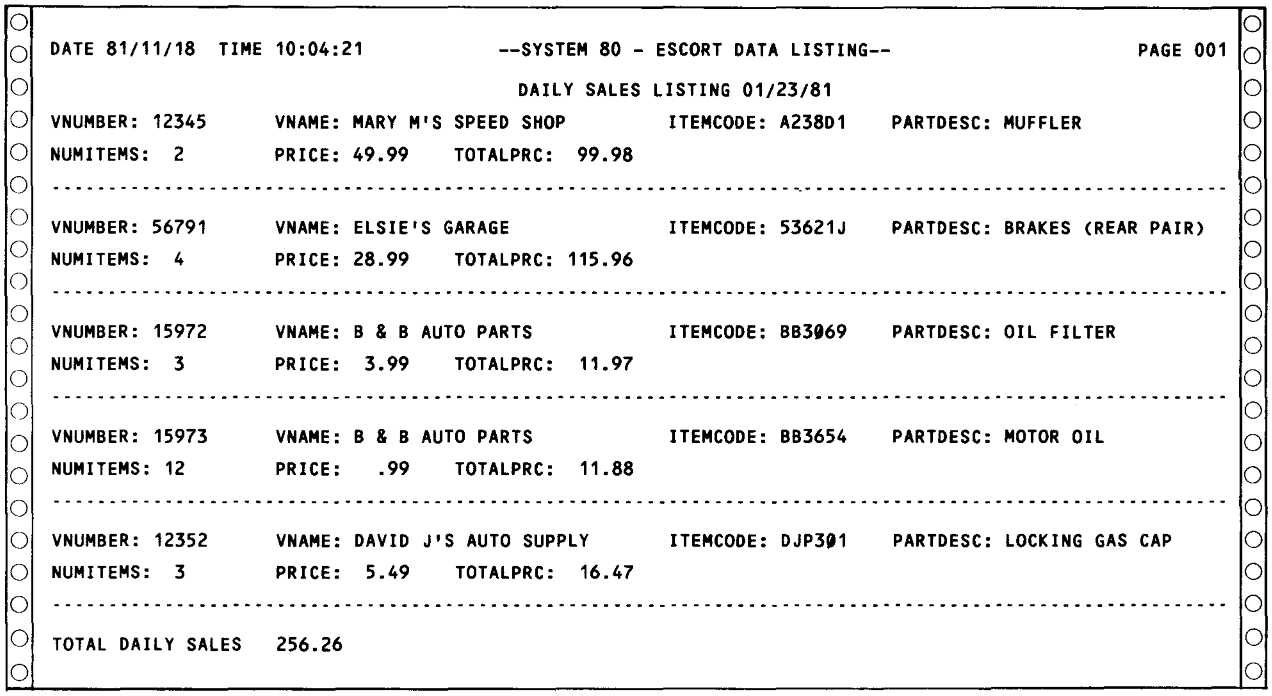


10. Prints the literal "TOTAL DAILY SALES" and the grand total of the field TOTALPRC.

The data shown in the printout is random data that we placed in our files.

Example output

Your output for this program will differ, depending on the data you key into your original files. The sample program produces the following output:



```
DATE 81/11/18  TIME 10:04:21           --SYSTEM 80 - ESCORT DATA LISTING--                 PAGE 001
                                              DAILY SALES LISTING 01/23/81
VNUMBER: 12345      VNAME: MARY M'S SPEED SHOP          ITEMCODE: A238D1    PARTDESC: MUFFLER
NUMITEMS:  2        PRICE: 49.99    TOTALPRC:  99.98
-------------------------------------------------------------------------------------------------
VNUMBER: 56791      VNAME: ELSIE'S GARAGE               ITEMCODE: 53621J    PARTDESC: BRAKES (REAR PAIR)
NUMITEMS:  4        PRICE: 28.99    TOTALPRC: 115.96
-------------------------------------------------------------------------------------------------
VNUMBER: 15972      VNAME: B & B AUTO PARTS             ITEMCODE: BB3Ø69    PARTDESC: OIL FILTER
NUMITEMS:  3        PRICE:  3.99    TOTALPRC:  11.97
-------------------------------------------------------------------------------------------------
VNUMBER: 15973      VNAME: B & B AUTO PARTS             ITEMCODE: BB3654    PARTDESC: MOTOR OIL
NUMITEMS: 12        PRICE:   .99    TOTALPRC:  11.88
-------------------------------------------------------------------------------------------------
VNUMBER: 12352      VNAME: DAVID J'S AUTO SUPPLY        ITEMCODE: DJP3Ø1    PARTDESC: LOCKING GAS CAP
NUMITEMS:  3        PRICE:  5.49    TOTALPRC:  16.47
-------------------------------------------------------------------------------------------------
TOTAL DAILY SALES   256.26
```

4.15. USING THE SCREEN MENU METHOD TO CREATE A DELETE DATA PROGRAM

You use the DELETE DATA statement to remove data records from an existing file. The DELETE DATA statement provides you with three ways to remove data from your files:

Deletion methods

1. Inquiry from the workstation

2. Matching from another file

3. Based on IF conditions

The sample programs illustrating these three methods use the file structures INVREORD and INVDEL.

Example structures

STRUCTURE OF INVREORD

VENDNAME	20A	
VENDADDR	35A	
VENDZIP	5N	
PRODNO	7N	K1
UNITSIZE	10A	
UNITCOST	4V2	
PRODESC	25A	

STRUCTURE OF INVDEL

PRODNO	7N	K1

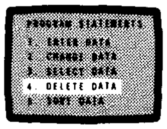
4.16. DELETE DATA BY WORKSTATION INQUIRY

Example definition

The first of the three methods of removing data records from your file is the workstation inquiry method. The sample program instructs ESCORT to provide a prompt field, which in this case is your key field PRODNO. You key in the product number that identifies the record you want to remove, and ESCORT marks it as deleted.

Initiation of DELETE DATA

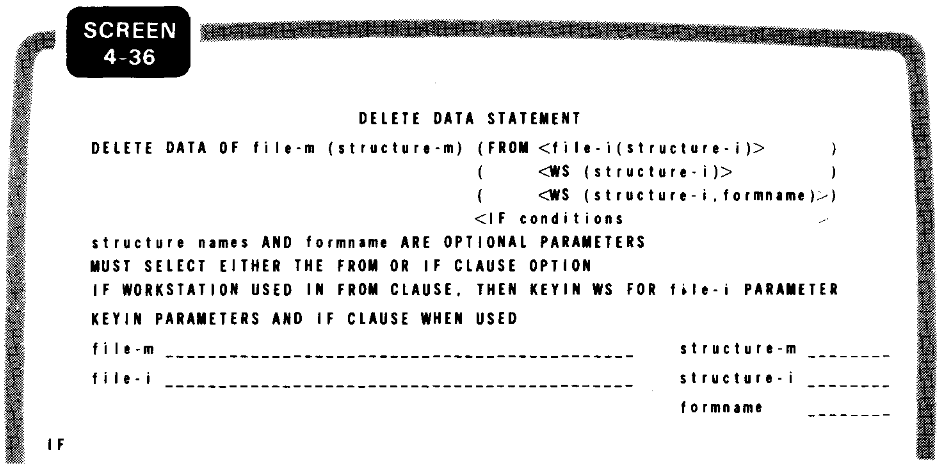
To get the DELETE DATA function, choose number 4 on the PROGRAM STATEMENTS menu (screen 4-2).



This displays the screen containing the format for the DELETE DATA program statement (screen 4-36). (We'll give information on the DELETE DATA program statement in 7.14.)

DELETE DATA statement

SCREEN 4-36



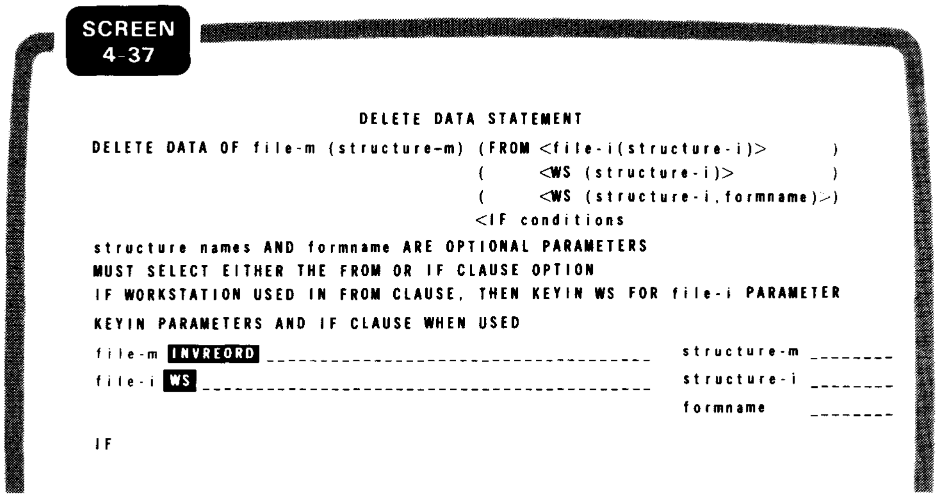
```
                        DELETE DATA STATEMENT
DELETE DATA OF file-m (structure-m) (FROM <file-i(structure-i)>       )
                                    (      <WS (structure-i)>         )
                                    (      <WS (structure-i,formname)>)
                                    <IF conditions                    >
structure names AND formname ARE OPTIONAL PARAMETERS
MUST SELECT EITHER THE FROM OR IF CLAUSE OPTION
IF WORKSTATION USED IN FROM CLAUSE, THEN KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS AND IF CLAUSE WHEN USED
file-m ____________________________________________  structure-m ________
file-i ____________________________________________  structure-i ________
                                                     formname    ________
IF
```

Parameter keyin

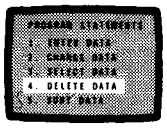
Notice that there are several parameter choices. For the workstation inquiry method, use the *file-m* and *FROM WS* parameters. Fill in the *file-m* parameter with your file name INVREORD and, because you're using the *FROM WS* parameter, key in WS in the *file-i* space. Your screen now looks like screen 4-37.

DELETE DATA parameters

SCREEN 4-37



```
                        DELETE DATA STATEMENT
DELETE DATA OF file-m (structure-m) (FROM <file-i(structure-i)>       )
                                    (      <WS (structure-i)>         )
                                    (      <WS (structure-i,formname)>)
                                    <IF conditions
structure names AND formname ARE OPTIONAL PARAMETERS
MUST SELECT EITHER THE FROM OR IF CLAUSE OPTION
IF WORKSTATION USED IN FROM CLAUSE, THEN KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS AND IF CLAUSE WHEN USED
file-m INVREORD ____________________________________  structure-m ________
file-i WS __________________________________________  structure-i ________
                                                      formname    ________
IF
```



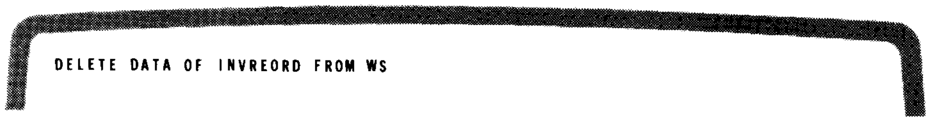
Statement completion

When you complete your DELETE DATA statement, press the **XMIT** key. Because there are no additional clauses you can use with the DELETE DATA statement, ESCORT compiles your program, and the PROGRAM COMMANDS menu (screen 4-35) reappears.

Program display

To get an idea of what your finished program looks like, choose number 4, DISPLAY PROGRAM. Then ESCORT displays your sample DELETE DATA program:

Example program



```
DELETE DATA OF INVREORD FROM WS
```

To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

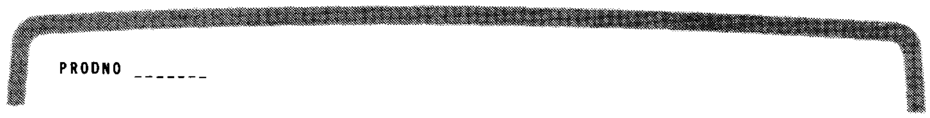
Program execution

Then choose number 2, RUN PROGRAM, to execute your program.

Program results

When your program executes, the records whose keys you entered at the workstation prompt are marked for deletion from the INVREORD file.

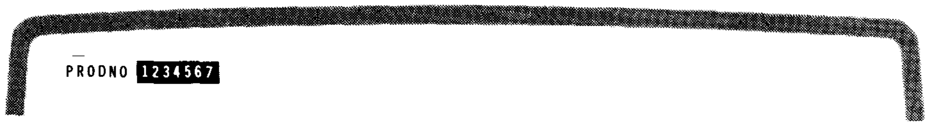
Keyed file deletion prompt



```
PRODNO _______
```

To mark the record for deletion, key in the product number of the record you want removed in the space allotted.

Record number to be deleted



```
PRODNO 1234567
```

4.17. DELETE DATA BY FILE MATCHING

Creating transactional files

When you want to use the DELETE DATA statement for file matching, you create a transactional file much like the one you created in the second CHANGE DATA example: CHANGE DATA by replacing (4.10). Then you enter data into this file by using an ENTER DATA program (3.5).

Example definition

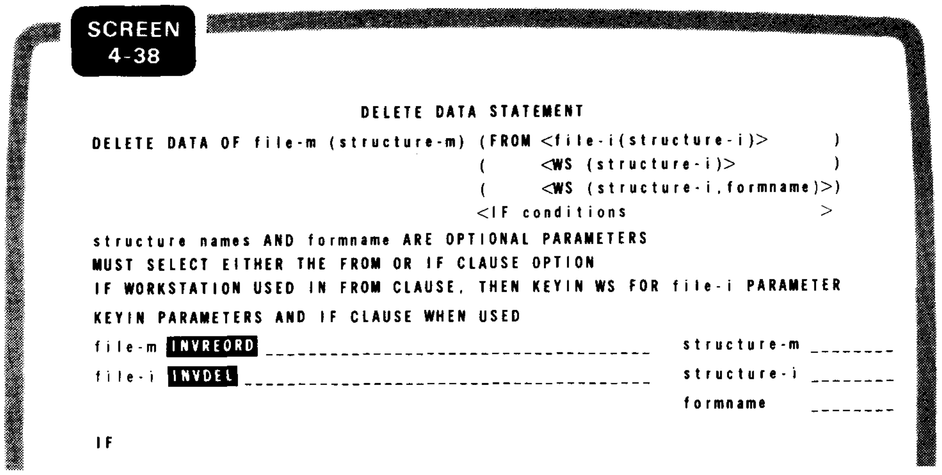
The structure for this transactional file is INVDEL (4.15). ESCORT matches a file record from INVDEL to a file record from INVREORD, using the key field PRODNO. When the record is found, ESCORT marks it as deleted. This process continues until ESCORT reaches the end of the input file (INVDEL), and then the program terminates.

Parameter keyin

For the file matching method of data deletion, you use the *file-m* parameter and the *FROM file-i* parameter (screen 4-38). Fill in the *file-m* parameter with the name of the file from which you want the records deleted (INVREORD), and fill in the *file-i* parameter with the name of the transactional file (INVDEL).

DELETE DATA statement

SCREEN 4-38

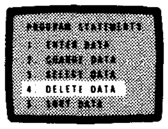


```
                        DELETE DATA STATEMENT
DELETE DATA OF file-m (structure-m) (FROM <file-i(structure-i)>        )
                                    (      <WS (structure-i)>          )
                                    (      <WS (structure-i,formname)>)
                                    <IF conditions                     >
structure names AND formname ARE OPTIONAL PARAMETERS
MUST SELECT EITHER THE FROM OR IF CLAUSE OPTION
IF WORKSTATION USED IN FROM CLAUSE, THEN KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS AND IF CLAUSE WHEN USED
file-m INVREORD ______________________________  structure-m ________
file-i INVDEL _________________________________  structure-i ________
                                                 formname    ________

IF
```

Statement completion

Program compilation

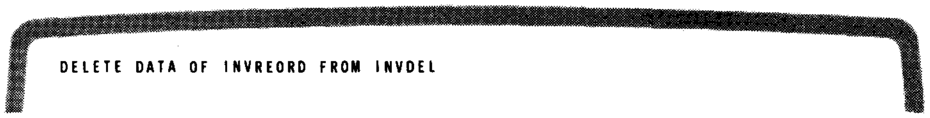
When you complete your DELETE DATA statement, press the **XMIT** key. Because there are no additional clauses you can use with the DELETE DATA statement, ESCORT compiles your program, and the PROGRAM COMMANDS menu (screen 4-35) appears.



Program display

To get an idea of what your program looks like, choose number 4, DISPLAY PROGRAM, and key in your program name. Then, ESCORT displays your program.

Example program



```
DELETE DATA OF INVREORD FROM INVDEL
```

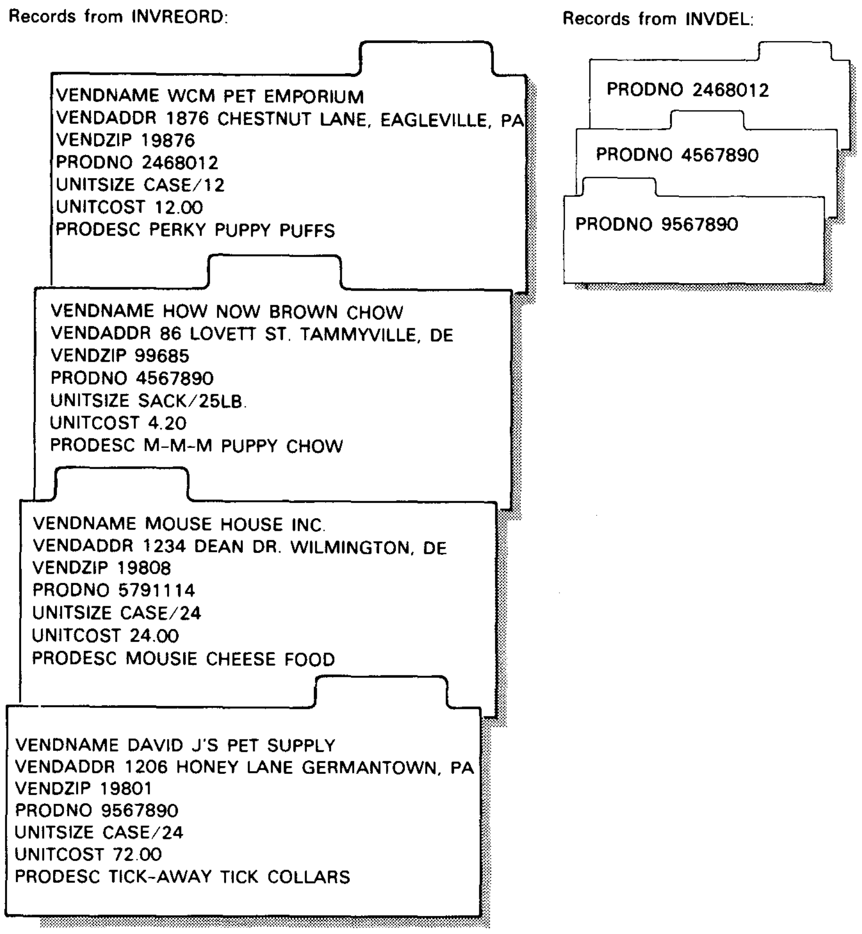
To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

Program execution

Then choose number 2, RUN PROGRAM, on the PROGRAM COMMANDS menu to execute your program.

Program action

When your program is executed, the records of INVREORD to be deleted are selected and marked as deleted, using the matching key PRODNO from the file INVDEL.

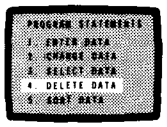


Records from INVREORD:

```
VENDNAME WCM PET EMPORIUM
VENDADDR 1876 CHESTNUT LANE, EAGLEVILLE, PA
VENDZIP 19876
PRODNO 2468012
UNITSIZE CASE/12
UNITCOST 12.00
PRODESC PERKY PUPPY PUFFS

VENDNAME HOW NOW BROWN CHOW
VENDADDR 86 LOVETT ST. TAMMYVILLE, DE
VENDZIP 99685
PRODNO 4567890
UNITSIZE SACK/25LB.
UNITCOST 4.20
PRODESC M-M-M PUPPY CHOW

VENDNAME MOUSE HOUSE INC.
VENDADDR 1234 DEAN DR. WILMINGTON, DE
VENDZIP 19808
PRODNO 5791114
UNITSIZE CASE/24
UNITCOST 24.00
PRODESC MOUSIE CHEESE FOOD

VENDNAME DAVID J'S PET SUPPLY
VENDADDR 1206 HONEY LANE GERMANTOWN, PA
VENDZIP 19801
PRODNO 9567890
UNITSIZE CASE/24
UNITCOST 72.00
PRODESC TICK-AWAY TICK COLLARS
```

Records from INVDEL:

```
PRODNO 2468012

PRODNO 4567890

PRODNO 9567890
```



Program results

As a result of your program's execution, records 1, 2, and 4 are marked as deleted because their product numbers (PRODNO) match. Record 3, for Mouse House Inc., is not deleted because there is not a corresponding record in INVDEL that specifies that it should be deleted.

4.18. DELETE DATA BASED ON IF CONDITIONS

Purpose

Removing data records based on IF conditions means that you want to ensure that certain criteria are met before the deletion can occur.

Example definition

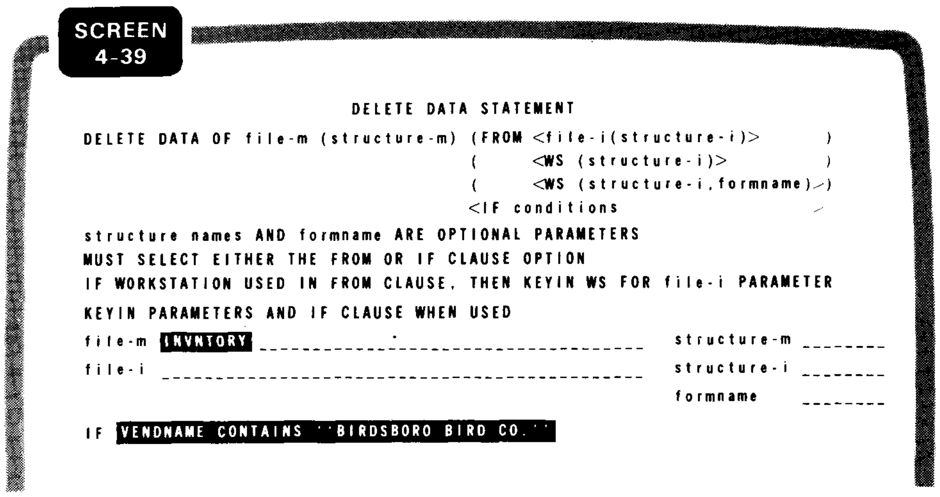
For this example, suppose one of your vendors goes out of business and you want to remove all inventory items supplied by that vendor from your INVNTORY master file. To do this, you use the DELETE DATA program statement and its corresponding IF clause (screen 4-39).

Parameter keyin

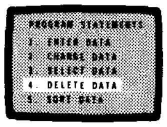
For the IF conditional method of data deletion, simply use the *file-m* parameter and the *IF conditions* parameter. Fill in the *file-m* parameter with the name of the file where you want the records deleted (INVNTORY). Then, fill in the *IF* parameter with the conditional clause of your choice. For this example, assume the vendor's name is the Birdsboro Bird Co. Your DELETE DATA screen for this example now looks like this:

DELETE DATA statement

SCREEN 4-39



```
                        DELETE DATA STATEMENT
DELETE DATA OF file-m (structure-m) (FROM <file-i(structure-i)>          )
                                    (     <WS (structure-i)>             )
                                    (     <WS (structure-i,formname)>    )
                                    <IF conditions                       >
structure names AND formname ARE OPTIONAL PARAMETERS
MUST SELECT EITHER THE FROM OR IF CLAUSE OPTION
IF WORKSTATION USED IN FROM CLAUSE, THEN KEYIN WS FOR file-i PARAMETER
KEYIN PARAMETERS AND IF CLAUSE WHEN USED
file-m INVNTORY ______________________________     structure-m ________
file-i ______________________________________      structure-i ________
                                                   formname    ________

IF VENDNAME CONTAINS ''BIRDSBORO BIRD CO.''
```



Statement completion

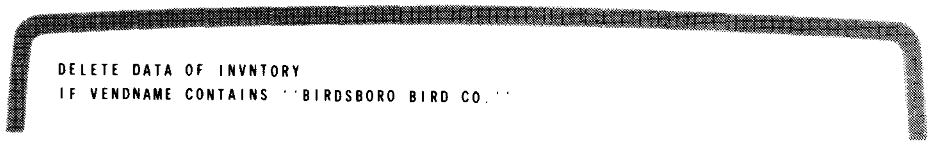
Program compilation

When you complete your DELETE DATA statement, press the **XMIT** key. Because there are no additional clauses you can use with the DELETE DATA statement, ESCORT compiles your program, and the PROGRAM COMMANDS menu appears (screen 4-35).

Program display

To get an idea of what your program looks like, choose number 4, DISPLAY PROGRAM. Then, ESCORT displays your program:

Example program



```
DELETE DATA OF INVNTORY
IF VENDNAME CONTAINS ''BIRDSBORO BIRD CO.''
```

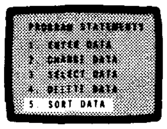
To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

Program execution

Then choose number 2, RUN PROGRAM, on the PROGRAM COMMANDS menu to execute your program.

Program results

The result of your program's execution is that every record containing the words "Birdsboro Bird Co." in the VENDNAME field is marked for deletion.



4.19. USING THE SCREEN MENU METHOD TO CREATE TWO KINDS OF SORT DATA PROGRAMS

Sort key definition

Sort key usage

The file structures for the previous examples in this section all contain at least one sort key. This means that they are keyed files and, when you reference them, the fields designated as key fields are output. When you select or change these records, you use these sort keys to determine which records you want to access. (We'll explain sort keys in 6.7.)

Nonindexed files

When you don't designate sort keys in a file, you have a nonindexed, sequential file. This means the records are written into your file in the exact order you enter them. This is not a very efficient method for a file because your records usually cannot be output in any meaningful order.

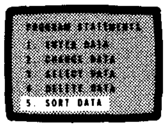
SORT DATA statement usage

The SORT DATA program statement lets you take a nonindexed, sequential file and create a new, sorted file that you can use to output reports or data listings.

Variations of the SORT DATA statement

There are two variations of the SORT DATA program statement:

- SORT DATA by designating field names
- SORT DATA by matching structure field



4.20. SORT DATA BY DESIGNATING FIELD NAMES

Example definition

This sample SORT DATA program uses the structure INVNTORY minus the sort key (PRODNO). It then creates a new file called MAILIST, which contains an alphabetized mailing list of suppliers. Because the list is alphabetized, the sample program uses the field VENDNAME as the sort field.

Example structures

STRUCTURE OF INVNTORY

PRODNO	7N
PRODESC	25A
UNITSIZE	10A
UNITCOST	4V2
MARKUP	1V2
RETPRICE	5V2
QONHAND	5N
QONORDER	5N
REORDERL	2N
QUANDISC	1V2
VENDNAME	20A
VENDADDR	35A
VENDZIP	5N

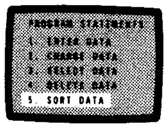
STRUCTURE OF MAILIST

VENDNAME	20A
VENDADDR	35A
VENDZIP	5N

We'll assume you've chosen the ENTER PROGRAM command from the PROGRAM COMMANDS menu (screen 4-1) and are now at the PROGRAM STATEMENTS menu (screen 4-2).

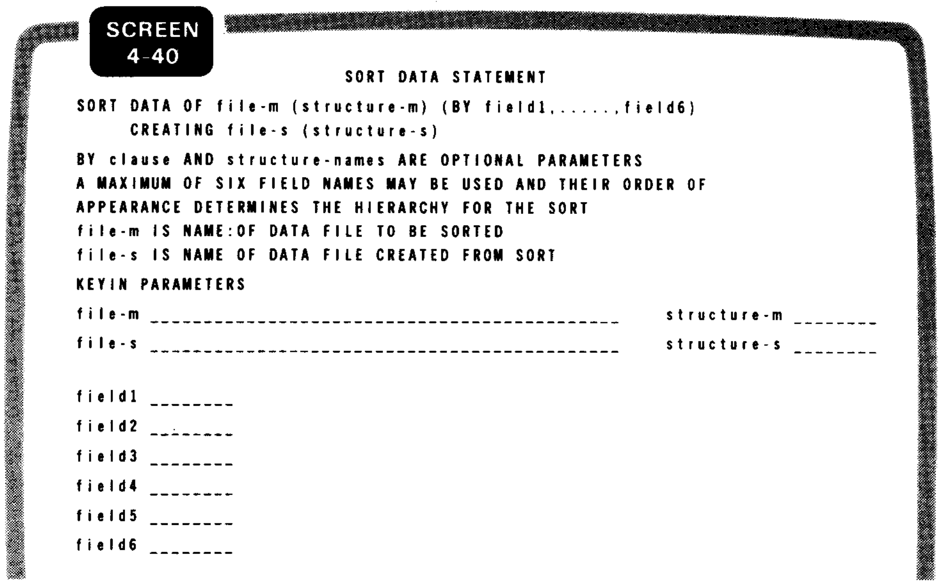
Initiation of SORT DATA

To get the SORT DATA function, choose number 5 on the PROGRAM STATEMENTS menu (screen 4-2).



This displays screen 4–40, which contains the format of the SORT DATA program statement. (We'll give more information on the SORT DATA statement in 7.13.)

SORT DATA statement



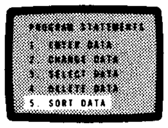
```
                    SORT DATA STATEMENT
SORT DATA OF file-m (structure-m) (BY field1,......,field6)
     CREATING file-s (structure-s)
BY clause AND structure-names ARE OPTIONAL PARAMETERS
A MAXIMUM OF SIX FIELD NAMES MAY BE USED AND THEIR ORDER OF
APPEARANCE DETERMINES THE HIERARCHY FOR THE SORT
file-m IS NAME:OF DATA FILE TO BE SORTED
file-s IS NAME OF DATA FILE CREATED FROM SORT
KEYIN PARAMETERS
file-m ____________________________________________   structure-m ________
file-s ____________________________________________   structure-s ________

field1 ________
field2 ________
field3 ________
field4 ________
field5 ________
field6 ________
```

Required parameters

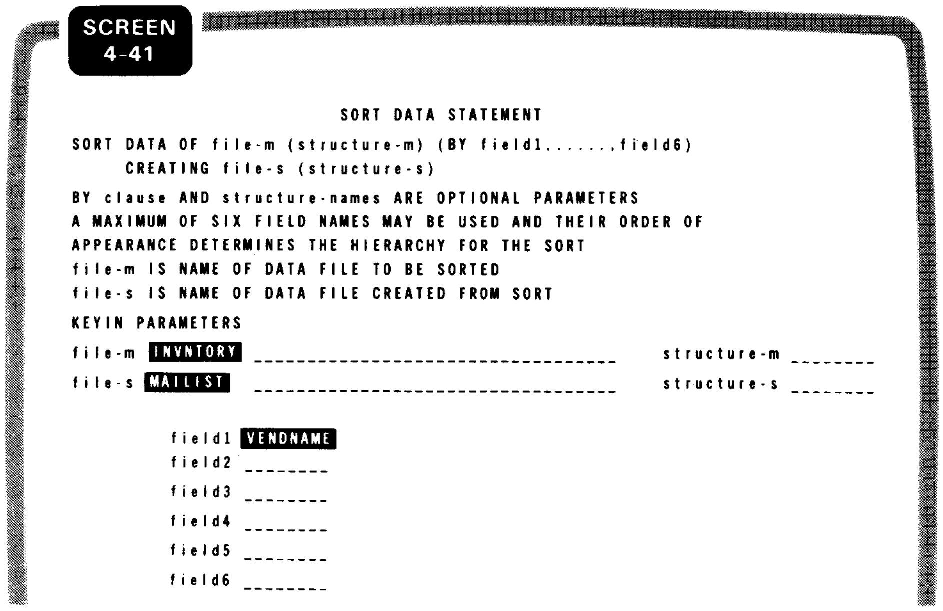
Notice that the only required parameters are *file-m* and the CREATING clause. The *file-m* parameter names the file containing the data to be sorted. The CREATING clause names the new sorted file. The *structure-m and structure-s* parameters are optional. You use them only if the structure name is different from the first eight characters of the file name or you want to use a different structure to output specific fields of a file. The BY clause, which lets you designate the fields to sort your data, is optional.

Parameter keyin

For this example, fill in the *file-m* parameter with the input file name INVNTORY. Also fill in the *BY field1* parameter with the field name VENDNAME, because you want to sort your list alphabetically. Then key in the *file-s* parameter with the output file name MAILIST. Your SORT DATA screen now looks like screen 4–41.



SORT DATA statement


SCREEN 4-41

```
                    SORT DATA STATEMENT
SORT DATA OF file-m (structure-m) (BY field1,.....,field6)
    CREATING file-s (structure-s)
BY clause AND structure-names ARE OPTIONAL PARAMETERS
A MAXIMUM OF SIX FIELD NAMES MAY BE USED AND THEIR ORDER OF
APPEARANCE DETERMINES THE HIERARCHY FOR THE SORT
file-m IS NAME OF DATA FILE TO BE SORTED
file-s IS NAME OF DATA FILE CREATED FROM SORT
KEYIN PARAMETERS
file-m INVNTORY ___________________________   structure-m ________
file-s MAILIST  ___________________________   structure-s ________

        field1 VENDNAME
        field2 ________
        field3 ________
        field4 ________
        field5 ________
        field6 ________
```

Statement completion

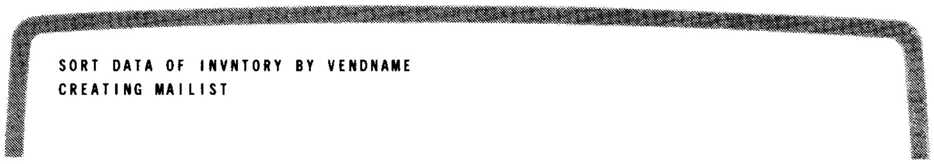
Program compilation

When you complete your SORT DATA statement, press the **XMIT** key. Because there are no additional clauses you can use with the SORT DATA statement, ESCORT compiles your program, and the PROGRAM COMMANDS menu (screen 4-2) appears.

Program display

To get an idea of what your finished program looks like, select number 4, DISPLAY PROGRAM, and key in your program name. Then, ESCORT displays your sample SORT DATA program:

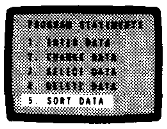
Example program


```
SORT DATA OF INVNTORY BY VENDNAME
CREATING MAILIST
```

To get back to the PROGRAM COMMANDS menu, press the **XMIT** key.

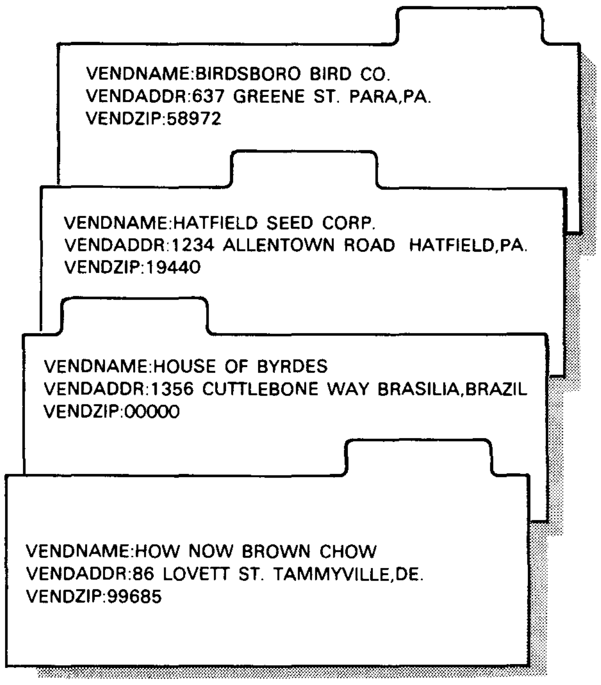
Program execution

Then choose number 2, RUN PROGRAM, on the PROGRAM COMMANDS menu to execute your program.



Program results

The result of your program's execution is that the fields VENDNAME, VENDADDR, and VENDZIP of the unordered file INVNTORY are written into the MAILIST file in alphabetical order. The records of MAILIST are:



```
VENDNAME:BIRDSBORO BIRD CO.
VENDADDR:637 GREENE ST. PARA,PA.
VENDZIP:58972
```

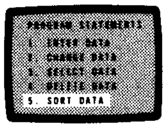
```
VENDNAME:HATFIELD SEED CORP.
VENDADDR:1234 ALLENTOWN ROAD  HATFIELD,PA.
VENDZIP:19440
```

```
VENDNAME:HOUSE OF BYRDES
VENDADDR:1356 CUTTLEBONE WAY BRASILIA,BRAZIL
VENDZIP:00000
```

```
VENDNAME:HOW NOW BROWN CHOW
VENDADDR:86 LOVETT ST. TAMMYVILLE,DE.
VENDZIP:99685
```

Printing the file

Notice you can't print your new sorted data file by using the SORT DATA program statement. This is because the PRINT and OUTPUT TO PRINTER clauses can't be used with this statement. You can, however, print your new file by using a SELECT DATA program (SELECT DATA OF MAILIST AND PRINT).



4.21. SORT DATA BY MATCHING STRUCTURE FIELD

Another way to use the SORT DATA program statement is to designate a field as matching sorted (MS) in the new structure designated for your sorted file.

Example definition

When you specify the structure (MAILIST), you pick the field you want to sort by (VENDNAME) and, in the KEYS portion of your ENTER STRUCTURE screen (screen 6-1), key in MS (matching sorted).

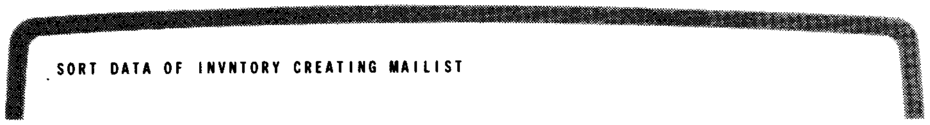
Your structure is:

Example structure

```
STRUCTURE OF MAILIST

VENDNAME       20A          MS
VENDADDR       35A
VENDZIP        5N
```

Example program

Then you can use the following program:



```
.SORT DATA OF INVNTORY CREATING MAILIST
```

ESCORT action

ESCORT creates your sorted MAILIST file by using the key you designated (VENDNAME), and your output is the same as with the previous program (4.20).

The next subsection shows how you can use the direct entry method to enter your program more quickly.

4.22. PROGRAM MODE - THE DIRECT ENTRY METHOD

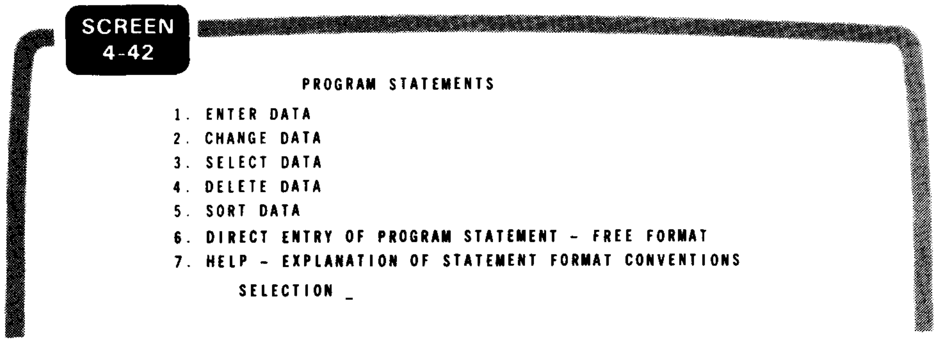
Direct Entry of Program Statements

Screen format

With the direct entry method, you have no statement or clause formats, only a blank screen, onto which you type your program statements.

How to access

When you want to use the direct entry method, you begin your ESCORT session the same as you do for the screen menu method (4.2). When the PROGRAM COMMANDS menu (screen 4-1) is displayed, you choose the ENTER PROGRAM command and key in a program name. Then the PROGRAM STATEMENTS menu (screen 4-42) is displayed:


SCREEN 4-42

```
                    PROGRAM STATEMENTS
1. ENTER DATA
2. CHANGE DATA
3. SELECT DATA
4. DELETE DATA
5. SORT DATA
6. DIRECT ENTRY OF PROGRAM STATEMENT - FREE FORMAT
7. HELP - EXPLANATION OF STATEMENT FORMAT CONVENTIONS
     SELECTION _
```

Choosing direct entry of statements

You get the direct entry method by choosing number 6, DIRECT ENTRY OF PROGRAM STATEMENT, from this screen.

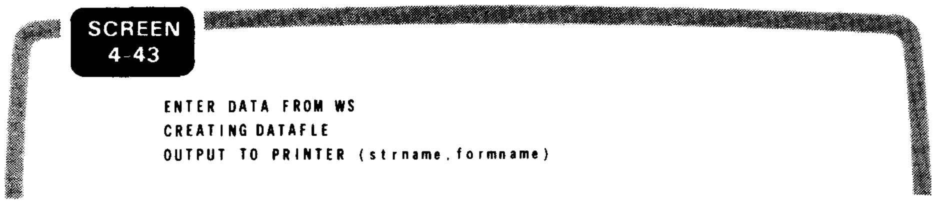
Then an empty screen appears, and you key in a program.

Program length

The type and length of the program are your choice.

Screen 4-43 shows an example of the direct entry method.

Direct entry example


SCREEN 4-43

```
ENTER DATA FROM WS
CREATING DATAFLE
OUTPUT TO PRINTER (strname.formname)
```

Keyin format

Note that each program statement or clause need not appear on a new line; however, it makes it easier to keep track of where you are in the program during keyin.

Direct Entry of Clauses

How to access

Choosing direct entry of clauses

When you use the screen menu method of entering your program, you also have the option to use DIRECT ENTRY OF CLAUSES. With this method, you use the screen menu formats to complete the program statement parameters and when you get to the clause selection screen for that program statement type, choose the appropriate number to enter the clauses directly. Because only the ENTER, CHANGE, and SELECT program statements have additional clauses you can use, there is no DIRECT ENTRY OF CLAUSES for the SORT or DELETE program statements. When you choose direct entry of clauses, a blank screen appears, and you key in the clauses you want to use. You are limited to 10 lines per screen.

How to terminate direct entry

To terminate, press the **XMIT** key and when a blank screen appears, press the **FUNCTION** and **F4** keys.

PART
4

SPECIFICS OF PROGRAM MODE

5. ESCORT Commands

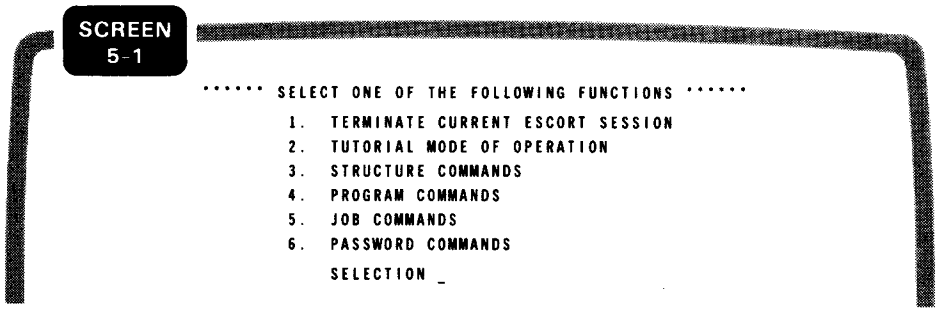
5.1. COMMAND SELECTION

Use

Before you can enter or access a program, structure, job, or data file password, you must first use an ESCORT command.

Once you choose program mode (see Section 4), the command selection menu (screen 5-1) is displayed:

SCREEN 5-1



```
****** SELECT ONE OF THE FOLLOWING FUNCTIONS ******
         1.  TERMINATE CURRENT ESCORT SESSION
         2.  TUTORIAL MODE OF OPERATION
         3.  STRUCTURE COMMANDS
         4.  PROGRAM COMMANDS
         5.  JOB COMMANDS
         6.  PASSWORD COMMANDS
             SELECTION _
```

Command selection

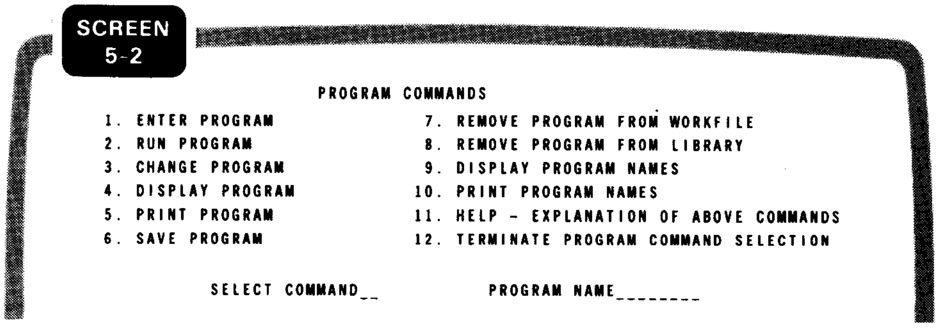
Select the command category you want by keying in the appropriate number and pressing the **XMIT** key.

Command menu

Example use: programs

This displays a menu listing all the commands in the category you choose. For example, suppose you want to enter a program. Key in 4, and the PROGRAM COMMANDS menu (screen 5-2) appears:

SCREEN 5-2



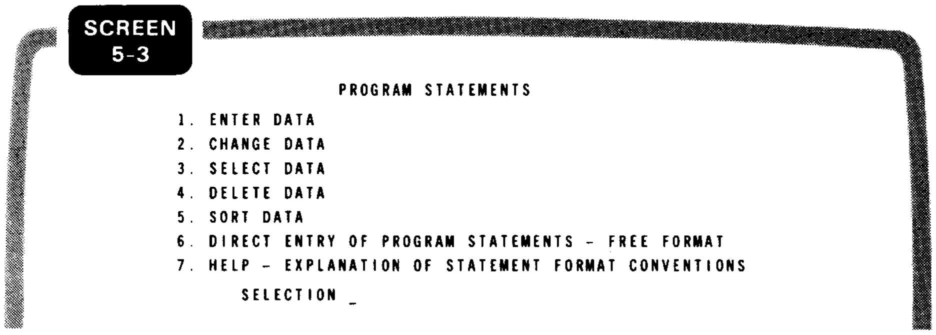
```
                      PROGRAM COMMANDS
 1. ENTER PROGRAM              7. REMOVE PROGRAM FROM WORKFILE
 2. RUN PROGRAM                8. REMOVE PROGRAM FROM LIBRARY
 3. CHANGE PROGRAM             9. DISPLAY PROGRAM NAMES
 4. DISPLAY PROGRAM           10. PRINT PROGRAM NAMES
 5. PRINT PROGRAM             11. HELP - EXPLANATION OF ABOVE COMMANDS
 6. SAVE PROGRAM              12. TERMINATE PROGRAM COMMAND SELECTION

           SELECT COMMAND__          PROGRAM NAME________
```

ENTER PROGRAM command

Then choose 1, ENTER PROGRAM, and enter the program name.

Note that when you choose the ENTER command (ENTER PROGRAM, ENTER STRUCTURE, ENTER JOB, ENTER PASSWORDS), you're choosing to initially key in a program, structure, job, or data file passwords. When you choose any of the other commands (DISPLAY, CHANGE, etc), you're just manipulating an existing program, structure, job or password.

When you choose ENTER PROGRAM, screen 5-3 appears:



```
                    PROGRAM STATEMENTS
1. ENTER DATA
2. CHANGE DATA
3. SELECT DATA
4. DELETE DATA
5. SORT DATA
6. DIRECT ENTRY OF PROGRAM STATEMENTS - FREE FORMAT
7. HELP - EXPLANATION OF STATEMENT FORMAT CONVENTIONS
     SELECTION _
```

Program statement choice

Pick the program statement you want to use. You then proceed through the sequence of screens you need to get the results you want. (We'll give more information on program statements in Section 7.)

Other command types

When you choose another type of command (structure, job, or passwords), the process is the same as in the preceding example.

Additional commands

The only three commands not related to a type are HELP, TUTORIAL, and TERMINATE. These commands are also selected by menu choices.

On screen 5-1, the TUTORIAL command is choice 2, and TERMINATE is choice 1. The HELP command is available as a selection on many of the ESCORT menus.

5.2. TERMS YOU SHOULD KNOW

Before you proceed with this section, familiarize yourself with the following terms, which appear in command descriptions.

RE$IDENT program and job

- RE$IDENT Program or Job

  When you enter or change an unnamed program or job, ESCORT assigns the default name RE$IDENT to it. There can be only one RE$IDENT program or job per ESCORT session. When you enter another one, you lose the original one (overlay), and the one you enter becomes RE$IDENT.

Session file

- Session File

  This file contains everything (programs, structures, jobs) you enter or change during an ESCORT session (your current session).

Library

- Library

  The term *library* refers to the specific ESCORT library designated to save your programs, structures, and jobs.

There are four types of libraries where different types of ESCORT data are stored:

Library types

- E$CS – for source programs
- E$CO – for object programs
- E$CJ – for jobs
- E$FD – for structures

You will never see these types displayed when you're using ESCORT; however, if you use the interactive services FSTAT command, these types are listed as ESCORT files.

Additional type

In addition to these four library types, there exists an E$AF type. E$AF is a file created as part of E$CO when your object programs use the automatic formatting PRINT or DISPLAY clauses.

Whenever you copy, rename, or delete any of these object programs, you must also copy, rename, or delete the E$AF file. You do this outside of ESCORT using interactive services. You will see the E$AF name displayed when you use the DISPLAY PROGRAM NAMES command.

PROGRAM COMMANDS

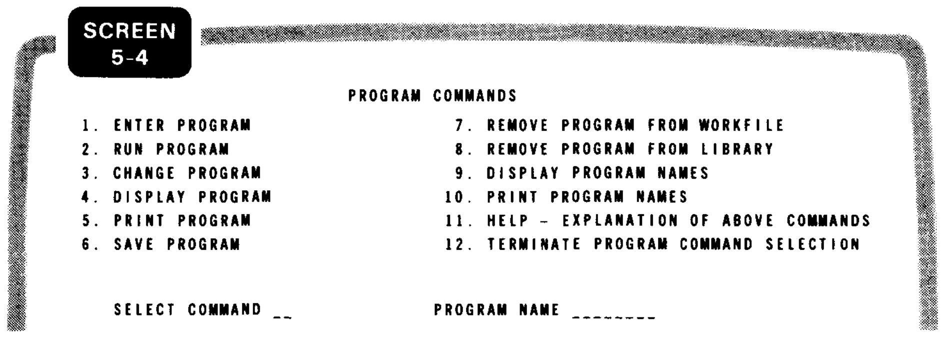
5.3. PROGRAM COMMANDS

Program makeup

Programs, as you remember, are comprised of an ESCORT program statement and its related clauses and are the means for solving your business problems. Before you can enter or access a program, you must use an ESCORT command.

Program commands menu

The program commands appear on the following menu (screen 5-4):



SCREEN 5-4

```
                         PROGRAM COMMANDS
1. ENTER PROGRAM                      7. REMOVE PROGRAM FROM WORKFILE
2. RUN PROGRAM                        8. REMOVE PROGRAM FROM LIBRARY
3. CHANGE PROGRAM                     9. DISPLAY PROGRAM NAMES
4. DISPLAY PROGRAM                   10. PRINT PROGRAM NAMES
5. PRINT PROGRAM                     11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE PROGRAM                      12. TERMINATE PROGRAM COMMAND SELECTION

   SELECT COMMAND __              PROGRAM NAME ________
```

Command selection

Key in the number of the command you want and the program name and press the **XMIT** key. We'll explain how to use program commands in 5.4 through 5.13.

5.4. HOW TO CHANGE YOUR PROGRAM (CHANGE PROGRAM COMMAND)

Purpose

Use this command to change a named program stored on your session file or the ESCORT library or to change the RE$IDENT program.

Named programs

When you name the program you want to change, ESCORT:

1. Gets the program from your session file or the ESCORT library
2. Lets you make your changes
3. Puts a copy of the changed program into your session file

NOTE:

The changes are made only to the session file copy of your program. To make the changes permanent, you must re-save the program under the same name.

Unnamed programs

When you don't name the program you want to change, you make the changes on the RE$IDENT program.

PROGRAM COMMANDS

5.5. HOW TO DISPLAY A PROGRAM ON THE WORKSTATION (DISPLAY PROGRAM COMMAND)

Purpose

Use this command to instruct ESCORT to display a program stored on your session file or the ESCORT library on the workstation.

Named programs

When you name the program you want to display, ESCORT:

1. Gets the program from your session file or the ESCORT library
2. Displays your program

Unnamed programs

When you don't name your program, ESCORT displays the RE$IDENT program.

5.6. HOW TO DISPLAY A LISTING OF PROGRAM NAMES (DISPLAY PROGRAM NAMES COMMAND)

Purpose

Use this command to display, on the workstation, a list of all source and object program names contained in both your session file and the ESCORT library.

E$AF

When you use the DISPLAY PROGRAM NAMES command, you'll notice the additional name E$AF. This is a file created as part of the object files (E$CO) when your programs use the automatic formatting PRINT or DISPLAY clauses. For more on E$AF, see 5.2.

5.7. HOW TO ENTER A PROGRAM (ENTER PROGRAM COMMAND)

Purpose

You use this command to enter a program statement from the workstation.

Named programs

When you name your program, the program statement is saved in the current session file under that name, and you can access it until you terminate your session.

Unnamed programs

When you don't name your program, ESCORT assigns a default name of RE$IDENT to your program, and it remains the RE$IDENT program until you enter another nameless program that overwrites it.

PROGRAM
COMMANDS

5.8. HOW TO PRINT A PROGRAM (PRINT PROGRAM COMMAND)

Purpose

Use this command to print a program stored on your session file or the ESCORT library.

Named programs

When you name the program you want to print, ESCORT:

1. Gets the program from your session file or the ESCORT library
2. Prints your program

Unnamed programs

When you don't name the program you want to print, ESCORT prints the RE$IDENT program.

5.9. HOW TO PRINT A LISTING OF PROGRAM NAMES (PRINT PROGRAM NAMES COMMAND)

Purpose

Use this command to print, on the printer, a list of all source and object program names contained in both the ESCORT library and your session file.

5.10. HOW TO REMOVE A PROGRAM FROM THE SESSION FILE (REMOVE PROGRAM FROM WORKFILE COMMAND)

Purpose

You use this command to instruct ESCORT to remove the program you name from your session file.

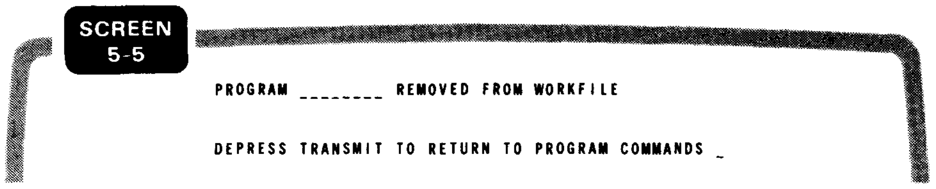
Action

When you specify this command, ESCORT:

1. Removes both the source language and object programs from the session file.
2. Displays a screen (screen 5-5) confirming that the program has been removed from the session file.

SCREEN 5-5

Program removed from workfile screen



```
PROGRAM ________ REMOVED FROM WORKFILE

DEPRESS TRANSMIT TO RETURN TO PROGRAM COMMANDS _
```

PROGRAM COMMANDS

5.11. HOW TO REMOVE A PROGRAM FROM THE ESCORT LIBRARY (REMOVE PROGRAM FROM LIBRARY COMMAND)

Purpose

You use this command to instruct ESCORT to remove the program you name from the ESCORT library.

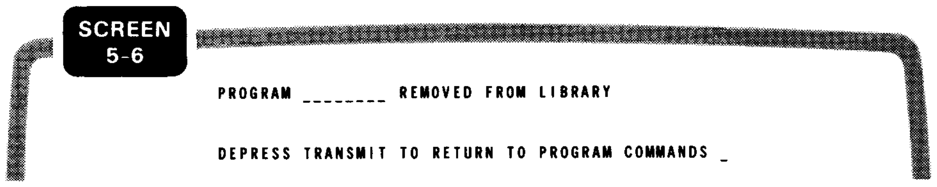
Action

When you choose this command, ESCORT:

1. Removes from the ESCORT library the source language and/or object program (whatever was previously saved).

2. Displays a screen (screen 5-6) confirming that the program has been removed from the ESCORT library.

Program removed from library screen


SCREEN 5-6

```
PROGRAM ________ REMOVED FROM LIBRARY

DEPRESS TRANSMIT TO RETURN TO PROGRAM COMMANDS _
```

5.12. HOW TO EXECUTE YOUR PROGRAM (RUN PROGRAM COMMAND)

Purpose

You use this command to execute the RE$IDENT program or one stored on your session file or the ESCORT library.

Named programs

When you name the program you want to run, ESCORT:

1. Gets the program from your session file or the ESCORT library

2. Executes your program

Unnamed programs

When you don't name your program, ESCORT executes the RE$IDENT program.

PROGRAM COMMANDS

5.13. HOW TO SAVE YOUR PROGRAMS (SAVE PROGRAM COMMAND)

Purpose

You use this command to save your program statements in the ESCORT library. ESCORT programs appear in the library under the types:

Library types

- E$CS – for source programs
- E$CO – for object programs
- E$AF – an addition of E$CO used when your object programs contain automatic formatting clauses (PRINT and DISPLAY)

For an explanation of library types, see 5.2.

When you use the SAVE PROGRAM command, you can save the source program, the object program, or both.

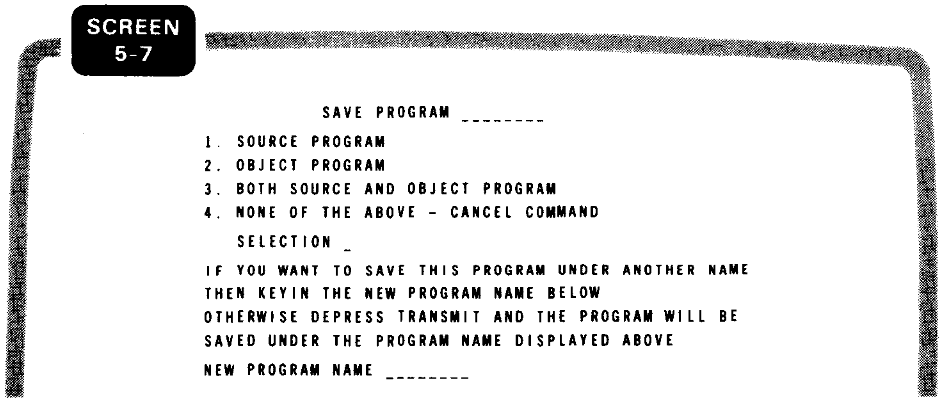
The source program is the program you key in before it is compiled; the object program is the compiled program.

Named programs

If you named the program you want to save on the PROGRAM COMMANDS menu (screen 5-4), ESCORT:

1. Displays screen 5-7, asking what you want to save. (The program name you specified on screen 5-4 appears after the SAVE PROGRAM heading.)

SCREEN 5-7



```
                    SAVE PROGRAM ________
1. SOURCE PROGRAM
2. OBJECT PROGRAM
3. BOTH SOURCE AND OBJECT PROGRAM
4. NONE OF THE ABOVE - CANCEL COMMAND
   SELECTION _
IF YOU WANT TO SAVE THIS PROGRAM UNDER ANOTHER NAME
THEN KEYIN THE NEW PROGRAM NAME BELOW
OTHERWISE DEPRESS TRANSMIT AND THE PROGRAM WILL BE
SAVED UNDER THE PROGRAM NAME DISPLAYED ABOVE
NEW PROGRAM NAME ________
```

2. Gets a copy of that program from your session file

3. Puts it into the ESCORT library file under its own name or a new one you designate in the space provided

NOTE:

Even though you've saved your program, a copy remains on your session file until you terminate your session.

Unnamed programs

When you don't name your program in the ENTER PROGRAM command but use the system-specified name, RE$IDENT, your screen heading says SAVE PROGRAM RE$IDENT.

Because you can't save a program under the name RE$IDENT, you must rename it in the space allotted for NEW PROGRAM NAME.

STRUCTURE
COMMANDS

5.14. STRUCTURE COMMANDS

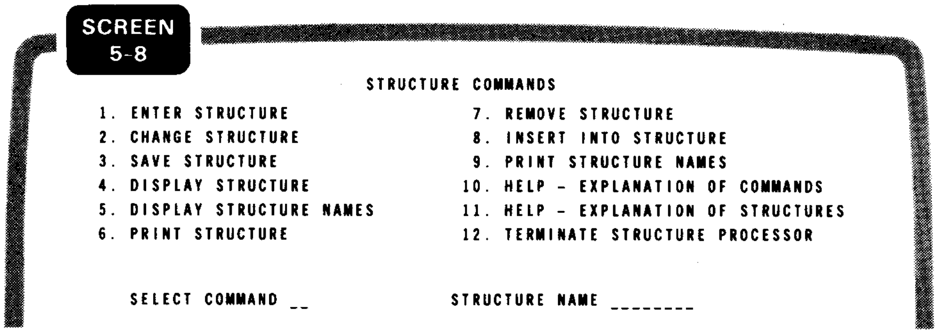
Structure makeup

Structures consist of structure statements that tell ESCORT what your file records look like. Structures are described in detail in Section 6. Before you can build or access a structure, you must use one of the structure commands.

The structure commands appear on the following menu (screen 5-8):

STRUCTURE COMMANDS menu

SCREEN 5-8



```
                          STRUCTURE COMMANDS
 1. ENTER STRUCTURE                    7. REMOVE STRUCTURE
 2. CHANGE STRUCTURE                   8. INSERT INTO STRUCTURE
 3. SAVE STRUCTURE                     9. PRINT STRUCTURE NAMES
 4. DISPLAY STRUCTURE                 10. HELP - EXPLANATION OF COMMANDS
 5. DISPLAY STRUCTURE NAMES           11. HELP - EXPLANATION OF STRUCTURES
 6. PRINT STRUCTURE                   12. TERMINATE STRUCTURE PROCESSOR

    SELECT COMMAND __                STRUCTURE NAME ________
```

Command selection

Key in the number of the command you want and the structure name and press the **XMIT** key.

We'll explain how to use structure commands in 5.15 through 5.23.

5.15. HOW TO CHANGE A STRUCTURE (CHANGE STRUCTURE COMMAND)

Purpose

Use this command to change a structure stored on your session file or the ESCORT library.

Action

When you specify this command, ESCORT:

1. Gets the structure from your session file or the ESCORT library

2. Lets you make your changes

3. Puts a copy of the changed structure into your session file

NOTE:

The changed structure exists only in your session file. To make the changes permanent, you must resave the structure under the same name.

STRUCTURE COMMANDS

When your structure exists on both the session file and the ESCORT library, the session file copy is the one that's changed when using this command.

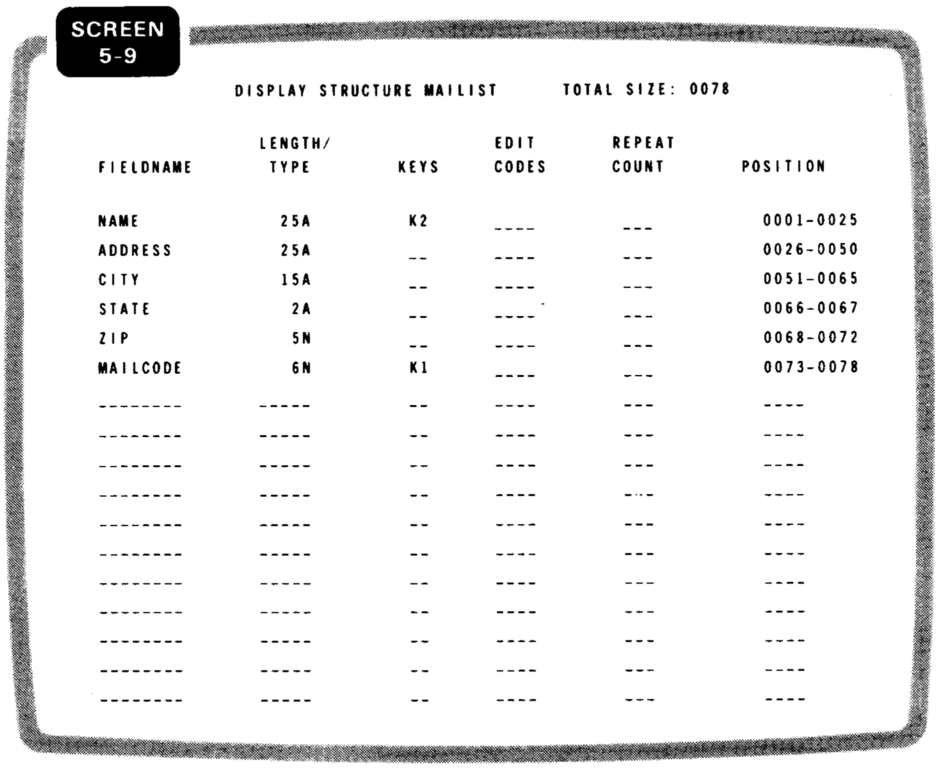
5.16. HOW TO DISPLAY A STRUCTURE ON THE WORKSTATION (DISPLAY STRUCTURE COMMAND)

Purpose You use this command to display a structure from your session file or the ESCORT library on the workstation.

Action When you specify this command, ESCORT:

1. Gets the structure from your session file or the ESCORT library
2. Displays the structure (screen 5-9):

SCREEN 5-9



```
            DISPLAY STRUCTURE MAILIST        TOTAL SIZE: 0078

              LENGTH/                EDIT      REPEAT
FIELDNAME     TYPE        KEYS       CODES     COUNT        POSITION

NAME          25A         K2         ----      ---          0001-0025
ADDRESS       25A         --         ----      ---          0026-0050
CITY          15A         --         ----      ---          0051-0065
STATE          2A         --         ----      ---          0066-0067
ZIP            5N         --         ----      ---          0068-0072
MAILCODE       6N         K1         ----      ---          0073-0078
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
--------      -----       --         ----      ---            ----
```

STRUCTURE
COMMANDS

5.17. HOW TO DISPLAY A LISTING OF STRUCTURE NAMES (DISPLAY STRUCTURE NAMES COMMAND)

Purpose

You use this command to display, on the workstation, a list of all structure names contained in both your session file and the ESCORT library.

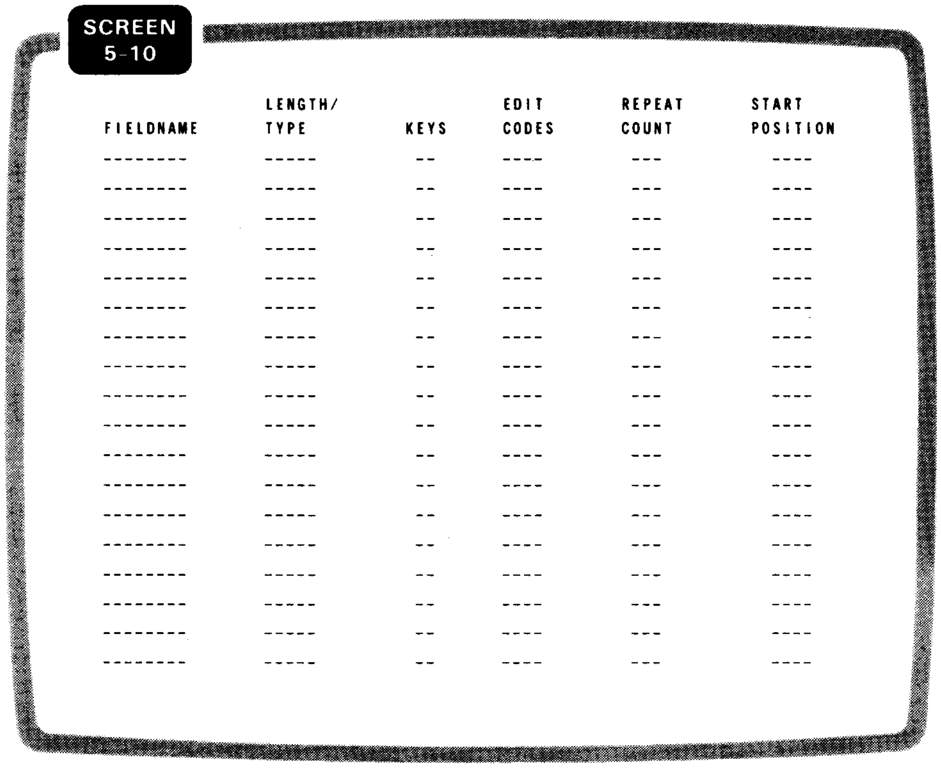
5.18. HOW TO BUILD A STRUCTURE (ENTER STRUCTURE COMMAND)

Purpose

You use this command to enter a structure from the workstation. When this command is executed, a screen appears (screen 5-10) that guides you in building your structure.

ENTER STRUCTURE screen

SCREEN 5-10



```
                LENGTH/              EDIT      REPEAT      START
FIELDNAME       TYPE        KEYS     CODES     COUNT       POSITION
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
--------        -----       --       ----      ---         ----
```

Action

The structure is saved in the current session file under the name you assign, and you can access it until you terminate your session. To permanently save a structure, use the SAVE STRUCTURE command.

5.19. HOW TO INSERT STATEMENTS INTO A STRUCTURE (INSERT INTO STRUCTURE COMMAND)

Purpose

You use this command to instruct ESCORT to insert new structure statements between existing statements of a structure.

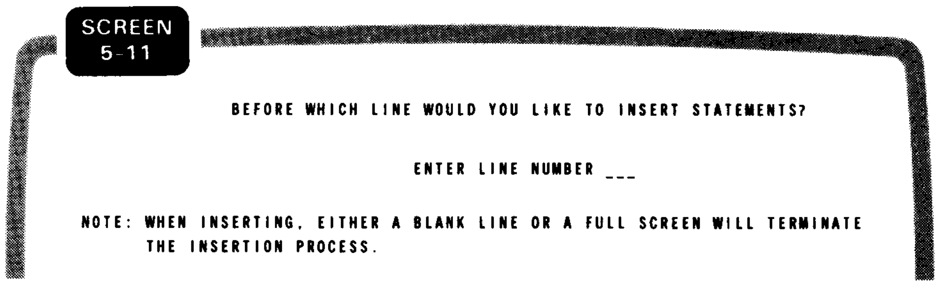
Action

When you specify this command, ESCORT:

1. Gets the structure from your session file or the ESCORT library.

2. Displays a screen (screen 5-11) on which you indicate where the new statements are to be inserted (before which line).

Identify statement line screen

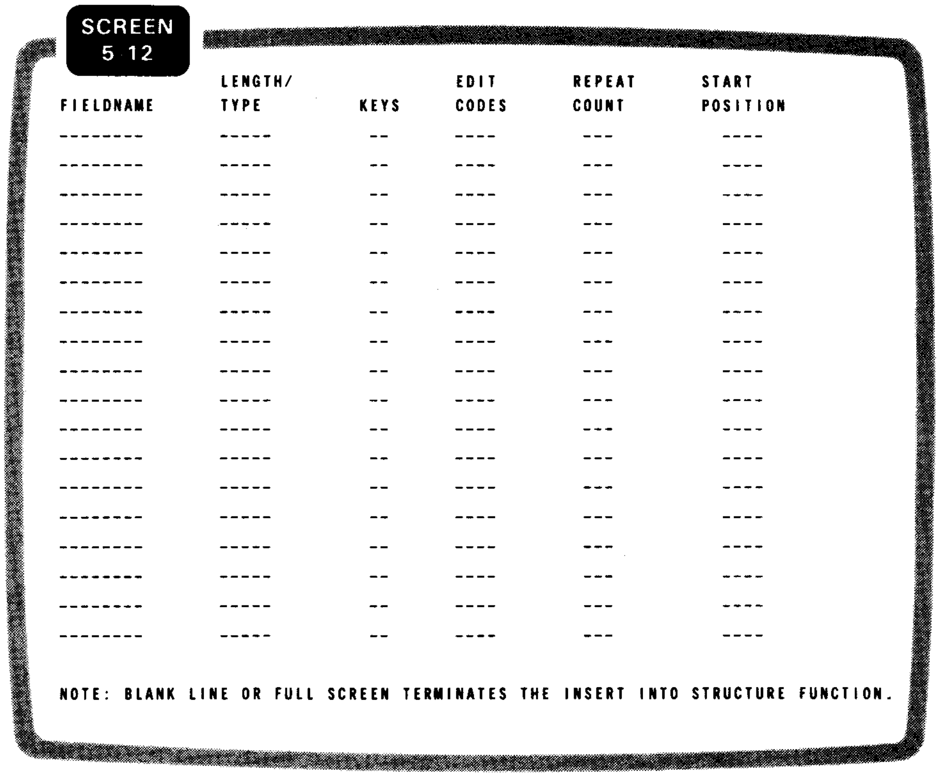


```
              BEFORE WHICH LINE WOULD YOU LIKE TO INSERT STATEMENTS?

                              ENTER LINE NUMBER ___

NOTE: WHEN INSERTING, EITHER A BLANK LINE OR A FULL SCREEN WILL TERMINATE
      THE INSERTION PROCESS.
```

3. Displays a screen (screen 5-12) on which you enter your new structure statements.

Insert into structure screen

SCREEN 5 12



```
                LENGTH/              EDIT      REPEAT     START
FIELDNAME       TYPE        KEYS     CODES     COUNT      POSITION
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----
--------        -----        --      ----       ---        ----

NOTE: BLANK LINE OR FULL SCREEN TERMINATES THE INSERT INTO STRUCTURE FUNCTION.
```

4. Puts a copy of the revised structure into your session file.

NOTES:

1. *The line number you enter cannot be zero or greater than the number of lines already in the structure.*

2. *You may insert up to a full screen of structure statements at one time.*

3. *The revised structure exists only in your session file. To make the insertions permanent, you must resave the structure under the same name.*

5.20. HOW TO PRINT A STRUCTURE (PRINT STRUCTURE COMMAND)

Purpose

You use this command to print a structure from your session file or the ESCORT library on the printer.

When you specify this command, ESCORT:

Action

1. Gets your structure from your session file or the ESCORT library
2. Prints your structure

5.21. HOW TO PRINT A LISTING OF STRUCTURE NAMES (PRINT STRUCTURE NAMES COMMAND)

Purpose

You use this command to print, on the printer, a list of all structure names contained in both your session file and the ESCORT library.

5.22. HOW TO REMOVE A STRUCTURE FROM THE ESCORT LIBRARY OR SESSION FILE (REMOVE STRUCTURE COMMAND)

Purpose

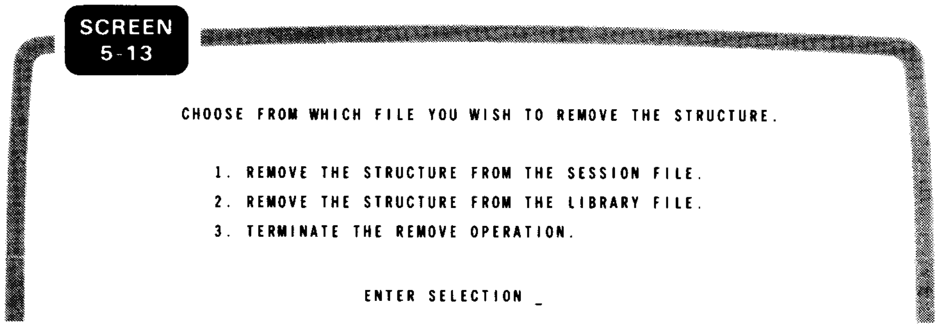
You use this command to instruct ESCORT to remove a structure from the ESCORT library or your session file.

Action

When you specify this command, ESCORT:

1. Displays a screen (screen 5-13) on which you identify the file from which the structure is to be removed.

Remove structure file selection screen



SCREEN 5-13

```
CHOOSE FROM WHICH FILE YOU WISH TO REMOVE THE STRUCTURE.

   1. REMOVE THE STRUCTURE FROM THE SESSION FILE.
   2. REMOVE THE STRUCTURE FROM THE LIBRARY FILE.
   3. TERMINATE THE REMOVE OPERATION.

                 ENTER SELECTION _
```

2. Removes the structure from the specified file.
3. Displays a message on the screen confirming that the structure has been removed from the file you specified.

STRUCTURE
COMMANDS

5.23. HOW TO SAVE A STRUCTURE (SAVE STRUCTURE COMMAND)

Purpose

You use this command to save a structure in the ESCORT library under a name you specify. Structures are placed in the ESCORT library under the type E$FD. For an explanation of library types, see 5.2.

Action

When you specify this command, ESCORT:

1. Displays a screen asking what you want to save
2. Gets a copy of your structure from your session file
3. Puts it into the ESCORT library under that name

NOTE:

Even though you saved your structure, a copy remains on your session file until you terminate your session.

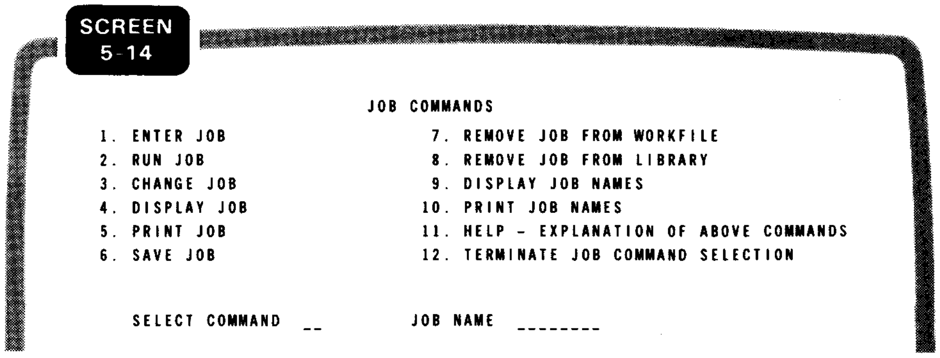
5.24. JOB COMMANDS

Job makeup

As you recall, a job is a group of related programs that are executed together. As with programs and structures, you can't enter or access a job without using one of the job commands.

The job commands appear on the following menu (screen 5-14):

JOB COMMANDS menu



```
                        JOB COMMANDS
 1. ENTER JOB                   7. REMOVE JOB FROM WORKFILE
 2. RUN JOB                     8. REMOVE JOB FROM LIBRARY
 3. CHANGE JOB                  9. DISPLAY JOB NAMES
 4. DISPLAY JOB                10. PRINT JOB NAMES
 5. PRINT JOB                  11. HELP - EXPLANATION OF ABOVE COMMANDS
 6. SAVE JOB                   12. TERMINATE JOB COMMAND SELECTION

     SELECT COMMAND __        JOB NAME ________
```

Command selection

Key in the number of the command you want and the job name and press the **XMIT** key.

We'll explain how to use job commands in 5.25 through 5.34.

5.25. HOW TO CHANGE A JOB (CHANGE JOB COMMAND)

Purpose

You use this command to change a list of program names associated with the RE$IDENT job or one stored on your session file or the ESCORT library.

Named jobs

When you name your job, ESCORT:

1. Gets that job from your session file or the ESCORT library

2. Lets you make your changes

3. Puts a copy of the changed job into your session file

NOTE:

The changes are made only on the session file copy of your job. To make the changes permanent, you must resave the job under the same name.

Unnamed jobs

When you don't name your job, ESCORT:

1. Gets the job named RE$IDENT

2. Lets you make your changes on the RE$IDENT job

5.26. HOW TO DISPLAY A JOB ON THE WORKSTATION (DISPLAY JOB COMMAND)

Purpose

You use this command to display, on the workstation, a job stored on your session file or the ESCORT library.

Named jobs

When you name the job you want to display, ESCORT:

1. Gets that job from your session file or the ESCORT library

2. Displays the job

Unnamed jobs

When you don't name the job you want to display, ESCORT displays the RE$IDENT job.

5.27. HOW TO DISPLAY A LISTING OF JOB NAMES (DISPLAY JOB NAMES COMMAND)

You use this command to display, on the workstation, a list of all job names contained in both the ESCORT library and your session file.

5.28. HOW TO ENTER A JOB (ENTER JOB COMMAND)

Purpose

You use this command to enter a list of up to eight program names, in order of execution, to form a job.

Named jobs

When you name your job, ESCORT stores it in the current session file under that name, and you can access it until you terminate your session.

Unnamed jobs

When you don't name your job, ESCORT assigns the default name RE$IDENT to it, and the job remains RE$IDENT for the duration of your session or until you enter another unnamed job that overwrites it.

JOB COMMANDS

5.29. HOW TO PRINT A JOB (PRINT JOB COMMAND)

Purpose

You use this command to print, on the printer, a list of program names associated with the job specified.

Named jobs

When you name the job you want to print, ESCORT:

1. Gets that job from your session file or the ESCORT library
2. Prints your job

Unnamed jobs

When you don't name the job you want to print, ESCORT prints the RE$IDENT job.

5.30. HOW TO PRINT A LISTING OF JOB NAMES (PRINT JOB NAMES COMMAND)

Purpose

You use this command to print, on the printer, a listing of all job names contained in both the ESCORT library and your session file.

5.31. HOW TO REMOVE A JOB FROM THE SESSION FILE (REMOVE JOB FROM WORKFILE COMMAND)

Purpose

You use this command to instruct ESCORT to remove the job you name from your session file.

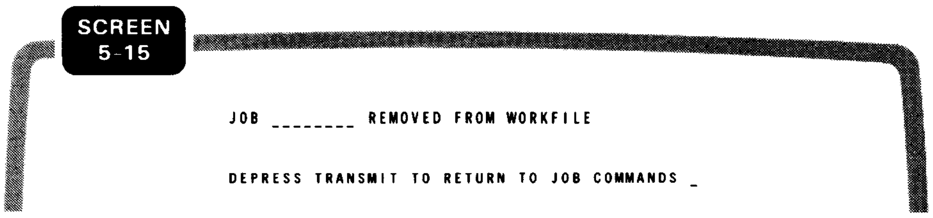
Action

When you specify this command, ESCORT:

1. Removes the job from the session file.
2. Displays a screen (screen 5-15) confirming that the job has been removed from the session file.

SCREEN 5-15

Job removed from workfile screen



```
JOB ________ REMOVED FROM WORKFILE

DEPRESS TRANSMIT TO RETURN TO JOB COMMANDS _
```

JOB COMMANDS

5.32. HOW TO REMOVE A JOB FROM THE ESCORT LIBRARY (REMOVE JOB FROM LIBRARY COMMAND)

Purpose

You use this command to instruct ESCORT to remove the job you name from the ESCORT library.

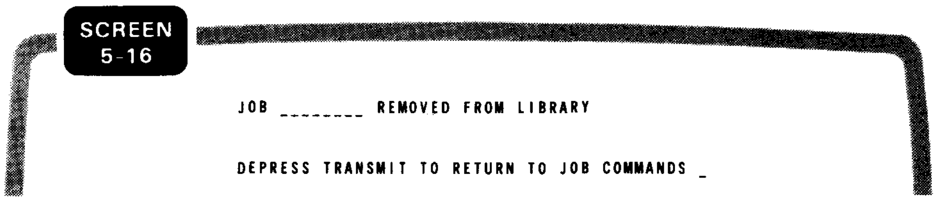
Named jobs

When you specify this command, ESCORT:

1. Removes the job from the ESCORT library.

2. Displays a screen (screen 5-16) confirming that the job has been removed from the ESCORT library.

Job removed from workfile screen


SCREEN 5-16

```
JOB ________ REMOVED FROM LIBRARY

DEPRESS TRANSMIT TO RETURN TO JOB COMMANDS _
```

5.33. HOW TO EXECUTE A JOB (RUN JOB COMMAND)

Purpose

You use this command to execute the RE$IDENT job or one stored on your session file or the ESCORT library.

Named jobs

When you name the job you want to execute, ESCORT:

1. Gets that job from your session file or the ESCORT library

2. Executes your job

Unnamed jobs

When you don't name the job you want to execute, ESCORT executes the RE$IDENT job.

5.34. HOW TO SAVE A JOB (SAVE JOB COMMAND)

Purpose

You use this command to save a job in the ESCORT library under a name that you specify. Jobs are placed in the ESCORT library under the type E$CJ. For an explanation of library types, see 5.2.

Named jobs

When you name the job you want to save, ESCORT:

1. Gets a copy of that job from your session file

2. Puts it in the ESCORT library under that name

NOTE:

Even though you've saved your program, a copy remains on your session file until you terminate your session.

Unnamed jobs

When you don't name the job you want to save, ESCORT assumes you want the RE$IDENT job, but you must rename it before you can save it because you can't save a job under the name RE$IDENT.

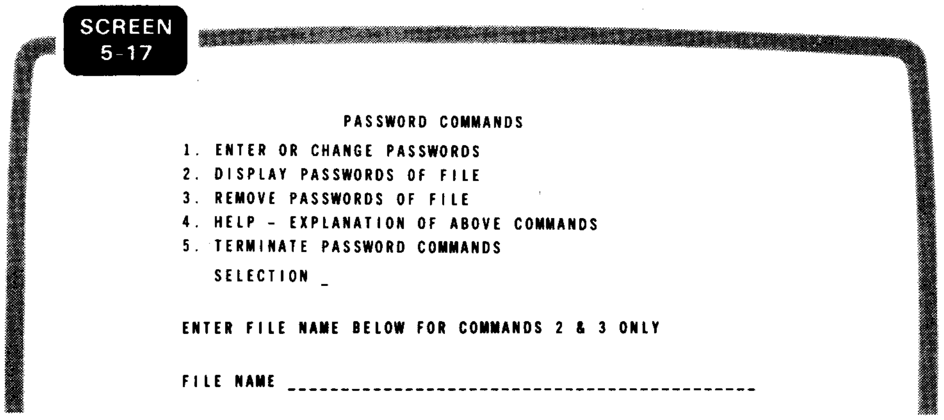
5.35. PASSWORD COMMANDS

Password requirements

Password commands are session oriented and allow you to specify during each ESCORT session any passwords required for the files you plan to use during the session. Passwords for a file must be made known to ESCORT before execution of any program that references that file. No password information is retained by ESCORT at the end of a session.

The password commands are shown on the following menu screen (screen 5-17).

Password commands menu



```
                    PASSWORD COMMANDS
1. ENTER OR CHANGE PASSWORDS
2. DISPLAY PASSWORDS OF FILE
3. REMOVE PASSWORDS OF FILE
4. HELP - EXPLANATION OF ABOVE COMMANDS
5. TERMINATE PASSWORD COMMANDS
   SELECTION _

ENTER FILE NAME BELOW FOR COMMANDS 2 & 3 ONLY

FILE NAME ____________________________________________
```

5.36. HOW TO DISPLAY THE PASSWORDS OF A FILE ON THE WORKSTATION (DISPLAY PASSWORDS OF FILE COMMAND)

Purpose

You use this command to display the previously entered passwords for a file on the workstation.

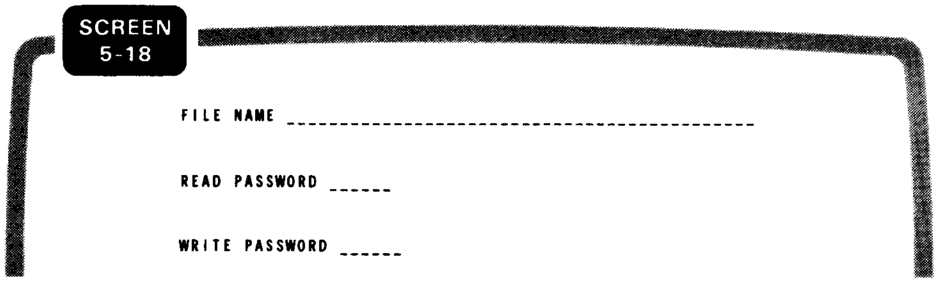
Action

When you specify this command, ESCORT:

1. Gets the passwords entered for the file during this session.

2. Displays the file name and passwords (screen 5-18).

Display passwords screen



```
FILE NAME ____________________________________________

READ PASSWORD ______

WRITE PASSWORD ______
```

JOB COMMANDS

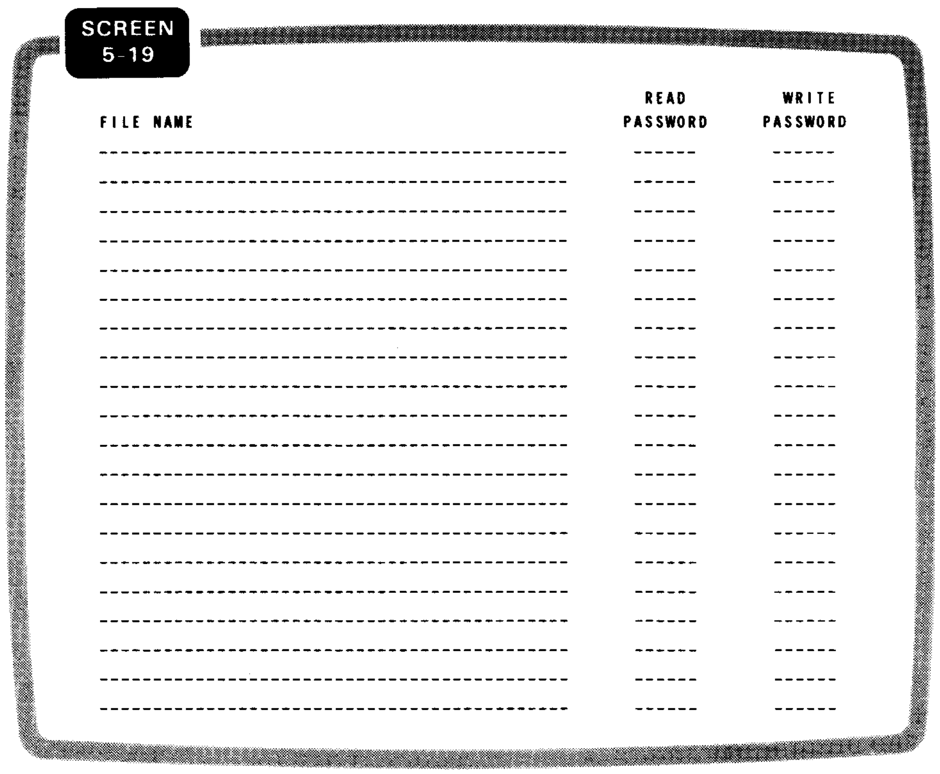
5.37. HOW TO ENTER OR CHANGE THE PASSWORDS OF A FILE (ENTER OR CHANGE PASSWORDS COMMAND)

Purpose

You use this command to enter or change file passwords from the workstation. When this command is executed, screen 5-19 appears.

Enter/change passwords screen

SCREEN 5-19



```
                                                      READ       WRITE
FILE NAME                                           PASSWORD   PASSWORD
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
--------------------------------------------         ------     ------
```

Enter file names and passwords as needed for your use during the current session.

To change or correct passwords previously entered, simply reenter the file name and supply the revised passwords. ESCORT automatically replaces the passwords when a duplicate file name is entered.

5.38. HOW TO REMOVE THE PASSWORDS OF A FILE (REMOVE PASSWORDS OF FILE COMMAND)

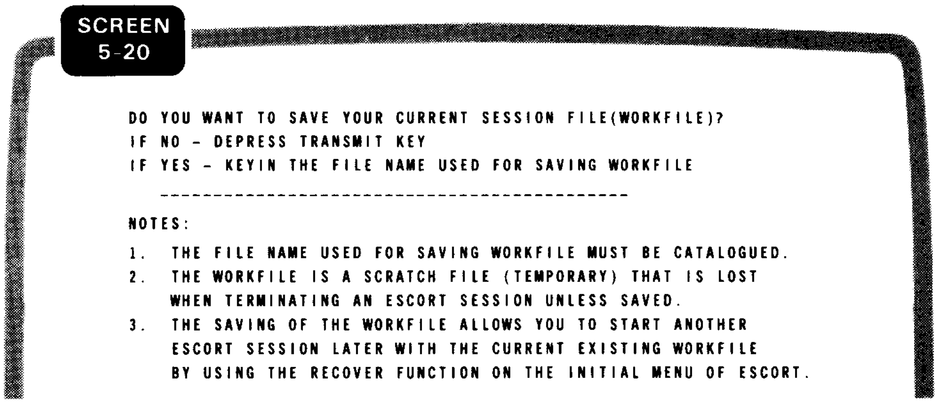
You use this command to remove the file name you specify and its associated passwords from the ESCORT session.

JOB COMMANDS

5.39. HOW TO TERMINATE AN ESCORT SESSION (TERMINATE CURRENT ESCORT SESSION COMMAND)

When you are ready to end your ESCORT session, choose selection 1, TERMINATE CURRENT ESCORT SESSION, from the COMMAND SELECTION menu (screen 5-1). The following screen (screen 5-20) appears, asking if you want to save your session file.

SCREEN 5-20



```
DO YOU WANT TO SAVE YOUR CURRENT SESSION FILE(WORKFILE)?
IF NO - DEPRESS TRANSMIT KEY
IF YES - KEYIN THE FILE NAME USED FOR SAVING WORKFILE
   --------------------------------------------
NOTES:
1.  THE FILE NAME USED FOR SAVING WORKFILE MUST BE CATALOGUED.
2.  THE WORKFILE IS A SCRATCH FILE (TEMPORARY) THAT IS LOST
    WHEN TERMINATING AN ESCORT SESSION UNLESS SAVED.
3.  THE SAVING OF THE WORKFILE ALLOWS YOU TO START ANOTHER
    ESCORT SESSION LATER WITH THE CURRENT EXISTING WORKFILE
    BY USING THE RECOVER FUNCTION ON THE INITIAL MENU OF ESCORT.
```

Saving your session file allows you to recover its contents when you initiate another ESCORT session. The RECOVER function is described in 2.3.

If you want to save your session file, enter the name of a file on which its contents are to be saved. Remember that the file you name must be cataloged.

If the file you name is password protected, the following message appears at the bottom of the screen:

```
ESC117   RECOVERY/SAVE REQUIRES PASSWORD
         ENTER READ OR WRITE PASSWORD__________
```

Enter either password for the file. When the other password is required, it is requested.

6. ESCORT Files and Structures

6.1. UNDERSTANDING FILES

This section deals with ESCORT file types and creating a file structure. Before we discuss file types, however, familiarize yourself with the following terms:

Field defined

- Field

  Kinds of fields

  The smallest unit of an ESCORT file. Each field holds an individual piece of data. There are two kinds of fields: system utility fields (6.9), which the system supplies, and variable data fields, which you supply.

Record defined

- Record

  A group of fields relating to a specific item. For example, a bank account record for each customer.

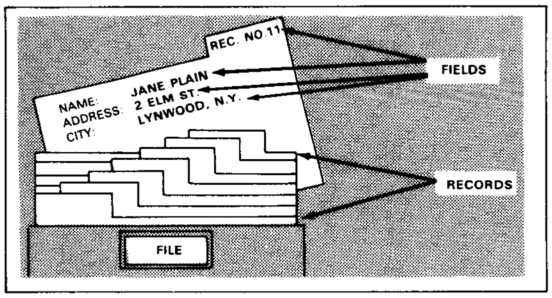
File defined

- File

  A group of related records taken as a whole. For example, all the bank records contained in a master file.

Example

Perhaps a simpler way of understanding files is to picture a filing cabinet of employee personnel folders. Each folder is a record for a specific employee. Each record is broken down into fields (name, address, etc), and all the records contained in the filing cabinet make up a file.



Now that you understand ESCORT file terminology, let's look at the two different types of ESCORT files available.

6.2. TYPES OF FILES

ESCORT data file types

ESCORT creates and uses exclusively MIRAM (multiple indexed random access method) data files and can also access MIRAM data files created by another processor, such as COBOL. Further information on the use of files created by another processor is given in Appendix D.

MIRAM file types

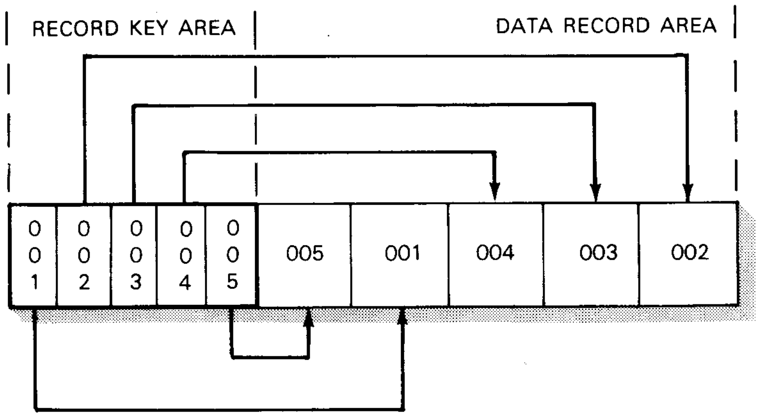
There are two types of MIRAM data files: indexed and nonindexed.

Indexed Files

An indexed file contains data records and an index of record keys. Each record key points to a certain record in your file.

Organization

Data records appear in a file in the order you enter them, and the record keys are arranged in ascending order.



Record keys point to data records.

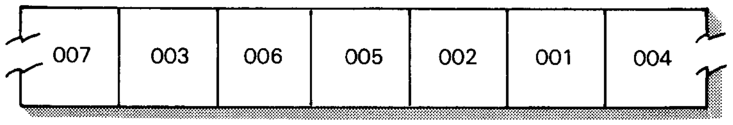
You can process indexed files in two ways:

- Randomly, by specifying the key of the record
- Sequentially, by beginning with the lowest key value

Nonindexed Files

Organization

A nonindexed file is organized sequentially. This means the records appear in a file in the order you enter them, and they are processed in this order.



Nonindexed files are organized sequentially and are processed in that order.

For more information on MIRAM files and access methods, see the consolidated data management concepts and facilities manual.

Now that you know the kinds of files that are available, let's see how you allocate them.

6.3. ALLOCATING FILE SPACE

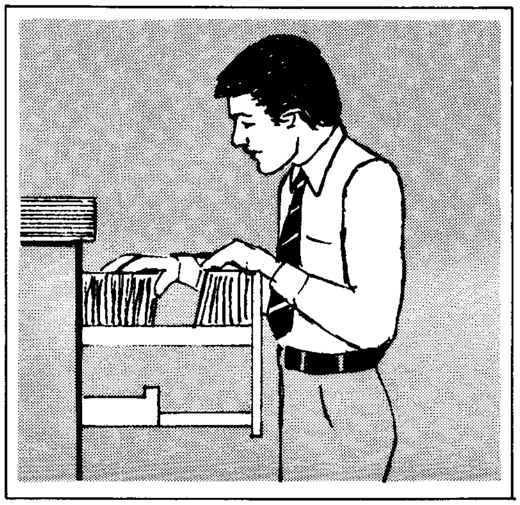
Allocating files for ESCORT

Before you can create files in ESCORT, you must first catalog the files you need. When you don't have any space allocated in storage, begin in interactive services with the ALLOCATE command. For example:

Sample interactive services ALLOCATE command

```
AL ST,FIL=MYJOB,VSN=MYPAC,SI=1
```

AL is the abbreviation for ALLOCATE. ST stands for SAT file. FIL names the file where you plan to store your ESCORT programs. VSN names the volume where your files are stored. SI indicates the size of the area you're allocating. The interactive commands and facilities user guide/programmer reference has more information on the ALLOCATE command. You can also allocate space with OS/3 job control statements.



Calling general editor (EDT)

When your ALLOCATE command is accepted, call the general editor by keying in EDT. EDT prompts you with a line number and an SOE. You're now ready to key in the job control stream that catalogs your files.

Job stream for System 80

If you're using a System 80, the following job stream is best for you. You should catalog the stream itself so you can reuse it. The job stream prompts you for the three pieces of information listed in the QGBL statement. When you key in the DVC, the VSN, and the file name, your file is cataloged.

Sample job control stream

```
1.0000 // JOB ESC.CAT
2.0000 // JNOTE 'DVC = 170 FOR FIXED DISK, 60 FOR REMOVABLE'
3.0000 // QGBL DVC,VSN,FILENM
4.0000 // DVC &DVC        // VOL &VSN
5.0000 // LBL &FILENM     // LFD TEST
6.0000 // CAT TEST
7.0000 /&
```

Explanation

The file name in the CAT statement must be the same as the file name in the LFD statement. Terms preceded by the ampersand (&) are variables that you should fill in when you first key in the job stream.

Executing stream from ESCORT

When you're already in ESCORT and you want to execute the job stream, you can issue an RV command without leaving ESCORT. See the job control user guide for more information on job control statements. See also the file cataloging concepts and facilities manual for further information on the file cataloging facilities.

Alternate job stream

If you're using a system other than System 80, use this job stream, which doesn't prompt you for input. You can easily change the stream when you want to catalog additional ESCORT files. To execute this stream, key in an RV command.

Example

```
1.0000 // JOB MYCAT
2.0000 // DVC 50        // VOL MYPAC
3.0000 // LBL ESCORT.MY
4.0000 // LFD MYFILE
5.0000 // CAT MYFILE
6.0000 /&
7.0000 @WRITE MYCAT,MYESC,MYPAC
7.0000 @D
1.0000 // JOB MY
2.0000 // LBL ESCORT.MY
3.0000 // EXEC PROG1
4.0000 /&
5.0000 @WRITE MY,MYESC,MYPAC
```

Explanation of statements

The first @WRITE statement writes the job MYCAT to the file MYESC. The @D statement returns the line number to line 1. The second @WRITE statement writes the job MY to the file MYESC. One RV statement executes both job streams. When you want to add new files to your catalog, call the general editor (EDT) by keying in EDT, and change only those statements that contain file names you want altered. See the job control user guide for more information on job control statements.

Changing stream to add files

ESCORT library

ESCORT requires a MIRAM library named ESC$ESCORT.LIBRARY.FILES for storage of all structures, programs, and jobs. Presently, ESCORT automatically allocates a 5-cylinder library on the RES disk the first time ESCORT is entered.

Creating the file

Once your file name is in the catalog, all you have to do is create your file in tutorial mode or in program mode by using the ENTER DATA program statement and the CREATING clause. ESCORT rules for file names are given in detail in 7.20.

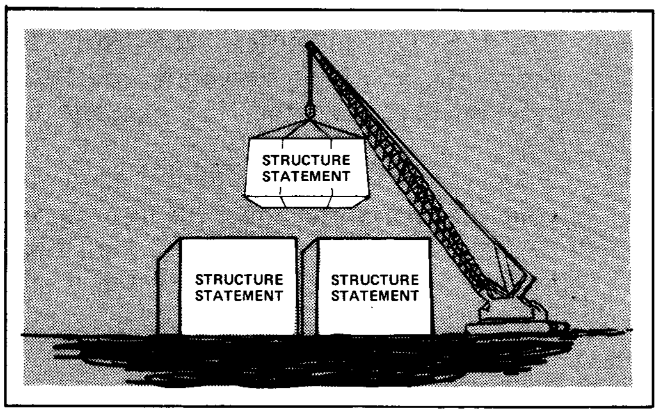
When ESCORT interprets the CREATING clause, it goes to the file catalog and finds your file name. It then goes to your structure and determines the key (sort keys) you designated. Then ESCORT allocates space for your file.

6.4. HOW TO BUILD A FILE STRUCTURE

Now that you're familiar with the types of ESCORT files and how to allocate space for them, let's look at file structures.

Structure defined

As you recall from Section 1, you use structures to tell ESCORT what your data records look like. You use one or more structure statements to build a structure.



Entering a structure

Structure statement makeup

You build (enter) a structure by using a screen menu provided by ESCORT. But before you do this, you should familiarize yourself with structure statement makeup. That is, you should know about the required and optional parts of a structure statement.

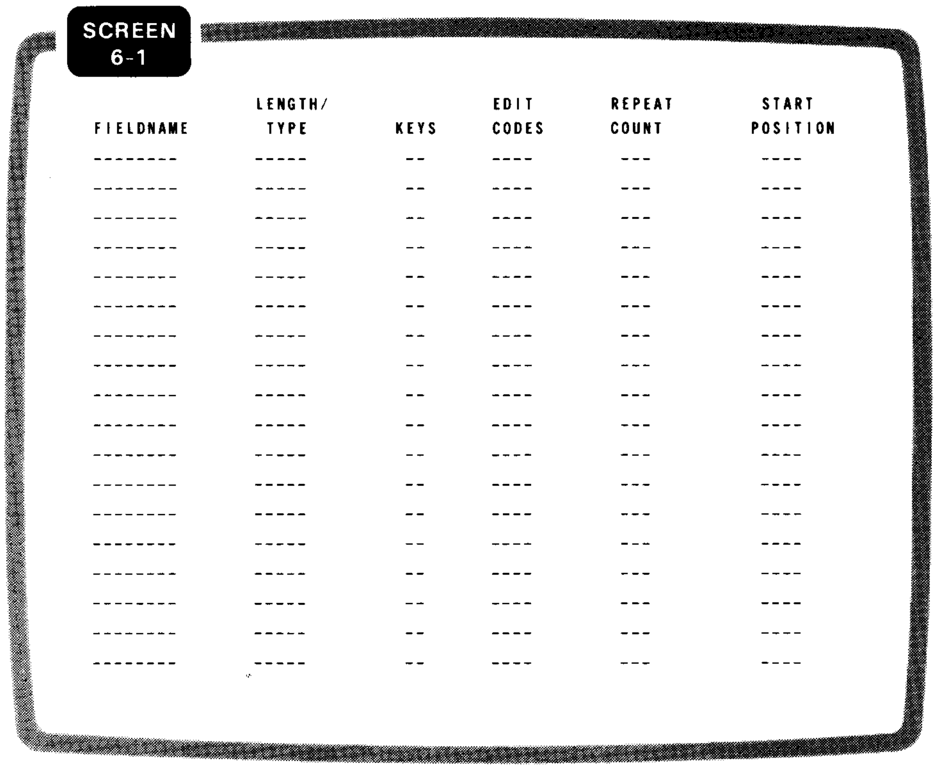
We'll tell you about these things and then explain how to enter and change a structure. Then, we'll tell how to manipulate your structure by using other structure commands.

6.5. STRUCTURE STATEMENT MAKEUP

Function of structure statements

Structure statements specifically define individual fields within your file by telling ESCORT exactly what those fields look like. You enter structure statements on this screen (screen 6-1):

SCREEN 6-1

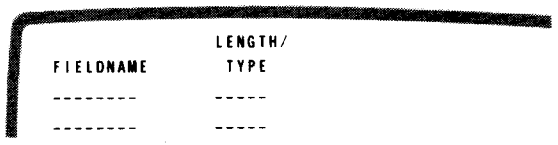


```
              LENGTH/           EDIT    REPEAT    START
 FIELDNAME     TYPE     KEYS    CODES   COUNT    POSITION
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
 --------     -----      --     ----     ---       ----
```

Required parts of a structure statement

As you can see, there are six parts to a structure statement, but only two are required:

- FIELDNAME
- LENGTH/TYPE

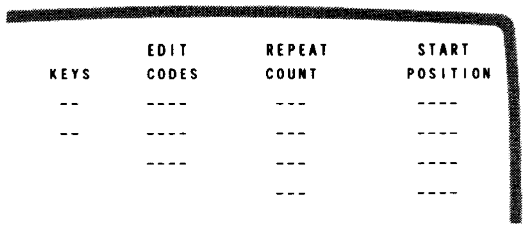


```
              LENGTH/
 FIELDNAME     TYPE
 --------     -----
 --------     -----
```

Optional parts of a structure statement

The other four parts are optional, and each performs a different function:

- KEYS
- EDIT CODES
- REPEAT COUNT
- START POSITION



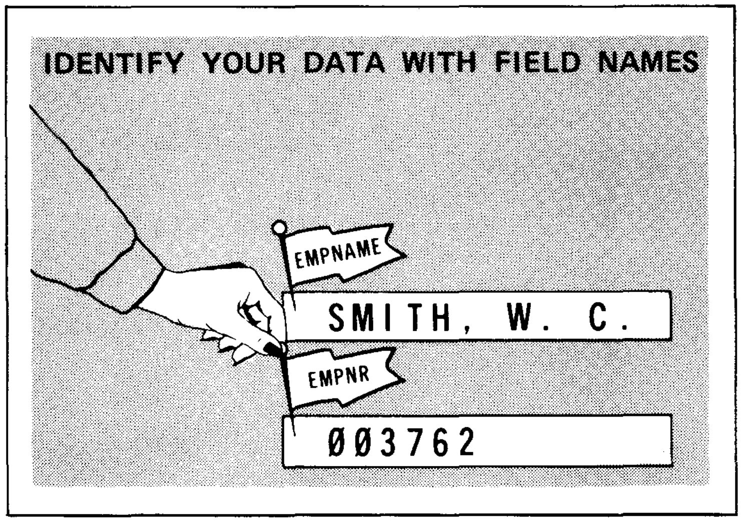
```
         EDIT    REPEAT     START
 KEYS    CODES   COUNT     POSITION
  --     ----     ---        ----
  --     ----     ---        ----
         ----     ---        ----
                  ---        ----
```

6.6. REQUIRED PARTS OF A STRUCTURE STATEMENT

FIELDNAME

Identifying the data

To identify each piece of data you put into a file, you must assign field names.



Format

Field names (also called variable data names) consist of one to eight alphanumeric characters of which the first must be alphabetic. We'll give some additional rules on field names in 6.8.

Restrictions

You can't use ESCORT reserved words (Appendix B) or any punctuation or special symbols in field names.

Examples

The following examples show correct and incorrect use of field names.

Correct	Incorrect	Why Incorrect
ACCTNO1	1ACCTNO	First character numeric
DATAONE	DATA	ESCORT reserved word
TRAN001	TRAN.001	Punctuation not allowed
KAPHIL	KA*PHIL	Symbols not allowed

LENGTH/TYPE

Specifying data length and type

You must specify how much and what kind of data you want to include in each field:

Type	Maximum Length
Alphanumeric	4092 characters
Numeric	15 digits
Assumed decimal	15 digits
Packed decimal	15 digits
Binary	9 digits

LENGTH designation

LENGTH is a numeric designation of how many characters are in the field. For example, a 5-number zip code has a LENGTH of 5.

TYPE designation

TYPE specifies the kind of data. The five types of data you can place in your fields are:

- Alphanumeric

Alphanumeric fields

Alphanumeric fields contain letters, numbers, and special symbols, including punctuation.

Length

The maximum length for an alphanumeric field is 4092 characters, and each character occupies one byte of storage.

Structure representation

Alphanumeric fields are represented in your structure by the letter A in the LENGTH/TYPE column of your ENTER STRUCTURE screen (screen 6-4).

Examples

Examples of alphanumeric fields are:

Field Name	Contains	Is Written
NAME	12 letters	12A
ADDRESS	25 letters and numbers	25A
CITY	30 letters and punctuation	30A

- Numeric

Numeric fields

Numeric fields contain only integers, or whole numbers. Any other designation results in an error.

Length

The maximum length for a numeric field is 15 digits.

Sign handling

Each digit occupies one byte in storage, and the sign is handled as an overpunch in the last digit, so you need not leave an extra space for a sign when entering negative numbers.

Because of the way ESCORT handles numeric data, when you include a sign in numeric fields you must designate this in the LENGTH/TYPE column by placing an S before the length designation. For example, a 5-digit numeric field that will contain negative values is written: S5N. You must include the "–" edit code in the statement specification.

Structure representation

Numeric fields are represented in your structure by the letter N in the LENGTH/TYPE column of your ENTER STRUCTURE screen (screen 6-4).

Examples

Examples of numeric fields are:

Field Name	Contains	Is Written
ZIPCODE	5 integers	5N
ACCTNO	7 integers	7N
INVNO	10 integers	10N

- Assumed Decimal

Assumed decimal fields

Assumed decimal fields contain integer and decimal digits.

Length

The length of assumed decimal fields is broken into numbers appearing before the decimal point and numbers appearing after the decimal point. The combined length of an assumed decimal field may not exceed 15 digits. The decimal portion may not exceed nine digits.

Sign handling

Each digit occupies one byte in storage; however, the sign is handled as an overpunch in the last digit, so you need not leave an extra space for a sign when entering negative numbers.

Because of the way ESCORT handles decimal data, when you include a sign in assumed decimal fields you must designate this in the LENGTH/TYPE column by placing an S before the length designation. For example, a 5-digit, 2-decimal field that will contain negative values is written: S5V2. You must include the "–" edit code in the statement specification.

Structure representation

Assumed decimal fields are represented in your structure by the letter V in the LENGTH/TYPE column of your ENTER STRUCTURE screen (screen 6-4).

Examples

Examples of assumed decimal fields are:

Field Name	Contains	Is Written
ACCTBAL	6 integers, 2 decimals	6V2
TOTFUNDS	13 integers, 2 decimals	13V2
INFLRATE	3 integers, 5 decimals	3V5

- Packed Decimal

Packed decimal fields

Packed decimal fields contain integer and decimal numbers that are compressed or expanded internally for calculation, display, and storage purposes. When you view these fields from a workstation display, they appear as standard numeric fields.

Length

The length of packed decimal fields is broken into numbers appearing before the decimal point and numbers appearing after the decimal point. The combined length of a packed decimal field may not exceed 15 digits. The decimal portion may not exceed nine digits.

Storage requirements

Each two digits of a packed decimal occupy one byte of storage except for the least significant digit (the digit farthest to the right) and the sign, which occupy one byte of storage together.

Sign handling

When the value in the field may be negative, you must include the "–" edit code in the statement specification.

Structure representation

Packed decimal fields are represented in your structure by the letter P in the LENGTH/TYPE column of your ENTER STRUCTURE screen (screen 6-4).

Examples

Examples of packed decimal fields are:

Field Name	Contains	Is Written
PACKONE	5 integers, 2 decimals	5P2
KAPHIL	10 integers, 5 decimals	10P5
DAVIDFIL	3 integers, 2 decimals	3P2

- Binary

Binary fields

Binary fields consist of integers, or whole numbers, and are stored in binary notation.

Length

The maximum length for a binary field is nine digits.

Storage requirements

Depending on the length of your field, your data takes up the following bytes of storage:

Field Length	Bytes of Storage
1 to 4	2
5 to 9	4

Negative numbers

When you plan to use a binary field to represent negative numbers, you must designate the negative sign as an edit code in the EDIT CODES column of your ENTER STRUCTURE screen.

Sign handling

When the value in the field may be negative, you must include the "–" edit code in the statement specification.

Structure representation

Binary fields are represented in your structure by the letter B in the LENGTH/TYPE column of your ENTER STRUCTURE screen (screen 6-4).

Examples

Examples of binary fields are:

Field Name	Contains	Is Written
BIFILE	5 integers	5B
GREENFIL	9 integers	9B
WORKORD	2 integers	2B

6.7. OPTIONAL PARTS OF A STRUCTURE STATEMENT

KEYS

Use of sort keys

Sort keys are fields in your structure you use to sort your data in a specific order.

Indexed files

When you have an indexed file, you designate index key fields as your sort keys. When you have nonindexed files, you use matching fields.

Nonindexed files

The KEYS portion of your ENTER STRUCTURE screen allows you to designate a field as an index key field or a matching field.

Matching criteria entry

The entry you make is called the matching criteria.

Use of matching criteria

When you need data from more than one file or device, you use matching criteria to define the file to the system so it can determine the input/output instructions it needs to generate and use in the program.

Types of matching criteria

There are three types of matching criteria:

- Index key
- Matching sorted
- Matching unsorted

Choosing a type

Because your files can be indexed or nonindexed (6.2), the type of criteria you use depends on the type of file you choose.

- Indexed Files Use Index Key Fields

Index key fields

Uses

Index key fields are the matching criteria for indexed files. Designate up to five index key fields per structure by entering K1 through K5 in the KEYS portion of your ENTER STRUCTURE screen. Use index keys only with indexed files; you can't use them in a structure that has matching key fields.

Multiple index keys

When you create a file with multiple keys, you can access the file randomly by any one of the keys. You indicate which key to use by creating alternate structures for your file.

Example

In the following example, the structure NAMEADDR has three keyed fields.

STRUCTURE NAMEADDR

FIELDNAME	LENGTH/ TYPE	KEYS
CUSTNO	5N	K1
NAME	15A	K2
ADDR	15A	
CITY	12A	
STATE	2A	K3
ZIP	5N	

CUSTNO (customer number) is the primary key. NAME and STATE are secondary keys. When you execute this program:

```
ENTER DATA FROM WS CREATING NAMEADDR
```

the file NAMEADDR is created with three keys. The file may later be accessed by CUSTNO, NAME, or STATE. Assume the following data is entered:

```
CUSTNO   NAME           ADDR            CITY           STATE ZIP

10325    SMITH JOHN     111 OAK LANE    PHILADELPHIA   PA    19024
59999    JONES MARY     P O BOX A       TRENTON        NJ    20355
35411    DOE JANE       1520 ESSARY     KNOXVILLE      TN    37918
72760    THOMAS JAMES   78 HILL AVENUE  LANCASTER      PA    19455
```

Accessing by primary key

You can use the structure NAMEADDR to access the file by the primary key, CUSTNO. When the structure applied to your file has more than one key designated, the primary key (K1) is used.

Output arranged by primary key

When you use this SELECT DATA program, the output is organized in CUSTNO sequence because CUSTNO is the field designated K1:

```
SELECT DATA OF NAMEADDR PRINT
```

The output of this program is:

```
CUSTNO    NAME          ADDR             CITY           STATE ZIP

10325     SMITH JOHN    111 OAK LANE     PHILADELPHIA   PA    19024
35411     DOE JANE      1520 ESSARY      KNOXVILLE      TN    37918
59999     JONES MARY    P O BOX A        TRENTON        NJ    20355
72760     THOMAS JAMES  78 HILL AVENUE   LANCASTER      PA    19455
```

An inquiry program, such as,

```
SELECT DATA OF NAMEADDR FROM WS DISPLAY
```

returns a workstation prompt for the field CUSTNO. When you enter a value, the record whose CUSTNO value matches the value entered is displayed on the workstation screen.

Accessing by secondary keys

To access the file by a secondary key, you must create an alternate structure. To output a listing of the file NAMEADDR in NAME or alphabetic order, you need to create the following alternate structure, NAMEADD2.

Alternate structure for K2

STRUCTURE NAMEADD2

FIELDNAME	LENGTH/ TYPE	KEYS
CUSTNO	5N	
NAME	15A	K2
ADDR	15A	
CITY	12A	
STATE	2A	
ZIP	5N	

Then the program

```
SELECT DATA OF NAMEADDR (NAMEADD2) PRINT
```

Output arranged by secondary key

produces the following listing:

```
CUSTNO    NAME          ADDR             CITY           STATE ZIP

35411     DOE JANE      1520 ESSARY      KNOXVILLE      TN    37918
59999     JONES MARY    P O BOX A        TRENTON        NJ    20355
10325     SMITH JOHN    111 OAK LANE     PHILADELPHIA   PA    19024
72760     THOMAS JAMES  78 HILL AVENUE   LANCASTER      PA    19455
```

SELECT DATA program with alternate structure

NOTE:

In the preceding program, the structure name is not the same as the file name. Therefore, it is necessary to include the structure name in parentheses. When the structure name is the same as the file name, you don't need to list any structure name at all.

The inquiry program

```
SELECT DATA OF NAMEADDR (NAMEADD2) FROM WS DISPLAY
```

returns a workstation prompt for the field NAME.

The following structure allows access to the file NAMEADDR by the third key, STATE.

Alternate structure for K3

STRUCTURE NAMEADD3

FIELDNAME	LENGTH/ TYPE	KEYS
CUSTNO	5N	
NAME	15A	
ADDR	15A	
CITY	12A	
STATE	2A	K3
ZIP	5N	

The program

```
SELECT DATA OF NAMEADDR (NAMEADD3) PRINT
```

Output arranged by K3

produces the listing in order by STATE:

```
CUSTNO    NAME           ADDR           CITY          STATE ZIP

59999     JONES MARY     P O BOX A      TRENTON       NJ    20355
10325     SMITH JOHN     111 OAK LANE   PHILADELPHIA  PA    19024
72760     THOMAS JAMES   78 HILL AVENUE LANCASTER     PA    19455
35411     DOE JANE       1520 ESSARY    KNOXVILLE     TN    37918
```

The customer number and name fields contain unique values. The state is not unique; it contains duplicates. When key matching from an input file or the workstation is present, ESCORT can't access all records having duplicate values. Therefore, the program

```
SELECT DATA OF NAMEADDR (NAMEADD3) FROM WS DISPLAY
```

SELECT DATA program with duplicate keyed fields

presents a workstation prompt for the field STATE. But when a value, for example PA, is entered, only the first record found with STATE equaling PA is displayed. You can select all records with duplicate fields by not using key criteria. This program shows one way to do this:

```
SELECT DATA OF NAMEADDR
IF STATE = 'PA' DISPLAY END
```

This program has to be changed and recompiled if the selection changes.

Another technique accomplishes this task and eliminates the need for changing the program. You need an additional file, here named SEARCH, with the following structure:

STRUCTURE SEARCH

FIELDNAME	LENGTH/ TYPE	KEYS
STATE	2A	K1

In the following program, enter a value for STATE, such as PA, when prompted. When the prompt appears again, press the **FUNCTION** and **F4** keys to terminate input to the program.

```
ENTER DATA FROM WS CREATING SEARCH
```

Then execute this second program.

```
SELECT DATA OF NAMEADDR
USING SEARCH
IF STATE = STATE OF SEARCH DISPLAY END
```

All records where STATE equals PA are displayed. To get all records containing NJ, rerun the two programs, entering NJ instead of PA. You can place the programs in an ESCORT job so that a single job is run instead of two programs. You can use this technique on any fields that contain duplicate values.

- Nonindexed Files Use Matching Fields

Matching fields

Types of matching fields

Nonindexed files have no index key fields, but use a specific data field as the matching criteria. In nonindexed files, records are stored sequentially and may or may not be sorted; therefore, there are two kinds of matching criteria you can enter in the KEYS portion of your ENTER STRUCTURE screen: matching sorted and matching unsorted.

- Matching Sorted (MS)

Matching sorted

Tells the system the matching data field is contained in an unkeyed file that is sorted in ascending order according to that data field.

Structure representation

To designate matching sorted in the KEYS portion of your structure, key in MS.

- Matching Unsorted (MU)

Matching unsorted

Tells the system that the matching data field is contained in an unkeyed file and the data records are not in order.

Structure representation

To designate matching unsorted in the KEYS portion of your structure, key in MU.

Restrictions

You can't use matching for a sorted (MS) and an unsorted (MU) file in the same structure or in an indexed file structure.

- Applying Matching Criteria to a Device

Matching criteria with a device

You can also use matching criteria when you apply a structure to a device, such as a workstation.

For example, when you use the workstation as an input device, the matching criteria field supplies the prompt field for selection of a record from a disk file.

EDIT CODES

Function

Symbols and functions

The EDIT CODES column of your ENTER STRUCTURE screen lets you specify symbolic editing codes that perform certain functions on numeric fields when you output them to the printer or workstation. Table 6-1 lists the edit code symbols and their functions.

Table 6-1. Edit Codes

Symbol	Function
Z	Suppress leading zeros
*	Replace leading zeros with asterisks
,	Insert commas where applicable
–	Insert sign after least significant digit If the number is: Positive – the sign inserted is a blank character Negative – the sign inserted is a minus symbol
/	Insert slash character where applicable in date field In 4-digit date field, one slash is inserted. In 6-character date field, two slashes are inserted.
*$	Insert fixed dollar sign Fixed dollar sign is assumed when the asterisk is in the first position of the field.
$	Insert floating dollar sign Floating is assumed when the asterisk is omitted.

NOTE:

When using a form to output your data, the slash, sign, comma, and decimal point codes are characters that are added to the field and should be allowed for in your output. For example, to output a 5V2 field with a comma, you must allow for nine positions: eight for the number and decimal point and one for the comma.

REPEAT COUNT

Function

You use the REPEAT COUNT column of your ENTER STRUCTURE screen to specify the number of times an array field is repeated contiguously in the record.

Array fields defined

Array fields are fields of similar data that occur several times and have the same data type and length for each occurrence. Instead of specifying fields individually, you specify them as one array field.

Example of array field

Let's assume you want an array field named MONTHS that lists a 3-character abbreviation for each month.

An example of the array MONTHS looks like this:

1	2	3	4	5	6	7	8	9	1Ø	11	12
JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC

Specifying array fields in a structure

To specify this field in your structure as an array field, you fill in the following columns on your ENTER STRUCTURE screen:

FIELDNAME	LENGTH/TYPE	REPEAT COUNT
MONTHS	3A	12

This means your field named MONTHS is 3 alphanumeric characters long and occurs 12 times.

Referencing array fields in a program

To reference a certain month of the array in a program, specify the field name and the occurrence number you want, and ESCORT gets it for you. For example, when you want September, you specify MONTHS (9) in your program. This tells ESCORT to reference the ninth item in the array named MONTHS.

START POSITION

Use

The optional START POSITION column of your ENTER STRUCTURE screen lets you designate a specific record position for a data field to begin. When you don't use this column, ESCORT automatically assigns positions, beginning with position 1.

START POSITION is especially helpful when you want to define more than one field name to occupy the same record position (overlay) because it lets you reference a specific part of a field. For example, assume you want to specify a field named DATE. Within this field, you have the month, day, and year, which you may want to reference individually at some point. Fill in the following portions of your ENTER STRUCTURE screen:

FIELDNAME	LENGTH/TYPE	START POSITION	
DATE	6N	1	
MONTH	2N	1	*
DAY	2N	3	*
YEAR	2N	5	*

Since MONTH, DAY, and YEAR occupy the same record positions as the field DATE, you can reference individual parts of the field by specifying any one of them in your program.

For example, when you want only the month portion of the DATE field printed, you specify MONTH, and you only get the month. When you want the entire date, just specify DATE.

Overlay field

When you designate an overlay field in a structure, ESCORT places an asterisk after the START POSITION column of the structure.

Placement of asterisks

Where the asterisk is placed depends on the order in which you specify the fields. In the preceding example, the asterisks appear after the MONTH, DAY, and YEAR fields because the field DATE precedes them in the structure. If the fields MONTH, DAY, and YEAR were entered in the structure before the field DATE, the asterisk would be on the DATE field.

6.8. VARIABLE NAME QUALIFIERS QUALIFY FIELD NAMES

Purpose

There are no set rules for the number of times you can use a variable data (field) name in different structures. But when you use the same name more than once, you can run into problems when using multiple files in a program. For this reason, ESCORT lets you distinguish which file name you want by using variable name qualifiers.

Use

Variable name qualifiers, as their name implies, tell the system exactly which file structure the field name is in. When you reference two structures containing the same field name in a program and don't use variable name qualifiers, the system default selects the field based on the order of the file's appearance in the program.

Example

Consider the following structures:

MASTVEND

VNUMBER	5N	K1
VNAME	20A	
VADDRESS	30A	
VCITY	30A	

TRANVEND

VNUMBER	5N	MU
AMTPAID	5V2	
COST	4V2	

NAMESHIP

VNAME	20A	K1
VADDRESS	30A	
VCITY	30A	
AMTPAID	5V2	

Based on these structures, the following examples show two methods for using variable name qualifiers.

Example 1

You can use variable name qualifiers by specifying the file name:

```
IF VNUMBER OF MASTVEND=VNUMBER OF TRANVEND
```

or

```
IF VNAME OF NAMESHIP CONTAINS ''HATFIELD''
```

Example 2

When you don't use variable name qualifiers, ESCORT chooses the field based on the order of the file's appearance in the program.

If your program is:

```
SELECT DATA OF MASTVEND FROM TRANVEND USING
NAMESHIP
```

the following table shows how ESCORT attributes the following clauses using field names.

If Your Program Statement Clause Reads	ESCORT Assumes You Mean	Why
IF VNUMBER>500	VNUMBER OF MASTVEND	Because MASTVEND is the first file containing the field VNUMBER named in your program
IF VNAME STARTS WITH "STA"	VNAME OF MASTVEND	Because MASTVEND is the first file containing the field VNAME named in your program
COMPUTING AMTPAID=COST*1.5	AMTPAID OF TRANVEND	Because TRANVEND is the first file containing the field AMTPAID in your program

Example 3

Another way of using variable name qualifiers is to use a combination of the previous two examples: by file name and by the order of appearance.

If your program clause reads:

```
IF VNUMBER=VNUMBER OF TRANVEND
```

ESCORT interprets the VNUMBER on the left side of the equal sign to be contained in the structure MASTVEND because it's the first file mentioned in the program that contains the field VNUMBER.

If your program clause reads:

```
COMPUTING AMTPAID=AMTPAID OF NAMESHIP
```

ESCORT interprets the AMTPAID on the left side of the equal sign to be contained in the structure TRANVEND because it's the first file mentioned in the program that contains the field AMTPAID.

6.9. SYSTEM UTILITY FIELDS

Function

In addition to field names you supply, there are also eight system utility fields you can reference in any ESCORT statement. These fields perform often used functions, such as date, time, and report titles, and save you the trouble of programming these fields yourself.

How to distinguish

System utility fields (Table 6-2) are distinguishable from variable fields by the dollar sign ($) prefix.

Table 6-2. System Utility Fields

Field	Length	Function
$DATE	8A	Gives you the current system date (mm/dd/yy)
$DAY	2N	Gives you the current system day
$LINE	3N	Sets the system printer to a specified line number before printing. (See Appendix D for expanded capabilities.)
$MONTH	2N	Gives you the current system month
$PAGE	4N	Sets the system page number to a specific page number
$TIME	5A	Gives you the current system time (hh:mm)
$TITLE	40A	Allows you to provide a title line for printed reports
$YEAR	2N	Gives you the current system year

6.10. HOW TO ENTER A STRUCTURE

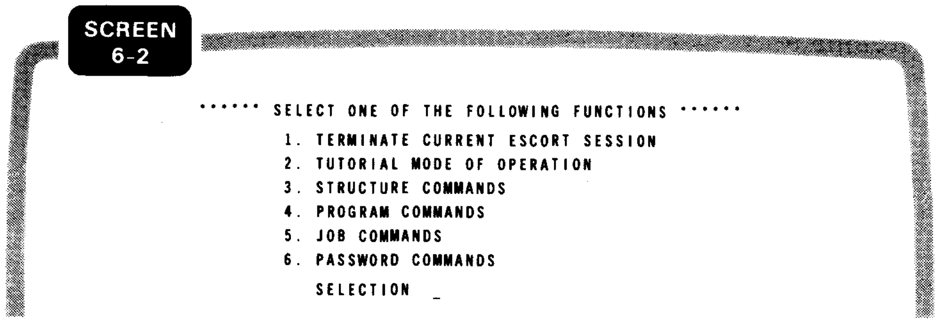
Entering a structure

Now that you know what's required for structure statements, you're ready to enter your own.

As you recall, when you enter program mode the first screen you get is the command selection menu (screen 6-2):

Command selection menu

SCREEN 6-2

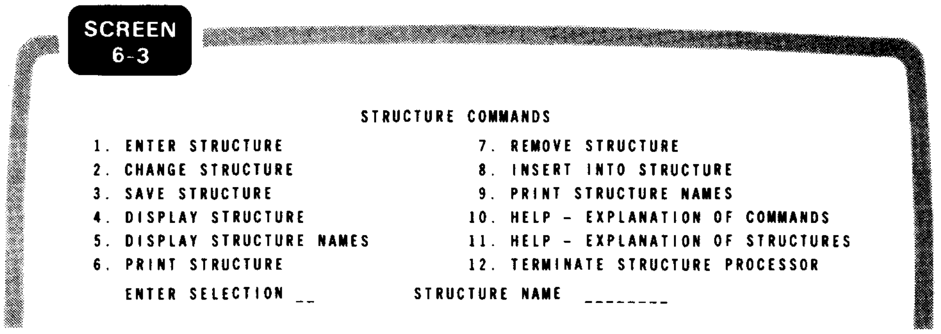


```
****** SELECT ONE OF THE FOLLOWING FUNCTIONS ******
          1. TERMINATE CURRENT ESCORT SESSION
          2. TUTORIAL MODE OF OPERATION
          3. STRUCTURE COMMANDS
          4. PROGRAM COMMANDS
          5. JOB COMMANDS
          6. PASSWORD COMMANDS
             SELECTION _
```

Since you want to enter a structure, choose number 3, STRUCTURE COMMANDS, and screen 6-3 appears:

***STRUCTURE COMMANDS* menu**

SCREEN 6-3



```
                          STRUCTURE COMMANDS
1. ENTER STRUCTURE                      7. REMOVE STRUCTURE
2. CHANGE STRUCTURE                     8. INSERT INTO STRUCTURE
3. SAVE STRUCTURE                       9. PRINT STRUCTURE NAMES
4. DISPLAY STRUCTURE                   10. HELP - EXPLANATION OF COMMANDS
5. DISPLAY STRUCTURE NAMES             11. HELP - EXPLANATION OF STRUCTURES
6. PRINT STRUCTURE                     12. TERMINATE STRUCTURE PROCESSOR
   ENTER SELECTION __          STRUCTURE NAME ________
```

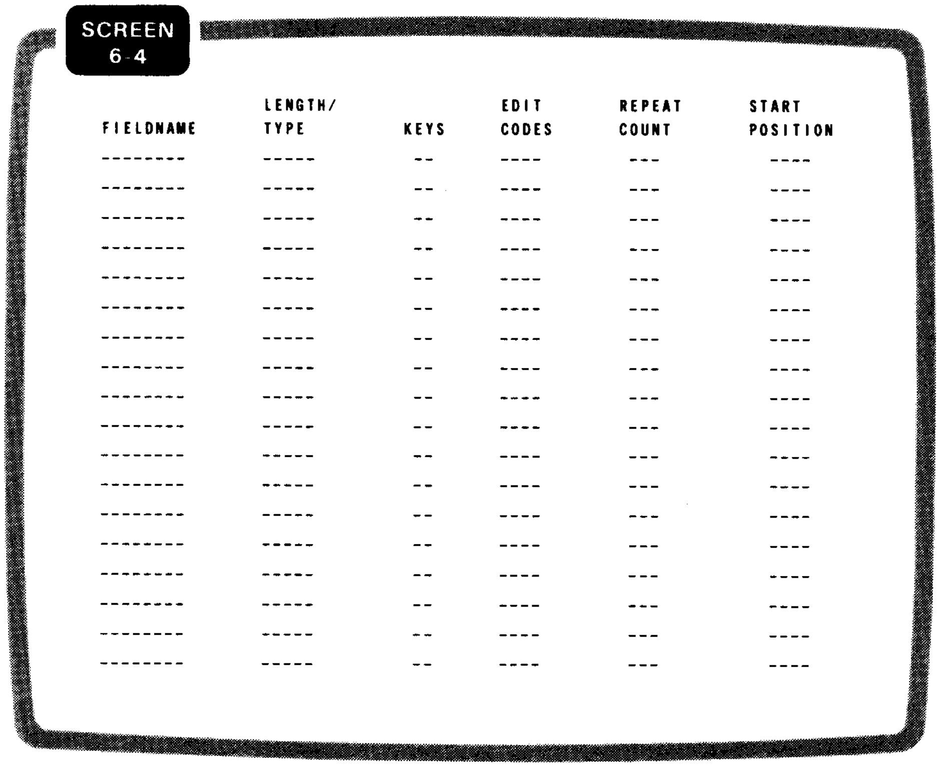
Now you pick the operation you want. Since you want to enter a structure, pick number 1, ENTER STRUCTURE, and name your structure. (The sample structure is named INVNTORY.)

Structure names are one to eight characters long. They must begin with a letter (A–Z) and contain only letters and numbers (0–9). Often, a structure name is the same as the file name or the first eight characters of a longer file name. ESCORT rules for file names and associated structure names are detailed in 7.20, the CREATING clause.

Now screen 6-4, the ENTER STRUCTURE screen, is displayed:

ENTER STRUCTURE screen

SCREEN 6-4



```
                LENGTH/                 EDIT     REPEAT    START
FIELDNAME       TYPE         KEYS       CODES    COUNT     POSITION
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
--------        -----         --        ----      ---        ----
```

Entering the structure

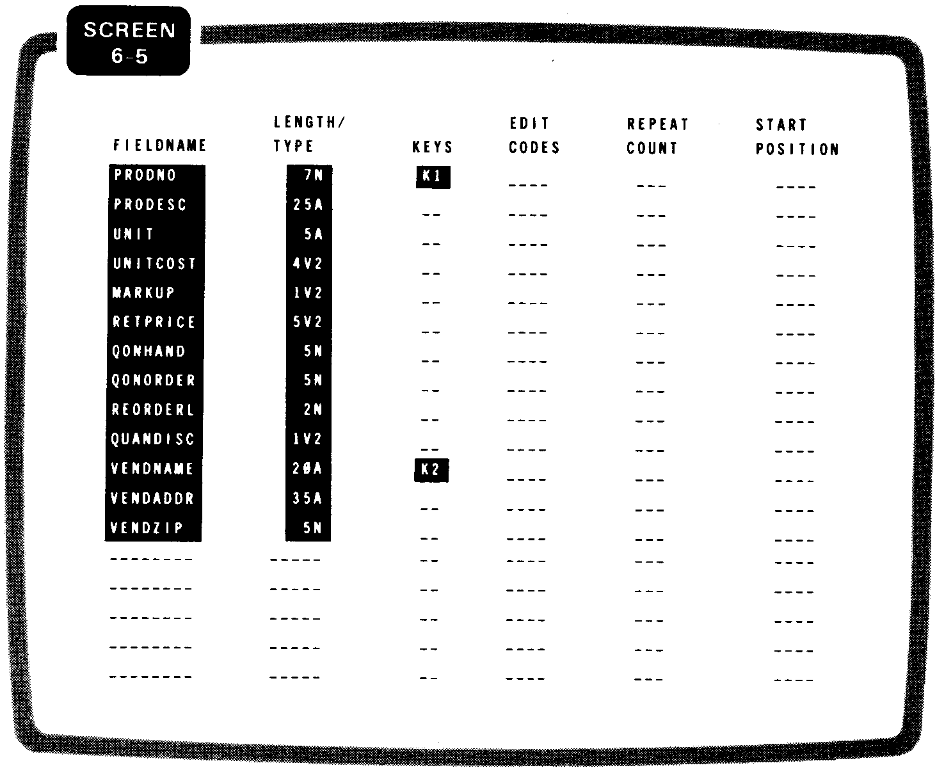
Fill in the blanks with structure statements. Remember, only the FIELDNAME and LENGTH/TYPE portions are required.

Moving from field to field

To move the cursor across the line to each column, press the **TAB** key. To go back to previous fields, press the **TAB BACK** key. To move to the next input line, press the **RETURN** key.

When you key in some structure statements, your screen looks like this (screen 6-5):

Completed structure screen


SCREEN 6-5

FIELDNAME	LENGTH/ TYPE	KEYS	EDIT CODES	REPEAT COUNT	START POSITION
PRODNO	7N	K1	----	---	----
PRODESC	25A	--	----	---	----
UNIT	5A	--	----	---	----
UNITCOST	4V2	--	----	---	----
MARKUP	1V2	--	----	---	----
RETPRICE	5V2	--	----	---	----
QONHAND	5N	--	----	---	----
QONORDER	5N	--	----	---	----
REORDERL	2N	--	----	---	----
QUANDISC	1V2	--	----	---	----
VENDNAME	20A	K2	----	---	----
VENDADDR	35A	--	----	---	----
VENDZIP	5N	--	----	---	----
--------	-----	--	----	---	----
--------	-----	--	----	---	----
--------	-----	--	----	---	----
--------	-----	--	----	---	----
--------	-----	--	----	---	----

Processing the structure

To process your structure, press the **XMIT** key. When a blank screen appears, press the **FUNCTION** and **F1** keys. Then the STRUCTURE COMMANDS menu (screen 6-3) reappears so you can perform your desired processing.

Now, we're ready to discuss changing your structure (6.11) and the various command options available to you in processing your structure (6.12).

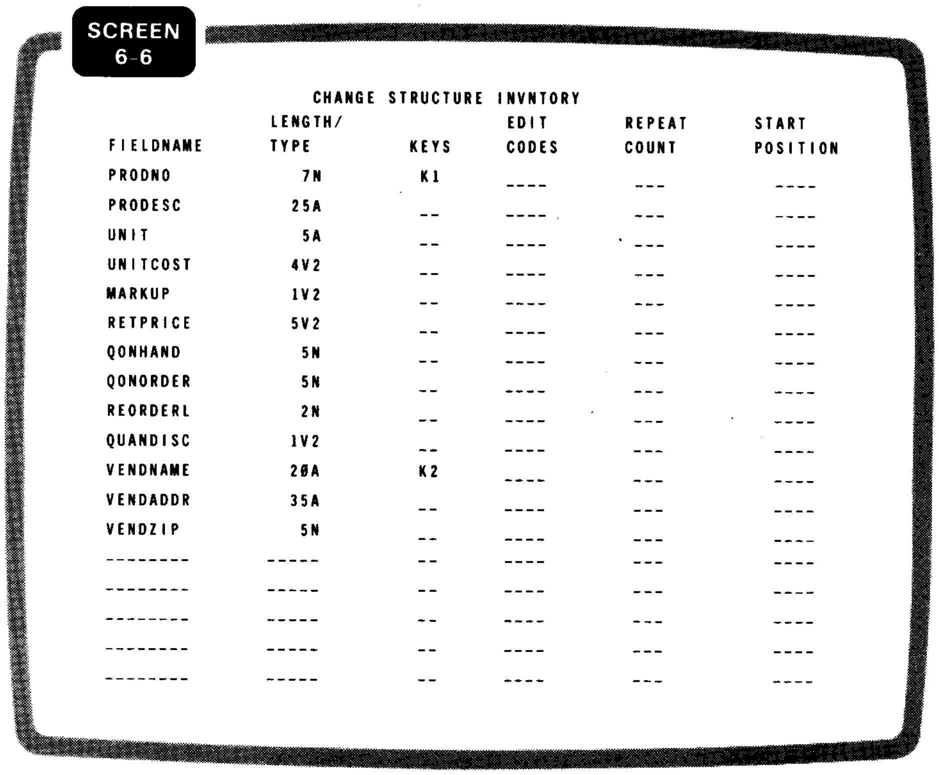
6.11. HOW TO CHANGE A STRUCTURE

Keying in changes

To change a structure, pick choice 2, CHANGE STRUCTURE, on the STRUCTURE COMMANDS menu (screen 6-3). Screen 6-6 then appears, and you make your changes. Key in any changes over the existing entries.

ESCORT displays structure to be changed

SCREEN 6-6



```
                    CHANGE STRUCTURE INVNTORY
            LENGTH/              EDIT      REPEAT    START
FIELDNAME   TYPE        KEYS     CODES     COUNT     POSITION
PRODNO          7N      K1       ----      ---       ----
PRODESC        25A      --       ----      ---       ----
UNIT            5A      --       ----      ---       ----
UNITCOST       4V2      --       ----      ---       ----
MARKUP         1V2      --       ----      ---       ----
RETPRICE       5V2      --       ----      ---       ----
QONHAND         5N      --       ----      ---       ----
QONORDER        5N      --       ----      ---       ----
REORDERL        2N      --       ----      ---       ----
QUANDISC       1V2      --       ----      ---       ----
VENDNAME       2ØA      K2       ----      ---       ----
VENDADDR       35A      --       ----      ---       ----
VENDZIP         5N      --       ----      ---       ----
--------    -----       --       ----      ---       ----
--------    -----       --       ----      ---       ----
--------    -----       --       ----      ---       ----
--------    -----       --       ----      ---       ----
--------    -----       --       ----      ---       ----
```

Making changes permanent

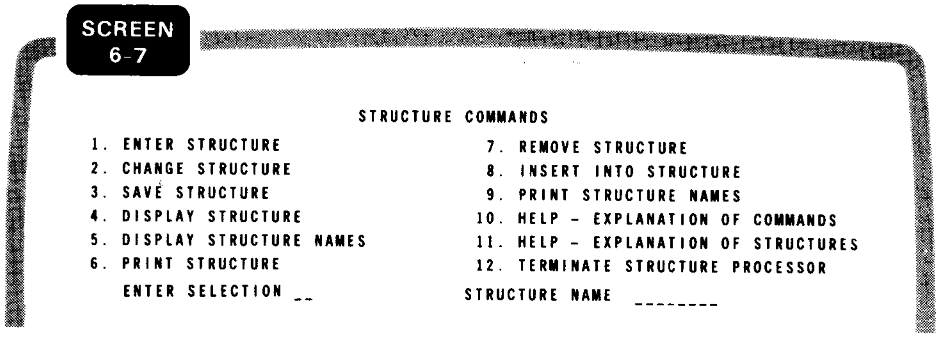
Remember, the changes are made only on the session file copy of your structure. To make the changes permanent, you must re-save the structure under the same name.

6.12. USING COMMANDS TO MANIPULATE YOUR STRUCTURE

Now that you've entered and changed your structure and are back to the STRUCTURE COMMANDS menu (screen 6-7), you can use various command choices to manipulate your structure.

STRUCTURE COMMANDS menu

SCREEN 6-7



```
                        STRUCTURE COMMANDS
1. ENTER STRUCTURE                    7. REMOVE STRUCTURE
2. CHANGE STRUCTURE                   8. INSERT INTO STRUCTURE
3. SAVE STRUCTURE                     9. PRINT STRUCTURE NAMES
4. DISPLAY STRUCTURE                 10. HELP - EXPLANATION OF COMMANDS
5. DISPLAY STRUCTURE NAMES           11. HELP - EXPLANATION OF STRUCTURES
6. PRINT STRUCTURE                   12. TERMINATE STRUCTURE PROCESSOR
   ENTER SELECTION __               STRUCTURE NAME  ________
```

Selection:

Selection action

1	Displays the ENTER STRUCTURE screen (6-4)
2	Displays the CHANGE STRUCTURE screen (6-6)
3	Saves your structure on a library under a name you specify
4	Displays your structure on the workstation
5	Displays a list of all existing structure names on the workstation
6	Prints your structure on the printer
7	Lets you remove a structure from either the ESCORT library file or the session file
8	Lets you insert statements into an existing structure
9	Prints your structure names on the printer
10	Explains the command choices
11	Explains what structures are
12	Lets you return to the command selection menu (screen 6-2)

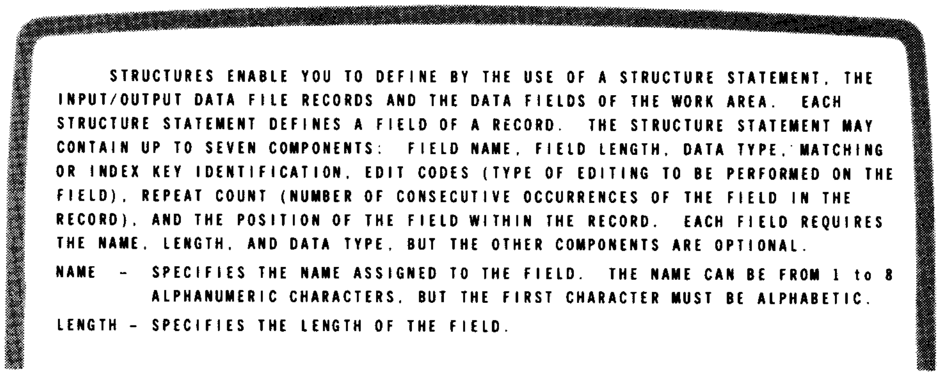
For more detail on choices 1 through 9, see 5.16, STRUCTURE COMMANDS.

6.13. THE STRUCTURE HELP FACILITY

When to use

The structure HELP facility is no different from the HELP facility in program mode except that it works with structures. When you don't understand something on the structure screen menus, just key in the selection number for the HELP facility, and a series of screens explaining the part you are having trouble with appears. A sample structure HELP screen is:

Sample structure HELP screen



```
     STRUCTURES ENABLE YOU TO DEFINE BY THE USE OF A STRUCTURE STATEMENT, THE
INPUT/OUTPUT DATA FILE RECORDS AND THE DATA FIELDS OF THE WORK AREA.  EACH
STRUCTURE STATEMENT DEFINES A FIELD OF A RECORD.  THE STRUCTURE STATEMENT MAY
CONTAIN UP TO SEVEN COMPONENTS:  FIELD NAME, FIELD LENGTH, DATA TYPE, MATCHING
OR INDEX KEY IDENTIFICATION, EDIT CODES (TYPE OF EDITING TO BE PERFORMED ON THE
FIELD), REPEAT COUNT (NUMBER OF CONSECUTIVE OCCURRENCES OF THE FIELD IN THE
RECORD), AND THE POSITION OF THE FIELD WITHIN THE RECORD.  EACH FIELD REQUIRES
THE NAME, LENGTH, AND DATA TYPE, BUT THE OTHER COMPONENTS ARE OPTIONAL.
NAME   -  SPECIFIES THE NAME ASSIGNED TO THE FIELD.  THE NAME CAN BE FROM 1 to 8
          ALPHANUMERIC CHARACTERS, BUT THE FIRST CHARACTER MUST BE ALPHABETIC.
LENGTH -  SPECIFIES THE LENGTH OF THE FIELD.
```

7. Programs and Program Statements

7.1. WHAT ARE PROGRAM STATEMENTS?

Program statements

As you recall from Section 1, program statements are combinations of ESCORT words that make up different types of ESCORT programs. Depending on the type of program statement you use, ESCORT performs different operations on your data.

Use in tutorial mode

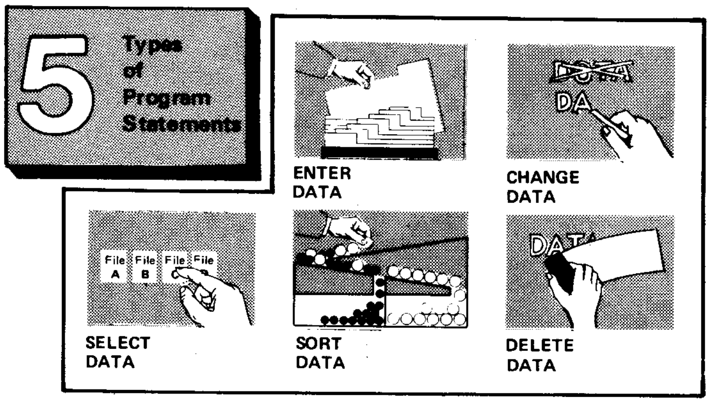
In tutorial mode, ESCORT asks you questions about what you want your program to do and generates program statements based on your answers to these questions.

Use in program mode

In program mode, ESCORT helps you create programs by using screen menu choices or lets you key in your own programs by using the direct entry method. The ESCORT compiler verifies your program statements and translates them into executable code. Then, when you issue a RUN PROGRAM command, the run-time processor executes your program.

Types of program statements

There are five types of program statements:



Function of program statements

Every time you write a program, you use one of these statements. In their basic format (without clauses), they describe the general operation you want to perform on your file. In their expanded form (with clauses), they give more specifics about how you want the operation performed. You'll get an idea of how you can be more specific with clauses if you look at Table 7-1, which lists the clauses and shows which clauses apply to each type of program statement.

Table 7-1. Program Statement and Clause Cross-reference

Clauses	ENTER DATA	CHANGE DATA	SELECT DATA	SORT DATA	DELETE DATA
BY				●	
CLEARING	●	●	●		
COMPUTING	●	●	●		
CREATING	●		●	●	
DISPLAY	●	●	●		
END	●	●	●		
EXIT	●	●	●		
EXTENDING	●		●		
IF	●	●	●		●
OUTPUT TO	●	●	●		
PRINT	●	●	●		
SUBTOTALING	●	●	●		
TOTALING	●	●	●		
UPDATING			●		
USING		●	●		
WHILE	●	●	●		
WORKAREA	●	●	●		

LEGEND

● Applicable

NOTE:

The FROM clause does not appear in this table because it is part of the basic program statement.

We'll tell more about the individual clauses in 7.16. But, first, we'll look at the formats of program statements and how they work.

7.2. PROGRAM STATEMENT FORMAT CONVENTIONS

The conventions used to present program statement formats are:

PROGRAM STATEMENT FORMAT CONVENTIONS

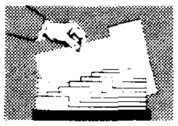
Uppercase words	A	Words in capital letters must be keyed in exactly as shown: `ENTER DATA FROM WS`
Lowercase words	a	Lowercase letters and embedded hyphens designate names you supply: `USING input-file-1,...,input-file-n`
Brackets	[ ]	Optional parameters are enclosed by brackets: `[IF conditions]`
Braces	{ }	Alternative choices are enclosed by braces. When you specify the parameter, you must pick one of the choices: $\left\{\begin{array}{l}\texttt{structure}\\ \texttt{structure,form}\end{array}\right\}$

7.3. ENTER DATA PROGRAM STATEMENT

Use

Probably one of the most used of the five types of program statements is the ENTER DATA program statement. You use this statement every time you create a new file or enter data into an existing file.

Related clauses



The clauses of the ENTER DATA statement let you:

Clear data fields (CLEARING clause)	Specify conditions for data entry (IF and WHILE clauses)
Compute arithmetic expressions (COMPUTING clause)	Output custom formatted data to a workstation or printer (OUTPUT TO clause)
Create a new file (CREATING clause)	Output automatically formatted data to the printer (PRINT clause)
Output automatically formatted data to the workstation (DISPLAY clause)	Subtotal and total fields (SUBTOTALING and TOTALING clauses)
Add data to an existing file (EXTENDING clause)	Allocate working storage (WORKAREA clause)

Other clauses

Other clauses used with the ENTER DATA program statement are:

- END clause
- EXIT clause

7.4. COMPLETE FORMAT OF THE ENTER DATA PROGRAM STATEMENT

The format of the ENTER DATA statement and its related clauses is:

Basic format

```
ENTER DATA FROM WS [{structure      }]
                   [{structure,form }]
WORKAREA structure [data replacement statements]
SUBTOTALING fieldnames [''literal expressions'']
TOTALING fieldnames [''literal expressions'']
IF {conditions} clauses [ELSE clauses] END
   {FIRSTIME  }
   {ENDOFILE  }
COMPUTING data replacement statements
WHILE conditions clauses END
CLEARING fieldnames
CREATING output-file [structure]
EXTENDING output-file [structure]
PRINT [{fieldnames [''literal expressions'']}]
      [{SUBTOTALS                            }]
      [{TOTALS                               }]
      [{structure                            }]
DISPLAY [{fieldnames [''literal expressions'']}]
        [{SUBTOTALS                            }]
        [{TOTALS                               }]
        [{structure                            }]
OUTPUT TO {WS     }{structure     }
          {PRINTER}{structure,form}
EXIT
```

ENTER DATA STATEMENT

7.5. BASIC ENTER DATA PROGRAM STATEMENT

Clauses

Because most of the clauses used by the ENTER DATA statement are shared by other program statements (see Table 7-1), we'll discuss them alphabetically beginning in 7.16. For now, just look at the parameters of the basic ENTER DATA statement:

Format

```
ENTER DATA FROM WS [{structure      }]
                   [{structure,form }]
```

Parameters

structure

Names the structure used for the input data record.

Parameter uses

Use this parameter to name the input structure when you input data to certain fields in a master file. To do this, you create another structure containing the fields you need to input. When your program is executed, enter data into the fields you designated in that structure. ESCORT matches these field names with the field names in the master file and enters your data accordingly.

Example

Consider the following example:

Example structures

Master File Structure

ACCTMAST	
ACCTNO	7N
NAME	20A
ADDRESS	20A
CITY	20A
PHONE	12A
ACCTBAL	5V2
DEPOSIT	5V2
WITHDRAW	5V2
OLDBAL	5V2

Input File Structure

ACCTINIT	
ACCTNO	7N
NAME	20A
ADDRESS	20A
CITY	20A
PHONE	12A
DEPOSIT	5V2

For this example, we've assumed that a bank will use these structures. The file ACCTMAST represents a file containing all the bank's accounts, and the structure ACCTINIT contains the fields a clerk needs to enter to start new accounts. This means the master file called ACCTMAST contains all the fields in the structure ACCTMAST, but with this program, you only enter data into the fields specified in ACCTINIT.

Example program

You write your program like this:

```
ENTER DATA FROM WS (ACCTINIT)
CREATING ACCTMAST
```

`structure,form`

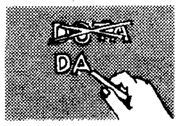
This enables you to use a customized form for input. *Structure* names the structure that's linked to the form. (If we used a form for our previous example, the structure would be ACCTINIT.)

Form names the form you created to use with the structure. For details on forms, see Section 9.

7.6. CHANGE DATA PROGRAM STATEMENT

Use

Use the CHANGE DATA program statement to replace or update data records within an existing file. As with the other program statements, you use a set of related clauses to perform other functions on this data.



The clauses of the CHANGE DATA statement let you:

Related clauses

Clear data fields (CLEARING clause)	Output automatically formatted data to the printer (PRINT clause)
Compute equations (COMPUTING clause)	Subtotal and total fields (SUBTOTALING and TOTALING clauses)
Output automatically formatted data to the workstation (DISPLAY clause)	Use additional files as input (USING clause)
Specify conditions for data change (IF and WHILE clauses)	Allocate working storage (WORKAREA clause)
Output custom formatted data to the workstation or printer (OUTPUT TO clause)	

Other clauses

Other clauses used with the CHANGE DATA program statement are:

- END clause
- EXIT clause

CHANGE DATA STATEMENT

7.7. COMPLETE FORMAT OF THE CHANGE DATA PROGRAM STATEMENT

The format of the CHANGE DATA statement is:

Basic format

```
CHANGE DATA OF master-file [structure]
  ⎡FROM⎧WS ⎡⎧structure      ⎫⎤ ⎤
  ⎢    ⎨   ⎣⎩structure,form ⎭⎦ ⎬
  ⎣    ⎩input-file [structure] ⎭⎦

[USING input-file-1 [structure],...,input-file-5 [structure]]
WORKAREA structure [data replacement statements]
SUBTOTALING fieldnames [''literal expressions'']
TOTALING fieldnames [''literal expressions'']
IF⎧conditions⎫clauses [ELSE clauses] END
  ⎨FIRSTIME  ⎬
  ⎩ENDOFILE  ⎭
COMPUTING data replacement statements
WHILE conditions clauses END
CLEARING fieldnames
PRINT⎡⎧fieldnames [''literal expressions'']⎫⎤
     ⎢⎪SUBTOTALS                          ⎪⎥
     ⎢⎨TOTALS                             ⎬⎥
     ⎣⎩structure                          ⎭⎦
DISPLAY⎡⎧fieldnames [''literal expressions'']⎫⎤
       ⎢⎪SUBTOTALS                          ⎪⎥
       ⎢⎨TOTALS                             ⎬⎥
       ⎣⎩structure                          ⎭⎦
OUTPUT TO⎧WS     ⎫⎧structure     ⎫
         ⎩PRINTER⎭⎩structure,form⎭
EXIT
```

CHANGE
DATA
STATEMENT

7.8. BASIC CHANGE DATA PROGRAM STATEMENT

Clauses

Because most of the clauses used by the CHANGE DATA statement are shared by other program statements (see Table 7-1), we'll discuss them alphabetically beginning in 7.16. For now, just look at the parameters of the basic CHANGE DATA statement:

Format

```
CHANGE DATA OF master-file [structure]
  ┌FROM┌WS┌┌structure       ┐┐ ┐┐
  │    │  │└structure,form  ┘┘ ││
  │    │                       ││
  └    └input-file [structure] ┘┘
```

Parameters

master-file
Names the file you want changed.

structure
Is included when the name of the structure to be used is not the same as the first eight characters of the master-file name.

WS
Changes are input from the workstation.

input-file
Names input file that contains changed data.

Parameter usage

When you use this parameter, the input file you name is a transactional file that contains the records you want to change in the master file. When you execute this program, ESCORT matches the fields in the transactional file with the fields in the master file and puts the contents of the input (transactional) file into the master file as changed data.

structure
Names structure used for input data record.

Use this parameter with an input file only if the structure name is different from the first eight characters of the input file name.

You can name a structure for workstation input that includes only the fields of the master file record you want to change. This is helpful because when you make changes you need not look at every field in the master file; only the data fields you want changed are displayed.

CAUTION

When you use this parameter with workstation input, make sure you key data into every field that is displayed. Fields that have the same name in the input and master files are replaced in the master file with the input you key in. If you don't key new data into the fields, ESCORT transfers them to the master file as blanks or zeros, thereby destroying your existing master file data.

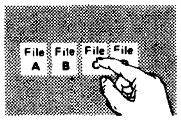
`structure,form` (Used only with WS parameter)
Names structure and form used to input changes

This displays your data in the form you specify, and you input your changes.

7.9. SELECT DATA PROGRAM STATEMENT

Use

Use the SELECT DATA program statement to extract data from an existing file. As in the other program statements, use a set of related clauses to perform other functions on this data.



The clauses of the SELECT DATA statement let you:

Related clauses

Clear data fields
(CLEARING clause)

Compute arithmetic expressions
(COMPUTING clause)

Create a new file from selected records in an existing file
(CREATING clause)

Output automatically formatted data to the workstation
(DISPLAY clause)

Add selected data to an existing file
(EXTENDING clause)

Specify conditions for data selection
(IF and WHILE clauses)

Output custom formatted data to the workstation or printer
(OUTPUT TO clause)

Output automatically formatted data to the printer
(PRINT clause)

Subtotal and total fields
(SUBTOTALING and TOTALING clauses)

Update records in a file
(UPDATING clause)

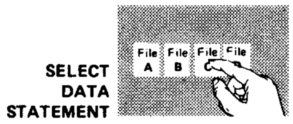
Use additional files as input
(USING clause)

Allocate working storage
(WORKAREA clause)

Other clauses

Other clauses used with the SELECT DATA program statement are:

- END clause
- EXIT clause

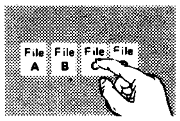


7.10. COMPLETE FORMAT OF THE SELECT DATA PROGRAM STATEMENT

The format of the SELECT DATA statement and its related clauses is:

Basic format

```
SELECT DATA OF master-file [structure]
   [FROM {WS [{structure       }]      }]
   [     {    [{structure,form }]      }]
   [     {input-file [structure]       }]
[USING input-file-1 [structure],...,input-file-5 [structure]]
WORKAREA structure [data replacement statements]
SUBTOTALING fieldnames [''literal expressions'']
TOTALING fieldnames [''literal expressions'']
IF {conditions} clauses  [ELSE clauses] END
   {FIRSTIME  }
   {ENDOFILE  }
COMPUTING data replacement statements
WHILE conditions clauses END
CLEARING fieldnames
CREATING output-file [structure]
EXTENDING output-file [structure]
UPDATING input-file
OUTPUT TO {WS     } {structure      }
          {PRINTER} {structure,form }
PRINT [{fieldnames [''literal expressions'']}]
      [{SUBTOTALS                            }]
      [{TOTALS                               }]
      [{structure                            }]
DISPLAY [{fieldnames [''literal expressions'']}]
        [{SUBTOTALS                            }]
        [{TOTALS                               }]
        [{structure                            }]
EXIT
```



SELECT DATA STATEMENT

7.11. BASIC SELECT DATA PROGRAM STATEMENT

Clauses

Because most of the other clauses used by the SELECT DATA statement are shared by other program statements (see Table 7-1), we'll discuss them alphabetically beginning in 7.16. For now, just look at the parameters of the basic SELECT DATA statement.

Format

```
SELECT DATA OF master-file [structure]
 ⎡FROM⎧WS ⎡⎧structure      ⎫⎤ ⎫⎤
 ⎢    ⎨   ⎣⎩structure,form ⎭⎦ ⎬⎥
 ⎣    ⎩input-file [structure] ⎭⎦
```

Parameters

`master-file`
Names the data file you're selecting.

`structure`
Is included when the name of the structure to be used is not the same as the first eight characters of the master-file name.

`WS`
You key in data selection criteria from the workstation.

`input-file`
Names the file containing the selection criteria.

`structure`
Names the structure used for the input data record.

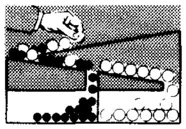
Use this parameter only if the structure name is different from the first eight characters of the input file name.

`structure,form` (Used with WS parameter only)
Names the structure and form used for the workstation. This displays your data in the form you specify.

7.12. SORT DATA PROGRAM STATEMENT

Uses

Use the SORT DATA program statement to sort a data file by field names or a sort key and create a new sorted file.



SORT DATA statement:

Clause restrictions

Whereas other program statements use many clauses, the CREATING and BY clauses are the only ones you're permitted to use with the basic SORT DATA statement. Because these clauses are the only ones used, they are shown as part of the basic SORT DATA statement. A description of the CREATING clause appears in 7.20.

Sort field restriction

The maximum total length of the fields used to determine the sequence of the records in the output file is 80 bytes.

Information on the sequencing of the records in the output file is given in Appendix D.

7.13. FORMAT OF THE SORT DATA PROGRAM STATEMENT

The format of the SORT DATA statement is:

Format

```
SORT DATA OF master-file [structure]
  [BY field-1,...,field-6]
  CREATING output-file [structure]
```

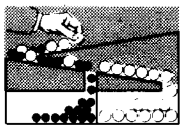
Parameters

`master-file`
Names the file you want sorted.

`structure`
This parameter is required only if the name of the structure is different from the first eight characters of the master-file name.

`BY field-1,...,field-6`
Names the fields in the master-file structure that you want to use to sort your file.

The order in which you list these fields determines the order of the sort (e.g., your data is sorted according to the first field you specify, then the second, etc). For an example of the use of the BY clause, see 4.20.



`CREATING output-file`

Names the file to be created as a result of the sort.

This must be a file other than the master-file.

`structure`

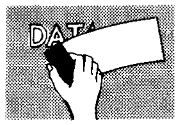
This parameter is required only if the name of the structure is different from the first eight characters of the output-file name.

If you do not include the BY clause, you must use a structure for the output file that has one field specified as an MS (matching sorted) data field. Otherwise, you get an error. For an example of this method of sorting data, see 4.21.

7.14. DELETE DATA PROGRAM STATEMENT

Uses

Use the DELETE DATA program statement to logically remove data records from a file.



DELETE DATA statement:

Whereas other program statements use many clauses, the IF clause is the only clause you're permitted to use with the basic DELETE DATA statement. Because IF is the only clause used, it is shown as part of the basic DELETE DATA statement. No END clause is included in the DELETE DATA program statement.

7.15. FORMAT OF THE DELETE DATA PROGRAM STATEMENT

The format of the DELETE DATA statement is:

Format

```
DELETE DATA OF master-file [structure]
 ⎧FROM⎧WS ⎡⎧structure          ⎫⎤ ⎫ ⎫
 ⎪    ⎨   ⎣⎩structure,form     ⎭⎦ ⎬ ⎪
 ⎨    ⎩input-file[structure]       ⎭ ⎬
 ⎩IF conditions                      ⎭
```

Parameters

master-file
: Names the file from which you want records deleted.

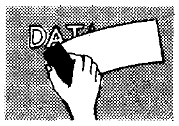
structure
: Use this parameter only when the name of the structure you use is not the same as the first eight characters of the master-file name.

FROM WS
: Enter matching criteria for record deletion from the workstation.

FROM input-file
: Names the file containing the matching criteria for record deletion.

structure
: Names the structure used for input data records.

: Use this parameter only if the structure name is different from the first eight characters of the input file name.



DELETE DATA STATEMENT

`structure,form` (Used with WS parameter only)
  Names structure and form used to enter input records.

`IF conditions`
  Criteria that must be satisfied for deletion to occur.

7.16. CLAUSES USED WITH PROGRAM STATEMENTS

Function of clauses

While ESCORT program statements describe the general operations to perform, ESCORT clauses describe the specific ones.

There are 17 ESCORT clauses, and though some apply exclusively to one program statement type, many are shared by more than one.

How to find clauses

For ease of reference, clauses are presented in alphabetical order.

1. Look up the clause you want to use.
2. Check the chart at the top of the clause description to see if the clause applies to the program statement you're using. The unshaded blocks show the program statements the clause is used with.
3. If the clause applies, read the description and examples.
4. If it doesn't, check the format for the program statement you're using and select another clause.

BY CLAUSE

7.17. HOW TO DESIGNATE SORT FIELDS (BY CLAUSE)

Use

Use the BY clause exclusively with the SORT DATA program statement to designate the fields you want your data sorted by.

The format is:

Format

```
BY field-1,...., field-6
```

Parameters

```
field-1,....,field-6
```

Names the fields you use to sort your files.

The order in which you list the fields determines the order of the sort.

CLEARING CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.18. HOW TO CLEAR DATA FIELDS (CLEARING CLAUSE)

Use

Use this clause to clear data fields you specify.

The format is:

Format

```
CLEARING fieldnames
```

Parameters

`fieldnames`
Names the fields you want cleared.

Example: clearing alphanumeric fields

Alphanumeric fields are cleared to blanks.

```
NAME ____________________
CITY ________________________
```

Example: clearing numeric, decimal, packed, and binary fields

Numeric, decimal, packed, and binary fields are zero filled.

```
ZIPCODE ØØØØØ
BALANCE ØØØØØ.ØØ
```

COMPUTING CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.19. HOW TO CALCULATE EXPRESSIONS AND STORE THE RESULTS (COMPUTING CLAUSE)

Use

Use this clause to calculate expressions and store the results.

The format is:

Format

```
COMPUTING data replacement statements
```

Parameters

`data replacement statements`
Specify arithmetic expressions to be completed or data movement to be performed.

Example: data replacement statements

Examples of the two types of data replacement statements are:

- Arithmetic

  Computes values of one or more fields to result in a single value for another field.

  ```
  COMPUTING BALANCE=DEPOSITS+OLDBAL-WITHDRAW
  ```

  or

  ```
  COMPUTING TOTPAY=HOURS*RATE
  ```

- Data Movement

  Moves the result of a computation of one or more fields into another field.

  ```
  COMPUTING FLDA=FLDB
  ```

  or

  ```
  COMPUTING FLDA=''ABC''
  ```

7.20. HOW TO CREATE A NEW DATA FILE (CREATING CLAUSE)

Use

Use this clause to create a new data file. This file can contain new data or the results of a SELECT or SORT operation.

The format is:

Format

```
CREATING output-file [structure]
```

Parameters

`output-file`

Names the file you're creating.

File name attributes

File names are 1 to 44 characters long and are specified in either of two ways:

- Only letters (A-Z), numbers (0-9), and the characters slash (/) and period (.) may be used in ESCORT file names. The first character must be a letter, and embedded blanks are not allowed.

  Example:

  ```
  XYZCO/PAYABLES.TAXES
  ```

- When you enclose the file name in double quotes (''), any character except a single quote (') or a double quote may be included, and embedded blanks are allowed.

  Example:

  ```
  ''INVENTORY MASTER FILE''
  ```

`structure`

Names the structure for the file you're creating.

Structure name attributes

A structure name may be from one to eight characters long. It must begin with a letter (A–Z) and contain only letters and numbers (0–9).

If your structure name is the same as your file name or as the first eight characters of your file name, you don't need to use this parameter. Otherwise, you must include, in parentheses, the name of the structure you want to use with your file.

Examples:

```
CREATING XYZFILE
CREATING ''#4INVENTORY FILE'' (INVNTORY)
```

DISPLAY CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.21. HOW TO DISPLAY DATA ON THE WORKSTATION (DISPLAY CLAUSE)

Use

Use this clause to output data on the workstation, using automatic formatting.

The format is:

Format

```
DISPLAY [ { fieldnames [''literal expressions''] } ]
        [ { SUBTOTALS                            } ]
        [ { TOTALS                               } ]
        [ { structure                            } ]
```

Parameters

fieldnames
: Specifies data field names you want displayed as part of your output.

literal expressions (Used only with *fieldnames* parameter)
: Specifies literal expressions you want displayed as part of your output.

SUBTOTALS
: Displays automatically formatted subtotaling data. After data is displayed, SUBTOTALING fields are reset to zero values.

TOTALS
: Displays automatically formatted totaling data. ←

structure
: Names structure containing data field names for the data you want displayed.

END CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.22. HOW TO ENCLOSE CONDITIONALS (END CLAUSE)

Use Use this clause as a sentinel to enclose the clauses affected by an IF or WHILE conditional clause.

The format is:

Format

```
END
```

EXIT CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.23. HOW TO SPECIFY EARLY TERMINATION OF A PROGRAM (EXIT CLAUSE)

Use

Use this clause to terminate execution of your program in an orderly manner before reaching the end of your input file. You usually use this clause to terminate following a conditional IF or WHILE clause. That way, when conditions are satisfied before the end of your file is reached, you can halt processing, saving time.

The format is:

Format

```
EXIT
```

EXTENDING CLAUSE

ENTER DATA CHANGE DATA SELECT DATA SORT DATA DELETE DATA

7.24. HOW TO ADD RECORDS TO AN EXISTING DATA FILE (EXTENDING CLAUSE)

Use

Use this clause to add records to an existing data file.

The format is:

Format

```
EXTENDING output-file [structure]
```

Parameters

`output-file`
Names file you're adding to.

`structure`
Names output structure you're using.

This parameter is not required if your structure name is the same as the first eight characters of your file name.

7.25. HOW TO SPECIFY CONDITIONS FOR PROCESSING (IF CLAUSE)

Use

Use this clause to specify conditions that determine whether to process the clauses that follow. A list of conditions that you can use with this clause appears in 7.34.

The format is:

Format

```
IF⎧conditions⎫clauses [ELSE clauses] END
  ⎨FIRSTIME  ⎬
  ⎩ENDOFILE  ⎭
```

Parameters

conditions

Statements you specify that must be satisfied before the clauses that follow are executed.

Examples

For example, consider the following programs:

Program 1:

```
ENTER DATA FROM WS
IF FIELD1=5
CREATING NEWFILE END
```

Program 2:

```
SELECT DATA OF FILEONE
IF BALANCE >5ØØ
OUTPUT TO PRINTER (strucl,form) END
```

In program 1, the data you enter is only placed in NEWFILE if the value of FIELD1 equals 5.

In program 2, the data you retrieve is output only if the field named BALANCE contains a value greater than 500.

NOTE:

The next two parameters are used only with the IF clause and are called special system operators.

FIRSTIME
: The clauses you specify to be executed (dependent on the IF clause) are executed once after you enter your initial input but before any processing begins (e.g., a title appearing at the top of a report).

ENDOFILE
: Specifies that when the end of the input file is reached, the clauses you specify are executed once.

Example

```
ENTER DATA FROM WS
EXTENDING FILE1
TOTALING FIELD1
IF ENDOFILE
PRINT TOTALS END
```

clauses
: Clauses to be executed, depending on the *conditions, FIRSTIME,* or *ENDOFILE* parameter.

ELSE clauses
: Clauses to be executed if the clauses specified in the *conditions, FIRSTIME,* or *ENDOFILE* parameter are not.

END
: Encloses clauses dependent on the *conditions, FIRSTIME,* or *ENDOFILE* parameters. For more on the END clause, see 7.22.

7.26. HOW TO OUTPUT YOUR DATA IN A UNIQUE FORMAT (OUTPUT TO CLAUSE)

Use

Use this clause to output your data records to the workstation or printer, using a format you supply. This format can be either a structure or a structure and a form you create, using the screen format generator.

The format is:

Format

```
OUTPUT TO{WS     }{structure      }
         {PRINTER}{structure,form }
```

Parameters

WS
: Displays formatted output on the workstation.

PRINTER
: Prints formatted output on the printer. When you output a print form, you are limited to 80 printer positions.

structure
: The structure by which output is formatted.

structure,form
: Names the structure and form you use for output.

  This parameter displays your data, record by record, in the screen format (form) you designate.

PRINT CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.27. HOW TO OUTPUT YOUR DATA TO THE PRINTER, USING AUTOMATIC FORMATTING (PRINT CLAUSE)

Use

Use this clause to output your data on the printer, using automatic formatting.

When you use this statement, your data is printed one of two ways:

1. If your record is short enough, ESCORT prints the fieldnames as column headings with the data records listed underneath.
2. If your record is long, ESCORT prints each field name and its data alongside it. See the printout examples in Section 4.

Format

The format is:

```
PRINT ⎡⎧fieldnames [''literal expressions'']⎫⎤
      ⎢⎨SUBTOTALS                           ⎬⎥
      ⎢⎪TOTALS                              ⎪⎥
      ⎣⎩structure                           ⎭⎦
```

Parameters

`fieldnames`
Data field names you want printed as part of your output.

`literal expressions` (Used only with *fieldnames* parameter)
Literal expressions you want printed as part of your output.

`SUBTOTALS`
Prints automatically formatted subtotaling data.

After data is listed, the SUBTOTALING fields are reset to zero values.

`TOTALS`
Prints automatically formatted totaling data.

`structure`
Names the structure containing the data field names for the data you want listed.

SUBTOTALING CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.28. HOW TO ACCUMULATE SUBTOTALS FOR YOUR FIELDS (SUBTOTALING CLAUSE)

Use

Use the SUBTOTALING clause to accumulate totals for the data fields you specify so you can output them on the workstation or printer. To get subtotals at the end of each page (24 lines for workstation, approximately 55 for printer), you must specify the IF conditions clause and specify a line counter variable (we used LINECNT=55) or ENDOFILE condition. Then the SUBTOTALING data fields are reset to zero values, and the accumulation can begin for the next page.

NOTE:

This clause only performs the SUBTOTALING operation. To receive SUBTOTALING output, you must specify the SUBTOTALS parameter of the PRINT clause.

The format is:

Format

```
SUBTOTALING fieldnames [''literal expressions'']
```

Parameters

`fieldnames`
Names the fields you want subtotaled.

`literal expressions`
Text you specify that is displayed as part of the SUBTOTALING line.

Consider the following example:

Example structures

STRUCTURE OF MYFILE	
FIELDA	5N
FIELDB	5N

STRUCTURE OF WORKAREA (WORK1)	
LINECNT	3N

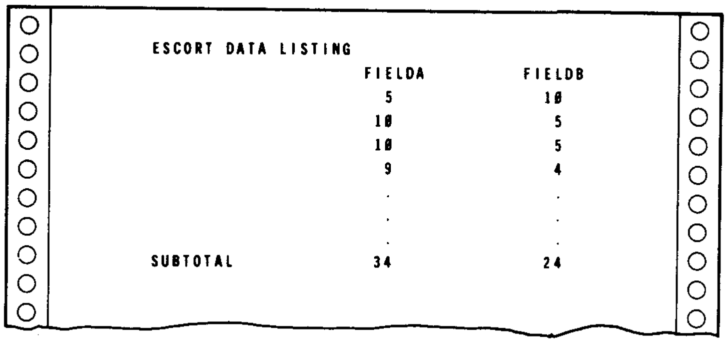
To get SUBTOTALING printed at the bottom of each page, you use the following program:

Subtotaling program example

PROGRAM	EXPLANATION
`SELECT DATA OF MYFILE`	*Simple select data*
`WORKAREA (WORK1) LINECNT=1`	*Designate work area*
`SUBTOTALING ''SUBTOTAL'',FIELDA,FIELDB`	*Subtotaling function*
`IF LINECNT<55`	*Conditional*
`PRINT FIELDA,FIELDB END`	*Action if line 4 is true*
`IF LINECNT=55`	*Conditional*
`PRINT SUBTOTALS`	*Action if line 6 is true*
`COMPUTING LINECNT=0 END`	*Resets LINECNT variable to 0*
`COMPUTING LINECNT=LINECNT+1`	*Counter using work area field*
`IF ENDOFILE`	*When end of file is reached*
`PRINT SUBTOTALS END`	*Perform this operation*

This produces output of:

Subtotaling program output



```
ESCORT DATA LISTING
                           FIELDA          FIELDB
                              5              10
                             10               5
                             10               5
                              9               4
                              .               .
                              .               .
                              .               .
SUBTOTAL                     34              24
```

Line 1
Line 55 or ENDOFILE

TOTALING CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.29. HOW TO ACCUMULATE A RUNNING TOTAL (TOTALING CLAUSE)

Use

Use this clause to accumulate running totals for the data fields you specify so you can output them on the workstation or printer.

The format is:

Format

```
TOTALING fieldnames [''literal expressions'']
```

Parameters

`fieldnames`
Names the fields you want totaled.

`literal expressions`
Text you specify that is displayed as part of the TOTALING line.

UPDATING CLAUSE

SELECT DATA

7.30. HOW TO UPDATE RECORDS IN A FILE (UPDATING CLAUSE)

Use

Use this clause to update records in an existing file.

The format is:

Format

```
UPDATING input-file
```

Parameters

`input-file`

Names the file in which records are to be updated.

The file named must be the master file or one of the input files used in the program.

USING CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.31. HOW TO SPECIFY ADDITIONAL FILES (USING CLAUSE)

Use

Use this clause to specify additional file names when you need to reference fields not contained in the files named in the basic statements. When you use this clause, it's considered part of the basic program statement and must be placed directly after it.

The format is:

Format

```
[USING input-file-1[structure],...,input-file-5[structure]]
```

Parameters

```
input-file-1,...,input-file-5
```

Names up to five additional input files.

For an example of the USING clause, see 4.14.

WHILE CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.32. HOW TO SPECIFY CONDITIONS FOR PROCESSING (WHILE CLAUSE)

Use

Use this clause to apply conditions that, while satisfied, cause the execution of the group of clauses that follow.

The format is:

Format

```
WHILE conditions clauses END
```

Parameters

`conditions`

Statements you specify that determine whether the clauses that follow are executed.

Example

An example of the conditional WHILE is:

```
ENTER DATA FROM WS
WHILE PRINC<20000
COMPUTING INT=PRINC*RATE
PRINT PRINC,RATE,INT
COMPUTING PRINC=1000+PRINC
END
```

This means that while the field PRINC is less than 20000, the computations are performed and the principal, rate, and interest are printed.

`clauses`

Clauses that are executed while conditions are satisfied.

In the preceding example, COMPUTING and PRINT were the clauses executed.

`END`

A sentinel to enclose the clauses affected by the WHILE conditions.

WORKAREA CLAUSE

ENTER DATA | CHANGE DATA | SELECT DATA | SORT DATA | DELETE DATA

7.33. HOW TO ALLOCATE WORKING STORAGE (WORKAREA CLAUSE)

Use

Use the WORKAREA clause to allocate temporary working storage space and initialize values of data fields not defined in input or output structures.

The format is:

Format

```
WORKAREA structure [data replacement statements]
```

Parameters

`structure`
: Names the structure containing data field names for working storage area.

`data replacement statements`
: Initialize fields in the named structure to values other than zeros or blanks.

For an example of WORKAREA, see Figure 4–1.

7.34. CONDITIONS AND SPECIAL OPERATORS USED WITH THE IF AND WHILE CLAUSES

Types of operators

When you use the IF and WHILE clauses, there are four types of operators you can use to specify conditions: relational operators, special character string operators, logical operators, and special system operators.

- Relational Operators

Use

Relational operators specify types of conditions you specify that regulate the entry, modification, or selection of your data. These conditions can be specified in either alphabetic or symbolic form:

Form		Meaning	Example
Alphabetic	Symbolic		
EQ	=	Equal to	IF FIELDA=FIELDB+FIELDC
NE	< >	Not equal to	IF FIELD1< >FIELD6
GT	>	Greater than	WHILE CRLIMIT GT 500
LT	<	Less than	WHILE ACCTBAL LT 1000
GE	>=	Greater than or equal to	IF COUNT GE 24
LE	<=	Less than or equal to	IF AGE LE 30

- Special Character String Operators

Use

Special character string operators let you set conditions on alphanumeric fields by specifying a character string designation. The six string operators are:

Operator	Example
STARTS WITH "string"	IF ACCTNO STARTS WITH "A21"
ENDS WITH "string"*	IF PRODESC ENDS WITH "COAT"
CONTAINS "string"	WHILE PRODESC CONTAINS "CHOCOLATE"
DOES NOT START WITH "string"	IF ADDRESS DOES NOT START WITH "194"
DOES NOT END WITH "string"*	IF PRODCODE DOES NOT END WITH "DIS"
DOES NOT CONTAIN "string"	IF DESC DOES NOT CONTAIN "RED"

*These two character string operators look only at the last *positions* of a field, not the last characters of data entered. Thus, if you use these characters on a field that is not full (i.e., five characters entered into an 8-character field) ESCORT checks only the last positions.

- Logical Operators

Logical operators

There are two logical operators, OR and AND, that you can use to connect one or more conditional expressions. Examples of these logical operators are:

Example 1:

```
IF FIELDA>FIELDB OR FIELDA>23
```

Example 2:

```
IF (FLDA=''36'' AND FLDC=FLDE) OR FLDF=22
```

- Special System Operators (IF clause only)

The special system operators include FIRSTIME and ENDOFILE.

FIRSTIME operator

FIRSTIME executes clauses you specify only once after you enter your initial input but before any processing begins (e.g., a title appearing at the top of a report). ←

ENDOFILE operator

ENDOFILE specifies that once the end of the input file is reached, the clauses you specify are executed once.

8. Jobs

8.1. WHAT IS A JOB?

Job definition

An ESCORT job is a group of related programs that are executed sequentially. You designate these programs and their order of execution when you enter your job. An ESCORT job is not the same as an OS/3 job.

Difference between job and program

A job is different from a program because you can only execute one program at a time, whereas in a job, you can execute up to eight programs in sequence.

Ways to use a job

For example, you might have the following functions (programs) you want to perform on a certain data file:

1. Enter data into a transactional file
2. Sort the data in that transactional file
3. Update the master file with data from the transactional file
4. Output the data of the updated master file

If you don't designate a job, you have to execute each of these programs individually, using the RUN PROGRAM command four times.

Advantages of jobs

With a job, you simply list the names of the programs on the ENTER JOB screen (screen 8-3), and ESCORT executes them sequentially.

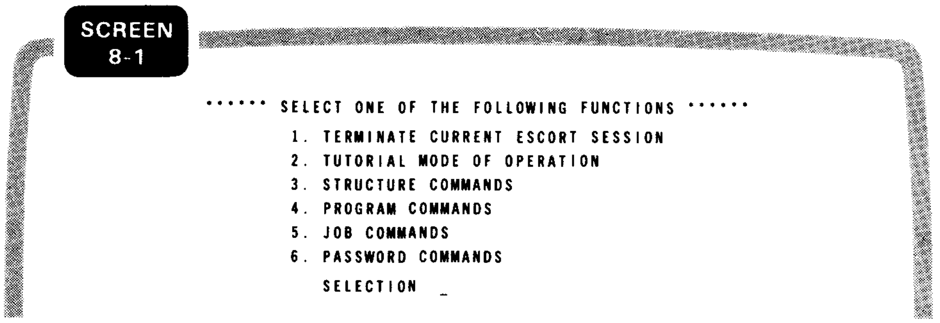
This section tells you how to enter and change your job and how to use other ESCORT commands to manipulate it.

8.2. HOW TO ENTER A JOB

Now that you're familiar with jobs, you're ready to enter your own.

As with all functions of program mode, the first screen that appears is the command selection menu (screen 8-1):

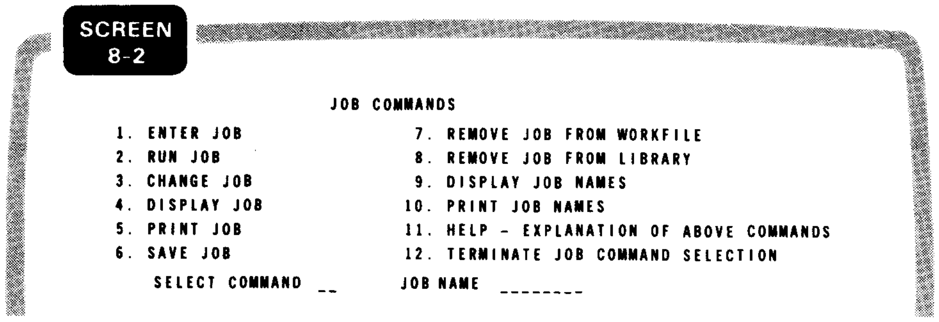
Command selection menu


SCREEN 8-1

```
****** SELECT ONE OF THE FOLLOWING FUNCTIONS ******
        1. TERMINATE CURRENT ESCORT SESSION
        2. TUTORIAL MODE OF OPERATION
        3. STRUCTURE COMMANDS
        4. PROGRAM COMMANDS
        5. JOB COMMANDS
        6. PASSWORD COMMANDS
           SELECTION _
```

Since you want to enter a job, choose number 5, JOB COMMANDS. Screen 8-2 then appears:

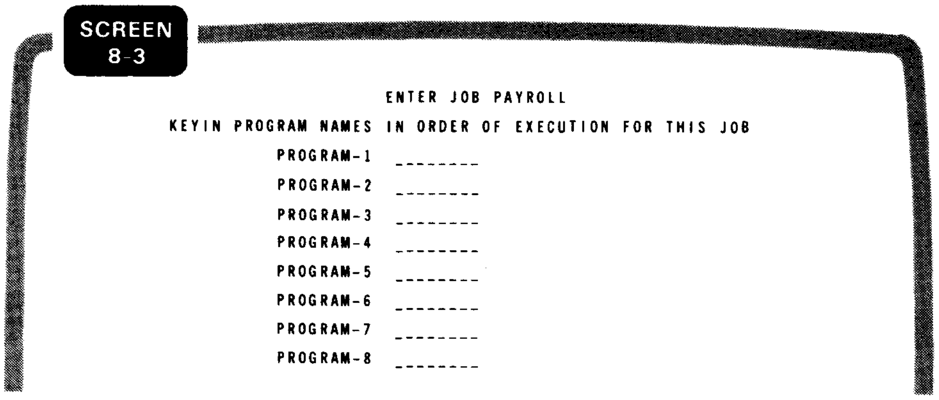
JOB COMMANDS menu


SCREEN 8-2

```
                          JOB COMMANDS
 1. ENTER JOB                    7. REMOVE JOB FROM WORKFILE
 2. RUN JOB                      8. REMOVE JOB FROM LIBRARY
 3. CHANGE JOB                   9. DISPLAY JOB NAMES
 4. DISPLAY JOB                 10. PRINT JOB NAMES
 5. PRINT JOB                   11. HELP - EXPLANATION OF ABOVE COMMANDS
 6. SAVE JOB                    12. TERMINATE JOB COMMAND SELECTION
    SELECT COMMAND __           JOB NAME ________
```

Now you pick the operation you want. Since you want to enter a job, pick number 1, ENTER JOB, and name your job. Assume you name your job PAYROLL. Now the ENTER JOB screen (screen 8-3) appears, and you key in one to eight program names to form a job.

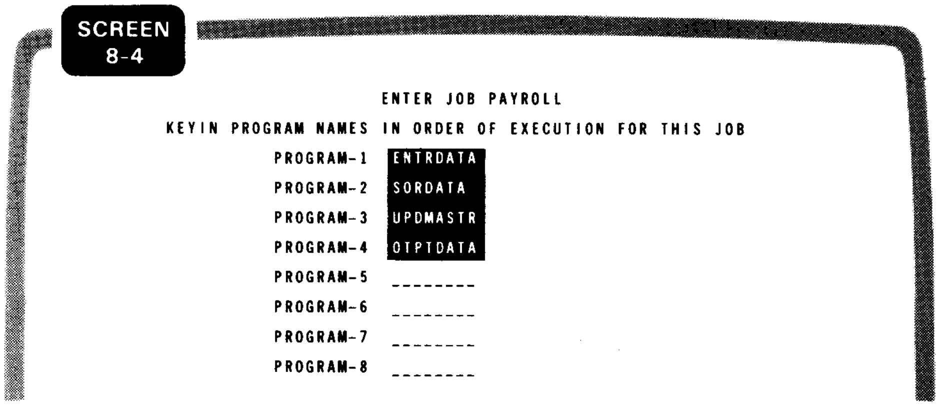
ENTER JOB screen


SCREEN 8-3

```
                ENTER JOB PAYROLL
KEYIN PROGRAM NAMES IN ORDER OF EXECUTION FOR THIS JOB
            PROGRAM-1 ________
            PROGRAM-2 ________
            PROGRAM-3 ________
            PROGRAM-4 ________
            PROGRAM-5 ________
            PROGRAM-6 ________
            PROGRAM-7 ________
            PROGRAM-8 ________
```

When you key in the program names you need for your job (we'll use the four programs mentioned in 8.1), your ENTER JOB screen (screen 8-4) looks like this:

Completed ENTER JOB screen


SCREEN 8-4

```
                    ENTER JOB PAYROLL
KEYIN PROGRAM NAMES IN ORDER OF EXECUTION FOR THIS JOB
          PROGRAM-1 ENTRDATA
          PROGRAM-2 SORDATA
          PROGRAM-3 UPDMASTR
          PROGRAM-4 OTPTDATA
          PROGRAM-5 ________
          PROGRAM-6 ________
          PROGRAM-7 ________
          PROGRAM-8 ________
```

Processing your job

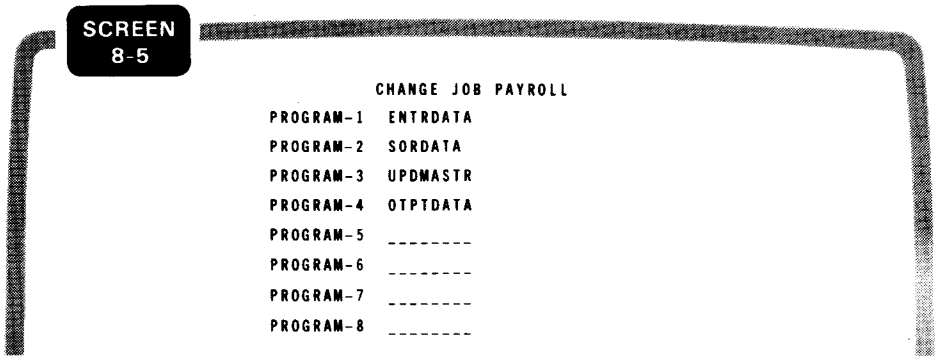
To process your job, press the **XMIT** key. The JOB COMMANDS menu (screen 8-2) then reappears so you can perform your desired processing.

In the next subsection (8.3), we'll discuss the CHANGE JOB command, and then we'll move on to the other command options available when processing your job (8.4).

8.3. HOW TO CHANGE A JOB

To change a job, pick choice 3, CHANGE JOB, from the JOB COMMANDS menu (screen 8-2). Screen 8-5 then appears, and you make your changes.

CHANGE JOB screen


SCREEN 8-5

```
           CHANGE JOB PAYROLL
PROGRAM-1  ENTRDATA
PROGRAM-2  SORDATA
PROGRAM-3  UPDMASTR
PROGRAM-4  OTPTDATA
PROGRAM-5  ________
PROGRAM-6  ________
PROGRAM-7  ________
PROGRAM-8  ________
```

Making changes permanent

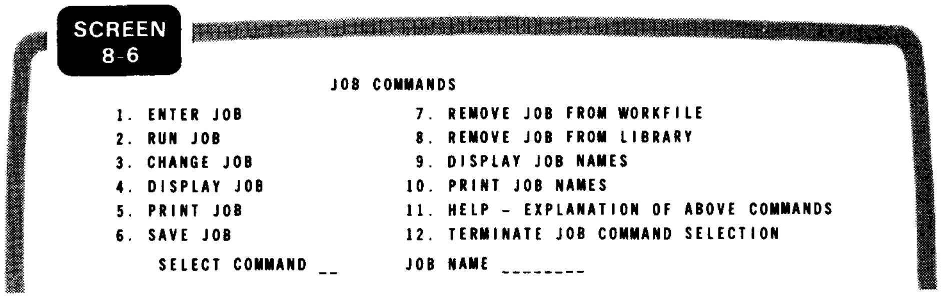
Remember, the changes are made only on the session file copy of your job. To make the changes permanent, you must resave the job under the same name.

If you need a review on changing a job, see the CHANGE JOB command (5.21).

8.4. USING COMMANDS TO MANIPULATE A JOB

Now that you've entered and changed your job and you're back to the JOB COMMANDS menu (screen 8-6), you can use various command choices to manipulate your job.

JOB COMMANDS menu



```
                    JOB COMMANDS
1. ENTER JOB               7. REMOVE JOB FROM WORKFILE
2. RUN JOB                 8. REMOVE JOB FROM LIBRARY
3. CHANGE JOB              9. DISPLAY JOB NAMES
4. DISPLAY JOB            10. PRINT JOB NAMES
5. PRINT JOB              11. HELP - EXPLANATION OF ABOVE COMMANDS
6. SAVE JOB               12. TERMINATE JOB COMMAND SELECTION
   SELECT COMMAND __          JOB NAME ________
```

Command initiated action

Selection:

Selection action

1	Displays the ENTER JOB screen (8-3)
2	Executes your job
3	Displays the CHANGE JOB screen (8-5)
4	Displays your job on the workstation
5	Prints your job on the printer
6	Saves your job in a library under a name you specify
7	Removes your job from the session file (workfile)
8	Removes your job from the ESCORT library
9	Displays a list of all existing job names on the workstation
10	Prints a listing of all existing job names on the printer
11	Explains the command choices
12	Lets you return to the command selection menu (screen 8-1).

For details on choices 1 through 12, see 5.24 through 5.34.

9. How to Create Customized Input/Output Forms

9.1. WHAT IS A FORM?

Concept of forms

We're all familiar with the concept of a form on paper; in fact, we use forms every day. Whether it be a check, an employment application, a tax form or whatever, we use forms because they are an efficient way of standardizing large amounts of data into readable reports.

Screen forms

The forms you use with ESCORT are not paper forms; they're screen forms, and they operate under the OS/3 screen format services component, which is completely external to ESCORT.

Screen format services functions

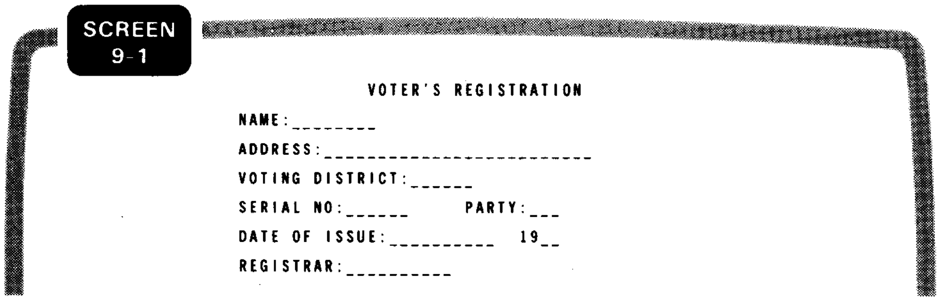
Screen format services lets you create customized forms on your workstation screen and use them to:

- Input data to a program
- Display output data from a program (on workstation or printer) ←
- Input and output data to or from a program

Using input screen forms

Using a screen form to input data is fast and easy because it employs a fill-in-the-blanks approach.

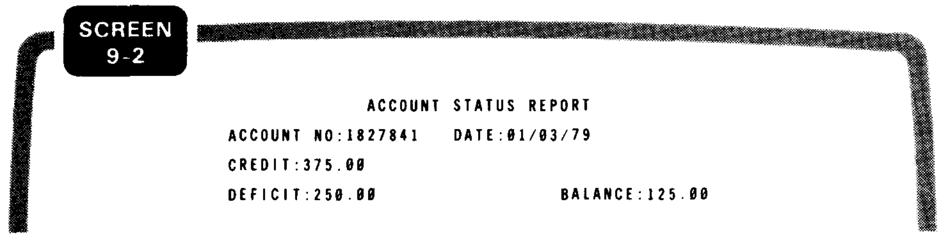
SCREEN 9-1



```
                VOTER'S REGISTRATION
NAME:________
ADDRESS:_________________________
VOTING DISTRICT:______
SERIAL NO:______      PARTY:___
DATE OF ISSUE:__________   19__
REGISTRAR:__________
```

Using output screen forms When you use a screen form for output, the program you link to the form automatically fills the blanks with data.

SCREEN 9-2



```
                    ACCOUNT STATUS REPORT
ACCOUNT NO:1827841    DATE:01/03/79
CREDIT:375.00
DEFICIT:250.00                  BALANCE:125.00
```

Purpose of section This section broadly describes the concept of creating input/output forms and gives an overview of how screen format services works with ESCORT. It does not go into the specifics of actually creating a form because the process is too lengthy to present in this manual.

For your reference For specific information on creating and manipulating forms, refer to the screen format services concepts and facilities manual.

9.2. HOW SCREEN FORMAT SERVICES LETS YOU CREATE YOUR FORM

Elements of screen format services Screen format services consists of the screen format generator and the screen format coordinator.

- Screen Format Generator

  This component allows you to create (generate) screen formats from the workstation.

  Function During generation, you specify the form layout, and the screen format generator automatically stores the completed form in the system library ($Y$FMT). The screen format generator also makes these forms readily available for you to modify, delete, or review.

- Screen Format Coordinator

  Accessing screen forms This component coordinates the transfer of input and output data between your program and the form displayed on the workstation screen. The screen format coordinator is activated when you call an existing form in your program. It retrieves the form from the $Y$FMT library and displays it on the workstation so you can begin your input or view your program's output.

9.3. HOW TO CALL THE SCREEN FORMAT GENERATOR

Although we are not discussing the forms entry procedure in this manual, you may want to call the screen format generator to familiarize yourself with it and perhaps experiment with a form on your own.

Before entering ESCORT

The procedure for calling the screen format generator before you enter ESCORT is simple. Once you've logged on the system:

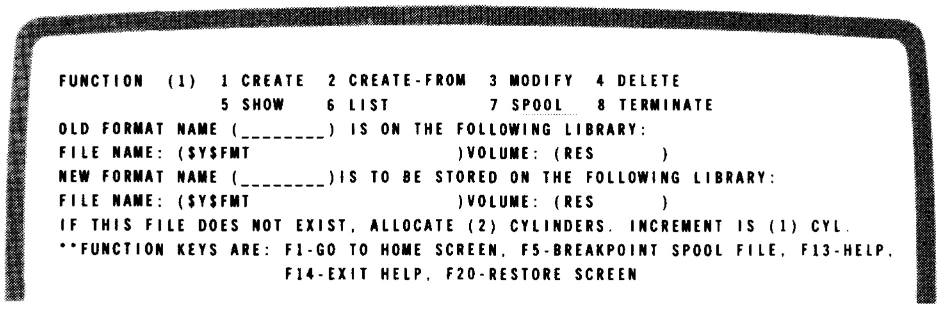
1. Key in the interactive services command:

   RV SFGEN

2. Press the **XMIT** key.

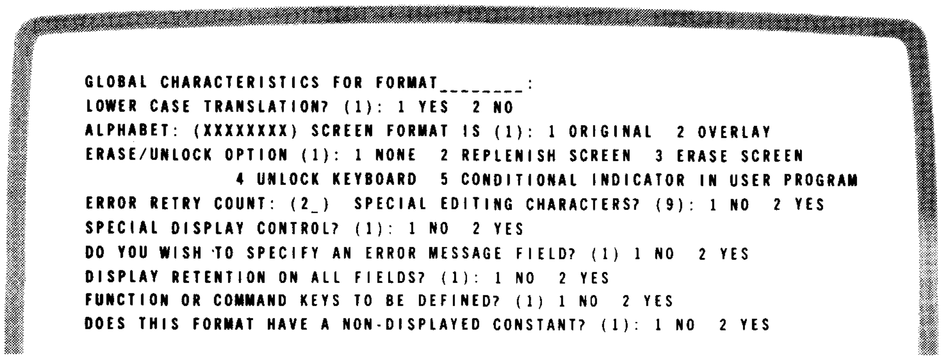
Now you have activated the screen format generator home screen.

First, this screen appears:



```
FUNCTION  (1)  1 CREATE  2 CREATE-FROM  3 MODIFY  4 DELETE
               5 SHOW    6 LIST         7 SPOOL   8 TERMINATE
OLD FORMAT NAME (________) IS ON THE FOLLOWING LIBRARY:
FILE NAME: ($Y$FMT                    )VOLUME: (RES      )
NEW FORMAT NAME (________)IS TO BE STORED ON THE FOLLOWING LIBRARY:
FILE NAME: ($Y$FMT                    )VOLUME: (RES      )
IF THIS FILE DOES NOT EXIST, ALLOCATE (2) CYLINDERS. INCREMENT IS (1) CYL.
**FUNCTION KEYS ARE: F1-GO TO HOME SCREEN, F5-BREAKPOINT SPOOL FILE, F13-HELP,
                     F14-EXIT HELP, F20-RESTORE SCREEN
```

Then, this screen appears:



```
GLOBAL CHARACTERISTICS FOR FORMAT________:
LOWER CASE TRANSLATION? (1): 1 YES  2 NO
ALPHABET: (XXXXXXXX) SCREEN FORMAT IS (1): 1 ORIGINAL  2 OVERLAY
ERASE/UNLOCK OPTION (1): 1 NONE  2 REPLENISH SCREEN  3 ERASE SCREEN
           4 UNLOCK KEYBOARD  5 CONDITIONAL INDICATOR IN USER PROGRAM
ERROR RETRY COUNT: (2_)  SPECIAL EDITING CHARACTERS? (9): 1 NO  2 YES
SPECIAL DISPLAY CONTROL? (1): 1 NO  2 YES
DO YOU WISH TO SPECIFY AN ERROR MESSAGE FIELD? (1) 1 NO  2 YES
DISPLAY RETENTION ON ALL FIELDS? (1): 1 NO  2 YES
FUNCTION OR COMMAND KEYS TO BE DEFINED? (1) 1 NO  2 YES
DOES THIS FORMAT HAVE A NON-DISPLAYED CONSTANT? (1): 1 NO  2 YES
```

By answering the questions on these screens, you select your functions and provide data about the form you're creating.

When you're already in ESCORT and you want to create a form, you exit from ESCORT and, when you're back in the system, perform the same steps listed above.

9.4. HOW TO LINK YOUR ESCORT PROGRAM TO A SCREEN FORM

When you create any ESCORT file, you create a structure for it, and the file is built in accordance with this structure.

Default screen format

When you designate a structure and don't designate a form, you have no choice about how your screen is formatted. ESCORT displays your fields according to the default screen format, which outputs your fields, each on a separate line, in the least possible amount of space (see 9.5).

Customized screen format

When you create a form, you give ESCORT a special format in which to output the file data. You still designate a structure, but instead of your file being output in the default structure format, it's linked with your screen form.

You can use either the screen format generator or custom structures to produce custom formats. Custom structures format output according to your specifications. You determine the placement of data on the page, including line spacing and page breaks. Appendix G gives an example of custom formatting using structures.

Rules for screen forms

When you designate a screen form, the fields you link to it must be identical to the fields contained in a structure. They must have the same name, and they must be in the same order as the corresponding fields in your structure. Both the form and your structure must also have the same number of fields.

For example, if you designate 10 fields in a form and link it to a structure with only 9 fields, you get an error because ESCORT keeps looking for the tenth field.

It is possible, however, to have a structure with 10 fields and an output form using only 5 fields, but only the first 5 fields of your structure are output. Although you can do it this way, it's always better to have a direct structure/form linkup to eliminate confusion.

9.5. EXAMPLE OF ESCORT STRUCTURE/FORM LINKUP

Example structure

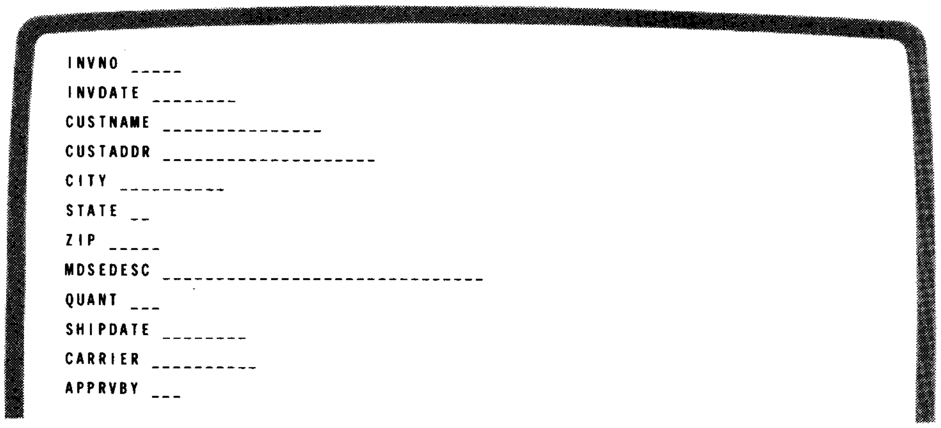
Let's assume we have the following structure (named SHIPFILE) for a shipping file:

STRUCTURE OF SHIPFILE

INVNO	5N
INVDAT	8A
CUSTNAME	15A
CUSTADDR	20A
CITY	10A
STATE	2A
ZIP	5N
MDSEDESC	30A
QUANT	3N
SHIPDATE	8A
CARRIER	10A
APPRVBY	3A

When this structure is output in the standard default screen format, it looks like this:

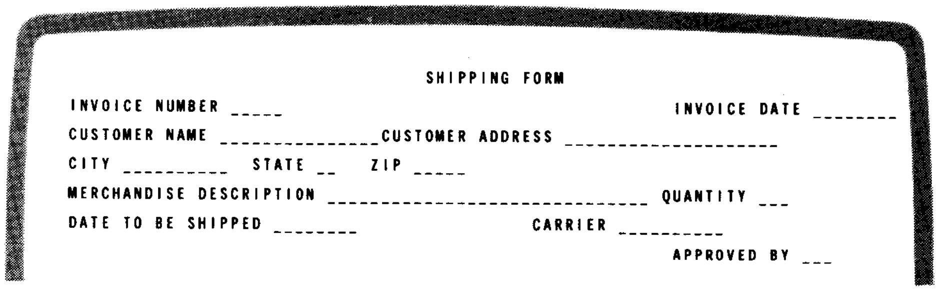
Default screen format for example structure



```
INVNO _____
INVDATE ________
CUSTNAME _______________
CUSTADDR ____________________
CITY __________
STATE __
ZIP _____
MDSEDESC ______________________________
QUANT ___
SHIPDATE ________
CARRIER __________
APPRVBY ___
```

Let's create a form (named SHIPFORM) to link with this structure, and you'll see how things change:

Example form



```
                                  SHIPPING FORM
INVOICE NUMBER _____                                          INVOICE DATE ________
CUSTOMER NAME _______________CUSTOMER ADDRESS ____________________
CITY __________  STATE __  ZIP _____
MERCHANDISE DESCRIPTION ______________________________ QUANTITY ___
DATE TO BE SHIPPED ________                  CARRIER __________
                                                          APPROVED BY ___
```

Now let's show how you'd designate this in a program. Since it's a shipping form, let's use it for input with an ENTER DATA program. To enter data onto this form into a file called SHIPFILE, your program reads:

Program example

```
ENTER DATA FROM WS (SHIPFILE,SHIPFORM)
CREATING SHIPFILE
```

When you execute the program, your form (SHIPFORM) then appears and you key in your data.

For more on screen formatting and the uses of screen forms, see the screen format services concepts and facilities manual.

9.6. SPECIAL ESCORT CONSIDERATIONS FOR FORMS CREATION

Depending on the type of ESCORT program you use with a form (ENTER DATA, CHANGE DATA, SELECT DATA), the following requirements apply:

ENTER DATA rules

- ENTER DATA

  Specify all fields in the form as input (I).

CHANGE DATA rules

- CHANGE DATA

  Specify all fields in the form as both input and output (B).

  Protect the decimal point in decimal numbers by using slashes on both sides of the decimal point (7/./8).

  You cannot use packed decimal fields in a form. The initial display does not display underlines for the packed field.

  When the form is first displayed, only the key field is read. ESCORT ignores all other data until the master record is displayed and changed.

SELECT DATA rules

- SELECT (display) DATA

  Specify all fields as both input and output (B) or as output only (O). When you display data using only output fields, you must define at least one input field to hold the screen until the **XMIT** key is pressed. ESCORT ignores any data sent from this input field (B or I).

  Regardless of the program type, when you output a print form, you are limited to 80 printer positions.

9.7. GUIDELINES FOR PRINTING CUSTOM FORMS WITH ESCORT

These guidelines apply when using the OUTPUT TO PRINTER clause with a screen format form:

- The number of lines printed for each form is exactly the same as the number of lines in the screen format generator form definition. If the form crosses a page boundary, a skip to top of page is done when the overflow line is reached. (The overflow line is defined by a SYSGEN parameter - the default is 55 lines per page.)
- You can control vertical spacing on the page by changing $LINE and using the OUTPUT TO PRINTER clause without a form to print a blank line.
- $LINE in ESCORT is not incremented when a form is used. To cause page ejects, you must include a variable and code for counting lines or forms per page. When the limit for lines per page or forms per page is reached, the program should set $LINE equal to 1 and issue the OUTPUT TO PRINTER (BLANK) clause to print a blank line at the top of the next page.

Example 1

The system-defined number of lines per page is 55. This example has an 11-line screen form with blank lines at the top and bottom.

In this case, no special processing is required. Five forms are printed on each page; the blank lines in the form control the vertical spacing.

Sample program:

```
SELECT DATA OF MASTER
OUTPUT TO PRINTER (DATA, FORM11)
```

Example 2

In this example, each time the form begins, it is printed on a new page. The ejects occur as the result of an OUTPUT TO PRINTER clause of a line that contains all blanks.

Sample program:

```
SELECT DATA OF MASTER
   COMPUTING $LINE=1
   OUTPUT TO PRINTER (BLANK)
   OUTPUT TO PRINTER (DATA, FORM)
```

Example 3

This example shows a form with eight lines. Four copies of the form are printed per page with five blank lines between each form.

In this example, control vertical spacing by changing $LINE and printing a blank line. Control ejects by incrementing a user-defined form counter.

Sample program:

```
SELECT DATA FROM MASTER
   WORKAREA (COUNTER)
   IF FIRSTIME
      COMPUTING $LINE   = 1,
                COUNTER = Ø
      END
   OUTPUT TO PRINTER (BLANK)
   OUTPUT TO PRINTER (DATA, FORM8)
   COMPUTING
      $LINE   = $LINE+4,
      COUNTER = COUNTER+1
   IF COUNTER = 4
      COMPUTING
      $LINE = 1
      COUNTER = Ø
      END
```

An example using custom structures to custom-format output is shown in Appendix G.

Appendix A. Program Statement Formats

A.1. CHANGE DATA PROGRAM STATEMENT

```
CHANGE DATA OF master-file [structure]
   ┌FROM┌WS┌┌structure      ┐┐ ┐┐
   │    │  └└structure,form ┘┘ ││
   │    │                      ││
   └    └input-file [structure]┘┘
[USING input-file-1 [structure],...,input-file-5 [structure]]
WORKAREA structure [data replacement statements]
SUBTOTALING fieldnames [''literal expressions'']
TOTALING fieldnames [''literal expressions'']
IF┌conditions┐ clauses [ELSE clauses] END
  │FIRSTIME  │
  └ENDOFILE  ┘
COMPUTING data replacement statements
WHILE conditions clauses END
CLEARING fieldnames
PRINT┌┌fieldnames [''literal expressions'']┐┐
     ││SUBTOTALS                           ││
     ││TOTALS                              ││
     └└structure                           ┘┘
DISPLAY┌┌fieldnames [''literal expressions'']┐┐
       ││SUBTOTALS                           ││
       ││TOTALS                              ││
       └└structure                           ┘┘
OUTPUT TO┌WS     ┐┌structure      ┐
         └PRINTER┘└structure,form ┘
EXIT
```

A.2. DELETE DATA PROGRAM STATEMENT

```
DELETE DATA OF master-file [structure]
 ⎧FROM⎧WS ⎡⎧structure      ⎫⎤ ⎫⎫
 ⎨    ⎨   ⎣⎩structure,form ⎭⎦ ⎬⎬
 ⎨    ⎩input-file[structure]  ⎭⎬
 ⎩IF conditions                ⎭
```

A.3. ENTER DATA PROGRAM STATEMENT

```
ENTER DATA FROM WS⎡⎧structure      ⎫⎤
                  ⎣⎩structure,form ⎭⎦

WORKAREA structure [data replacement statements]

SUBTOTALING fieldnames [''literal expressions'']

TOTALING fieldnames [''literal expressions'']

IF⎧conditions⎫clauses [ELSE clauses] END
  ⎨FIRSTIME  ⎬
  ⎩ENDOFILE  ⎭

COMPUTING data replacement statements

WHILE conditions clauses END

CLEARING fieldnames

CREATING output-file [structure]

EXTENDING output-file [structure]

PRINT⎡⎧fieldnames [''literal expressions'']⎫⎤
     ⎢⎨SUBTOTALS                           ⎬⎥
     ⎢⎨TOTALS                              ⎬⎥
     ⎣⎩structure                           ⎭⎦

DISPLAY⎡⎧fieldnames[''literal expressions'']⎫⎤
       ⎢⎨SUBTOTALS                          ⎬⎥
       ⎢⎨TOTALS                             ⎬⎥
       ⎣⎩structure                          ⎭⎦

OUTPUT TO⎧WS     ⎫⎧structure      ⎫
         ⎩PRINTER⎭⎩structure,form ⎭

EXIT
```

A.4. SELECT DATA PROGRAM STATEMENT

```
SELECT DATA OF master-file [structure]
  ┌FROM ⎧WS ⎡⎧structure     ⎫⎤ ⎤
  │     ⎨   ⎣⎩structure,form⎭⎦ ⎬
  └     ⎩input-file [structure]⎦⎭
[USING input-file-1 [structure],..., input-file-5 [structure]]
WORKAREA structure [data replacement statements]
SUBTOTALING fieldnames [''literal expressions'']
TOTALING fieldnames [''literal expressions'']
IF⎧conditions⎫clauses [ELSE clauses] END
  ⎨FIRSTIME  ⎬
  ⎩ENDOFILE  ⎭
COMPUTING data replacement statements
WHILE conditions clauses END
CLEARING fieldnames
CREATING output-file [structure]
EXTENDING output-file [structure]
UPDATING input-file
OUTPUT TO⎧WS     ⎫⎧structure     ⎫
         ⎩PRINTER⎭⎩structure,form⎭
PRINT⎡⎧fieldnames [''literal expressions'']⎫⎤
     ⎢⎨SUBTOTALS                           ⎬⎥
     ⎢⎨TOTALS                              ⎬⎥
     ⎣⎩structure                           ⎭⎦
DISPLAY⎡⎧fieldnames [''literal expressions'']⎫⎤
       ⎢⎨SUBTOTALS                           ⎬⎥
       ⎢⎨TOTALS                              ⎬⎥
       ⎣⎩structure                           ⎭⎦
EXIT
```

A.5. SORT DATA PROGRAM STATEMENT

```
SORT DATA OF master-file [structure] [BY field-1,...,field-6]
  CREATING output-file [structure]
```

Appendix B. Reserved Words

Definition

Value in sorts and comparison

Method of writing

All words used in ESCORT statements are reserved words. You can't use them as file or field names, but you can use them as alphanumeric constants or in data you input or output. You can write reserved words in upper or lowercase, but when sorting or comparing alphanumeric data fields, lowercase letters have greater value than their uppercase counterparts. You must leave at least one space or special character between each reserved word.

AND

BY

CHANGE
CLEARING
COMPUTING
CONTAIN
CONTAINS
CREATING

DATA
DELETE
DISPLAY
DOES

ELSE
END
ENDOFILE
ENDS
ENTER
EQ
EXIT
EXTENDING

FIRSTIME
FROM

GE
GT

IF

LE
LT

NE
NOT

OF
OR
OUTPUT

PRINT
PRINTER

SELECT
SORT
START

STARTS
SUBTOTALING
SUBTOTALS

TO
TOTALING
TOTALS

UPDATING
USING

WHILE
WITH
WORKAREA
WS

$DATE
$DAY
$LINE
$MONTH
$PAGE
$TIME
$TITLE
$YEAR

Appendix C. Function Key Use

There are four processing modes in ESCORT:

Processing modes

- Structure processor, which processes your structures
- Program mode, which processes the programs you write
- Tutorial mode, which processes the tutorial programs you write
- Run-time processor, which executes your programs

Table C-1 lists the function keys and their actions under the different processors.

Table C-1. Function Key Use

Key	Structure Processor	Program Mode	Tutorial Mode	Run-Time Processor
F1	End of input, return to master menu	Not used	Not used	Not used
F2	Cancel display output	Not used	Present a menu of subjects for which tutorial mode HELP is available	Not used
F3	Not used	Cancel current screen and return to previous screen	Abort current tutorial processing and return to beginning of tutorial mode	Not used
F4	Abort structure and return to menu	Terminate free-form input and return to previous menu*	Abort tutorial processing and return to command mode	Terminate data input or workstation display and return to caller**

* The XMIT key (instead of F4) in program mode will also terminate input and return to the previous menu.

** Caller refers to the ESCORT processor (program or tutorial mode) that called for a run-time function (i.e., program or job execution). If you're in tutorial mode, you'll return to tutorial mode. If you're in program mode, you'll return to the PROGRAM COMMANDS menu.

Appendix D. Programming Considerations and System Guidelines

D.1. ARITHMETIC CONVENTIONS

- Constants

Definition

A constant is data whose value doesn't change during program execution. ESCORT allows alphanumeric, numeric, and decimal constants. The following rules apply:

Types allowed

- Alphanumeric

  Enclose entire constant in quotation marks, e.g., "BIRD"

- Numeric and Decimal

  Arithmetic signs are permitted, but two signs cannot appear together. When they must, enclose the constant and its sign in parentheses, e.g., A=B*(-3).

- Arithmetic Operators

Definition

The arithmetic operators are:

+	Addition
-	Subtraction
*	Multiplication
/	Division
:	Modulo (a division operation where the result is the remainder portion only)

Operation hierarchy

In arithmetic expressions, multiplication, division, and modulo are performed before addition and subtraction.

The sequence for performing operations of the same level is left to right.

$$5 + 3 * 4 - 2 + 10 = 25$$

Use of parentheses

In this example, the first operation performed is 3 * 4. Then 5 is added to the product, 2 is subtracted, and 10 is added.

Expressions enclosed within parentheses are evaluated first. When nested parentheses are used, evaluation begins with the innermost set and works outward.

$$((5+3)*4-(2+10))=20$$

D.2. KEY MATCHING IN A MULTIKEY ENVIRONMENT

By default, the first key specified in a structure becomes the key of reference (i.e., the first key specified, not the lowest key number). Access to the file is based on this key. The following examples show how ESCORT accesses files using different key positions.

Example 1

STRUC1

FIELD1	4A	K1
FIELD2	4A	K2
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	K3

Program

```
SELECT DATA OF MASTERFIL (STRUC1) DISPLAY
```

In example 1, MASTERFIL is accessed sequentially by key 1 (field FIELD1) and the records are displayed accordingly.

Example 2

STRUC2

FIELD1	4A	
FIELD2	4A	K2
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	K3

Program

```
SELECT DATA OF MASTERFIL (STRUC2) DISPLAY
```

In example 2, MASTERFIL is accessed sequentially by key 2 (field FIELD2) and the records are displayed accordingly.

You can override the default key of reference by establishing key matching criteria between a master file and input file. The overriding of the default occurs once, creating a hierarchical relationship among input files matching the master file. Files or workstation keys in the FROM designation take precedence over all references of USING files. Likewise, each USING file takes precedence over the following USING file in establishing the master key of reference.

Key matching involves finding the key field in the input structure and finding the identical field in the master file. In the case of matching between the FROM input file and the master file (i.e., SELECT DATA OF MASTERFIL (STRUC1) FROM INPUT (STRUC3)), the matching field in the master file must be a key. Matching between secondary input and master file does not require that the matching master file field be a key.

In example 3, the secondary input file changes the key of reference from the default key (key 1) to key 3.

Example 3

STRUC1

FIELD1	4A	K1
FIELD2	4A	K2
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	K3

STRUC3

FIELD1	4A	
FIELD2	4A	
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	K1

Program

```
SELECT DATA OF MASTERFILE (STRUC1) USING SECINPUT (STRUC3) DISPLAY
```

Key matching is established between field FIELD5 in STRUC3 and field FIELD5 in STRUC1 by selecting the key field in the input structure and finding the matching field in the master file. Based on field FIELD5 as the matching field, MASTERFILE is read, the value in FIELD5 is used as the key in reading the file SECINPUT.

In example 3, if the structure STRUC1 was changed deleting key 3 and the program recompiled, the processing will change.

Example 4

STRUC4

FIELD1	4A	K1
FIELD2	4A	K2
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	

STRUC3

FIELD1	4A	
FIELD2	4A	
FIELD3	4A	
FIELD4	4N	
FIELD5	4N	K1

Program

```
SELECT DATA OF MASTERFIL (STRUC4) USING SECINPUT (STRUC3) DISPLAY
```

Again, key matching is established between field FIELD5 in STRUC3 and field FIELD5 in STRUC4. However, because symbol FIELD5 in the master file is not a key, overriding of the key does not take place. Key 1 (field FIELD1) is the key of reference by default. The master file is read sequentially by key FIELD1. After the master file is read, the value in FIELD5 is used as the key in the reading of the file SECINPUT. The access to SECINPUT could change if the way the master file is read changes.

D.3. USES OF THE $LINE SYSTEM UTILITY FIELD

- With a user-supplied format, structure only

  ESCORT clause: OUTPUT TO PRINTER (structure-name)

  The $LINE system utility field is most effectively used this way. Line control by the ESCORT program becomes effective when $LINE is set to some non-zero value. When $LINE is set before an OUTPUT TO PRINTER clause, ESCORT controls the printer output based on the current line position as compared to the value of $LINE.

  1. If $LINE is the same as the current printer line position, the line is printed and $LINE is incremented by one.

  2. If $LINE is greater than the current line position, the printer is advanced to $LINE, the line is printed, and $LINE is incremented by one.

  3. If $LINE is less than the current line position, the printer is skipped to the next page and advanced to $LINE. Then the line is printed, and $LINE is incremented by one.

- With a user-supplied format, structure and form

  ESCORT clause: OUTPUT TO PRINTER (structure-name,formname)

  The number of lines printed is controlled by the screen format coordinator, not by ESCORT. Therefore, the $LINE system utility field can only be used to cause page ejects. This is done by setting $LINE to a value greater than the number of lines that will fit on a page (e.g., $LINE = 67 if page size is 66).

- With automatic forms

  ESCORT clause: PRINT

  The $LINE system utility field has no effect on automatic forms.

D.4. USING A DATA FILE WRITTEN IN ANOTHER LANGUAGE

ESCORT can access files written in other languages, such as COBOL, BASIC, FORTRAN, and RPG II. The only restriction is that the files you want to access from ESCORT must be MIRAM files.

Call these files the same way you call an ESCORT file. When ESCORT prompts you for a file name, just key in the name of any cataloged file written in another language. ESCORT permits file names of up to 44 characters. The first character must be alphabetic.

When an MS-type file is created with COBOL, ORGANIZATION IS RELATIVE should be specified for the file. If ORGANIZATION IS SEQUENTIAL is specified and an attempt is made to access the records out of order, a data management error occurs, and execution of the ESCORT program terminates.

D.5. ESCORT DATA FILE SHARING GUIDELINES

File usage in an ESCORT program determines the type of OPEN generated. Basically, ESCORT uses four modes.

1. SRD – Read mode, allowing other readers.
2. EXC – Exclusive write, allowing no other users. The file is initialized.
3. EXCR – Update mode, allowing read users but no other update.
4. SADD – Update (extend) mode, allowing other users in the same mode.

The following program uses five files and illustrates all modes:

```
SELECT DATA OF PRCMASTER
FROM PRICE
USING PRCREF
COMPUTING TOTALPRC = PRICE * NUMITEMS
IF NUMITEMS = 0
CREATING PRCERROR
ELSE
EXTENDING PRCREC
UPDATING PRCMASTER
END
```

File	Access Mode	Clause
PRCMASTER	EXCR	UPDATING
PRICE	SRD	FROM
PRCREF	SRD	USING
PRCERROR	EXC	CREATING
PRCREC	SADD	EXTENDING

The object file of the DELETE or CHANGE verb is treated the same as a file used in an UPDATING clause.

Violation of the file sharing guidelines of the system will result in display of an error message.

D.6. ESCORT SORTING SEQUENCE

The sequence of the records in the output file of an ESCORT sort program is governed by the following rules:

1. The sort key is treated as an unsigned bit string. Therefore, numeric data (packed and decimal) will not be sorted according to algebraic sign.

2. The collating sequence for character strings is determined by the internal machine representation of the character set.

D.7. ESCORT LIBRARY MODULE TYPES

The ESCORT library is a MIRAM library. Five types of modules are included:

Type	Description
E$CS	Source language program
E$CO	Object program
E$AF	Created in conjunction with the object program when automatic formatting is used
E$CJ	Job
E$FD	Structure

If ESCORT object modules are copied, renamed, or deleted by a system processor other than ESCORT, the corresponding E$AF module must also be copied, renamed, or deleted.

Appendix E. Error Messages

This section contains the error messages you may receive in the course of your ESCORT processing:

- Compiler syntax errors
- Compiler errors
- Structure errors
- Program mode errors
- Tutorial mode errors
- Run-time errors

Most error messages tell what caused the error and how to remedy it.

The structure error messages tell only what caused the error. When a structure error occurs, a menu appears listing various options to remedy the error. Because the remedy is your choice, we've not listed the remedies here.

ESC065 WARNING - NAME EXCEEDS 8 CHARACTERS

A field name is longer than eight characters. The compiler truncates the field name at eight characters.

If the field name is unique in the first eight characters, your program will be correct. If not, shorten the field name.

ESC066 WARNING - 'END' CLAUSE MISSING

During processing of an IF or WHILE clause, the end of the program was reached, or during processing of a nested IF or WHILE clause, the ELSE portion of the outer IF clause was encountered. The compiler assumes that the clause being processed is to end at this point.

If the assumption is correct, your program will be correct. Otherwise, insert the missing END clause at the appropriate place in the program.

ESC067 INVALID UTILITY FIELD

A field name begins with a $ but is not the name of an ESCORT utility field.

Correct the field name in error and attempt to recompile the program.

ESC068 NUMBER IS TOO LARGE

A number contains more than 15 digits or more than 9 decimal places.

Correct the program and attempt to recompile it.

ESC069 MORE THAN 700 SYMBOLS USED

More than 700 different field names are used in the program.

Reduce the number of field names used in the program. Attempt to recompile the program.

ESC070 INVALID KEYWORD FOUND

The first word of the program is not ENTER, CHANGE, SELECT, SORT, or DELETE. Or these program statements are encountered somewhere other than the first position in a program.

Correct the program and attempt to recompile it.

ESC072 INVALID FILE NAME

This program is caused by one of the following:

1. The file name is WS and the clause is not FROM or OUTPUT TO.
2. Something other than WS or PRINTER is used in an OUTPUT TO clause.
3. Something other than WS is used in the FROM clause with the ENTER DATA program statement.

Correct the program and attempt to recompile it.

ESC073 COMPILER TABLE CAPACITY EXCEEDED

The program references more structures, symbols, or literals than the compiler table can store. The structure, symbol, or literal that caused the error is indicated by where the message appears in the program.

Restructure the program to remove excess structures, symbols, or literals.

ESC074 INVALID CLAUSE FOUND

This error could be caused by:

1. A misspelled clause
2. A syntax error
3. Punctuation of a clause

Correct the program and attempt to recompile it.

ESC075 INVALID STRUCTURE NAME

The structure name given is not a valid structure name. Either it does not exist as written or no structure name is specified.

Correct the program and attempt to recompile it.

ESC076 BAD DELIMITER FOUND

A right parenthesis is required but another delimiter was found.

Correct the program and attempt to recompile it.

ESC077 INVALID FORM NAME

Something other than a form name is present where a form name is expected.

Correct the program and attempt to recompile it.

ESC078 CONSTANT EXPECTED

A data replacement statement in a WORKAREA clause attempted to assign a value other than a constant to some variable. Data replacement statements in a WORKAREA clause are used only to assign constants as initial values of a WORKAREA variable.

Delete or correct the invalid data replacement statement. Attempt to recompile the program.

ESC079 NO CLAUSE FOR OUTPUT (E.G., CREATING)

An ENTER DATA or SELECT DATA program produces no output.

Include one of the following output clauses in the program: CREATING, EXTENDING, DISPLAY, PRINT, OUTPUT TO, or UPDATING (used only with the SELECT DATA program statement). Attempt to recompile the program.

ESC080 CREATING CLAUSE REQUIRED

The CREATING clause is missing from the SORT DATA program statement, or a clause other than BY precedes the CREATING clause.

Correct the program and attempt to recompile it.

ESC081 INVALID ALPHANUMERIC LITERAL

An alphanumeric literal was not enclosed by a quotation mark or two quotation marks appeared with nothing between them.

Correct the program and attempt to recompile it.

ESC082 '=' EXPECTED

A data replacement statement used in a COMPUTING or WORKAREA clause does not have an equal sign where one should appear.

Correct the program and attempt to recompile it.

ESC083 FIELD NAME EXPECTED

The BY clause of the SORT DATA program statement contains something other than a field name.

Correct the program and attempt to recompile it.

ESC084 INVALID CONDITION

A condition specified in an IF or WHILE clause is invalid.

Correct the program and attempt to recompile it.

ESC085 FILENAME EXCEEDS 44 CHARACTERS

The maximum length for a file name is 44 characters.

Correct the program and attempt to recompile it.

ESC086 BAD OPERAND IN CHARACTER RELATION

The first operand of a character relation (character string operand) is something other than a field name.

Correct the program and attempt to recompile it.

ESC087 BAD CONDITIONAL OPERATOR

Something other than a conditional operator (AND or OR) or a reserved word was encountered in an IF or WHILE conditional expression.

Correct the offending conditional expression. Attempt to recompile the program.

ESC088 ERROR IN XXXXXXXXXXXXXXXXXXXXXXXXX CLAUSE

The error is in one of the following clauses:

1. CLEARING - The CLEARING clause must contain a list of the field names to be cleared. This error occurs if something other than a field name is present.

2. COMPUTING - The COMPUTING clause must contain at least one data replacement statement. This error is caused by lack of or erroneous data in a data replacement statement.

3. DISPLAY or PRINT - A PRINT or DISPLAY clause contained an invalid specification. The only legal parameters are: SUBTOTALS, TOTALS, structure-name, and fieldnames and literal expressions.

4. SUBTOTALING or TOTALING - The SUBTOTALING or TOTALING clause contains something other than a field name, a system utility field, or a literal expression.

Correct the program and attempt to recompile it.

ESC089 FORM SPECIFICATION NOT ALLOWED

Use of the formname parameter is invalid. The formname parameter following a structure name is valid only with:

1. The OUTPUT TO clause

2. The FROM WS clause of the ENTER DATA, CHANGE DATA, and SELECT DATA program statements

Correct the program and attempt to recompile it.

ESC090 INVALID ARITHMETIC FACTOR

An invalid arithmetic factor was encountered (e.g., assigning an alphabetic value to a numeric field, using a literal expression, etc).

Correct the invalid factor. Attempt to recompile the program.

ESC091 PROGRAM EXCEEDS COMPILER TABLE SIZE

The program being compiled generated more code than can be handled in the compiler table area.

Reduce the size of the program. Attempt to recompile the program.

ESC092 STRUCTURE SPECIFICATION MISSING

The OUTPUT TO or WORKAREA clause does not have a specified structure, or the left parenthesis is missing from the structure specification.

Correct the program to include the missing structure. Attempt to recompile the program.

ESC093 INVALID FIELD NAME QUALIFIER

This error could be caused by:

1. A qualifier name longer than 44 characters

2. No qualifier is named even though OF is present

3. The qualifier name does not match the name given in the OF, FROM, or USING clauses.

Correct the program and attempt to recompile it.

ESC094 INVALID SUBSCRIPT

An error was encountered in a subscript (array) expression.

Correct the program and attempt to recompile it.

ESC095 ENDOFILE OR FIRSTIME IN NESTED IF

The ENDOFILE or FIRSTIME system operators must appear in an initial IF clause. These operators cannot be embedded within a nested IF. See the IF clause for usage rules.

Correct the program and attempt to recompile it.

ESC096 EXTRA XXXXXXXXXXXXXXXXXXXXXXXXX CLAUSE FOUND

The program contains more than one SUBTOTALING, TOTALING, or WORKAREA clause. Only one of each is allowed.

Correct the program and attempt to recompile it.

ESC097 SUBSCRIPT LITERAL NOT INTEGER

A number in a subscript (array) contains an illegal decimal. All numbers in subscripts must be integers.

Correct the subscript to an integer number. Attempt to recompile the program.

ESC098 INVALID ARITHMETIC EXPRESSION

Something was encountered that is illegal in an arithmetic expression (e.g., an invalid separator, parentheses with no value enclosed, etc).

Correct the invalid expression. Attempt to recompile the program.

ESC099 DELETE CRITERIA MISSING

The IF or FROM clause is missing from the DELETE DATA program statement, or an invalid clause precedes the IF or FROM clause.

Correct the program and attempt to recompile it.

ESC100 EXTRANEOUS INPUT ENCOUNTERED

A SORT DATA or DELETE DATA program statement contains more clauses than are allowed.

Correct the program and attempt to recompile it.

ESC101 IF, WHILE CLAUSES NESTED MORE THAN nn LEVELS

The program contains nested IF or WHILE clauses that exceed the storage capacity of the compiler.

Reduce the number of nested IF or WHILE clauses. Attempt to recompile the program.

ESC102 MORE THAN 12 FILES USED

More than 12 different file names, other than those used in the OUTPUT TO clause, appear in the program.

Reduce the number of files. Attempt to recompile the program.

ESC103 ONLY ONE XXXXXXXXXXXXXXXX ALLOWED

The program contains more than one IF ENDOFILE or IF FIRSTIME clause.

Put all processing statements that are dependent on these conditions under a single IF ENDOFILE or IF FIRSTIME condition. Attempt to recompile the program.

ESC104 SUBSCRIPTS MAY BE NESTED ONLY 2 LEVELS

Subscripted (array) fields in a subscripted expression are nested to a depth greater than 2.

Simplify the subscripted expression. Attempt to recompile the program.

ESC105 XXXXXXXXXXXXXXXX NOT PRINTED

The program contains a TOTALING or a SUBTOTALING clause but no corresponding PRINT or DISPLAY TOTALS or SUBTOTALS clause.

Insert the applicable PRINT or DISPLAY clause. Attempt to recompile the program.

ESC106 XXXXXXXXXXXXXXXX PRINTED, BUT NOT SPECIFIED

The program contains a PRINT TOTALS or DISPLAY TOTALS clause, but no TOTALING clause was specified. Or, the program contains a PRINT SUBTOTALS or DISPLAY SUBTOTALS clause, but no SUBTOTALING clause was specified.

Insert the applicable TOTALING or SUBTOTALING clause. Attempt to recompile the program.

ESC107 ILLEGAL CHARACTER FOUND

There is a character in the program which is neither alphanumeric or a valid ESCORT symbol.

Correct the program and attempt to recompile it.

ESC108 EXPRESSIONS MAY BE NESTED ONLY 20 LEVELS

Parentheses are nested to a depth greater than 20 in an expression.

Simplify the expression. Attempt to recompile the program.

COMPILER SYNTAX ERROR MESSAGES

ESC109 STRUCTURE NOT SPECIFIED FOR WS

No structure was specified for workstation input in an ENTER DATA program that does not include a CREATING or EXTENDING clause.

Specify a structure for use with workstation input or include a CREATING or EXTENDING clause. Attempt to recompile the program.

ESC110 ENDOFILE INVALID WITHOUT A FILE

An ENDOFILE condition was encountered in an ENTER DATA program that does not include a CREATING or EXTENDING clause.

Remove the ENDOFILE condition or add a CREATING or EXTENDING clause. Attempt to recompile the program.

ESC111 UPDATE STRUCTURE MUST MATCH INPUT

The structure used for the file named in an UPDATING clause is not the same as the structure used for another occurrence of the file.

Specify in the UPDATING clause the same structure used for the file in the OF, FROM, or USING clause. Attempt to recompile the program.

ESC112 MORE THAN 6 SORT FIELDS

Only six sort fields may be specified in the BY clause.

Correct the program and attempt to recompile it.

ESC113 NO BUFFER AVAILABLE FOR COMPILATION

The ESCORT compiler attempted to acquire a buffer from the system, but none was available.

Attempt to compile the program when the system is not so busy.

ESC114 FILE USE CONFLICTS WITH PREVIOUS USE

The file name appears in a CREATING clause and another type of clause, or the file name appears in an EXTENDING clause and another type of clause.

Correct the program and attempt to recompile it.

ESC115 UPDATE FILE MUST BE USED FOR INPUT

The file name appears in an UPDATING clause, but not in an OF, FROM, or USING clause.

Correct the program and attempt to recompile it.

ESC116 ILLEGAL XXXXXXXXXXXXXXXXXXXXXXXX CLAUSE

The error is in one of the following clauses:

1. ELSE – There is no matching IF preceding this clause.
2. END – There is no matching IF or WHILE preceding this clause.
3. CREATING – This clause is used only with the ENTER DATA, SELECT DATA, and SORT DATA program statements.
4. EXTENDING – This clause is used only with the ENTER DATA and SELECT DATA program statements.
5. UPDATING – This clause is used only with the SELECT DATA PROGRAM statement.

Correct the program and attempt to recompile it.

ESC120 ITEM IS NOT A FIELD NAME OR AN ALPHA LITERAL

One of the following clauses was used incorrectly:

1. TOTALING, SUBTOTALING – Only field names and alphanumeric literals may be used with these clauses.
2. PRINT, DISPLAY – A field name or alphanumeric literal was expected, but something else was encountered, or the clause was followed by an invalid clause.

ESC128 SYMBOL XXXXXXXX NOT DEFINED IN STRUCTURE

The field name shown is not defined in an input or WORKAREA structure.

Correct the program and attempt to recompile it.

ESC129 STRUCTURE XXXXXXXX NOT FOUND

The specified structure does not exist on either the session workfile or the ESCORT library file.

Check to make sure you are using a valid structure name. Correct the program and attempt to recompile it.

ESC130 ALPHANUMERICS NOT ALLOWED IN NUMERIC COMPUTING

A numeric computing clause may not include specification of alphanumeric fields and literals.

Correct the offending clause. Correct the program and attempt to recompile it.

ESC131 INPUT FILE MUST HAVE ONE/ONLY ONE KEY

All input files are allowed only one key.

Correct the program and attempt to recompile it.

ESC132 INVALID INPUT MASTER MATCH

The input key for SELECT does not have a matching symbol key value in the master file.

Correct the program and attempt to recompile it.

ESC133 MU MASTER INVALID FOR INPUT/MASTER MATCH

The master file must be sorted for input/master matching.

Correct the program and attempt to recompile it.

ESC134 INVALID SUBSCRIPT FOR THIS CLAUSE

Only simple subscripts may be used with PRINT or DISPLAY clauses. For example: A(B) is valid; A(B(C)) and A(B+C) are invalid.

Correct the program and attempt to recompile it.

ESC135 COMPUTING INVALID FOR THIS CLAUSE

No computation may be done with field names included in PRINT or DISPLAY clauses.

Correct the program and attempt to recompile it.

ESC136 CODE GENERATION OVERFLOW

Your code has exceeded the allowable limit.

Restructure the program so the code stays within the limits.

ESC137 STRUCTURE TABLE OVERFLOW

The number of fields defined in the program has caused the structure table to overflow.

Reduce the number of fields used in the program. Correct the program and attempt to recompile it.

ESC138 AUTOFORM WORKSTATION SIZE EXCEEDED

The physical size of the workstation cannot accommodate the structure size.

Try to reduce the structure size or, if output is involved, make more than one structure with multiple displays.

ESC139 AUTOFORM ARRAY SIZE EXCEEDED

The size of an array element is limited to the length of a line on the printer or workstation with which it is to be used.

Correct the program and attempt to recompile it.

ESC140 AUTOFORM TABLE OVERFLOW

All of the field names and field sizes from all of the PRINT and DISPLAY clauses will not fit in the maximum buffer size.

Reduce the number of PRINT and DISPLAY clauses. Correct the program and attempt to recompile it.

ESC141 INVALID WORKAREA INITIALIZATION

An alphanumeric field is not initialized with an alphanumeric literal, or a packed field is not initialized with a packed literal.

Correct the field in error. Attempt to recompile the program.

COMPILER
ERROR
MESSAGES

ESC142 SYSTEM MEMORY NOT AVAILABLE

ESCORT attempted to acquire a buffer from the system but none was available.

Attempt to compile the program when the system is not so busy.

ESC143 MAXIMUM SORT KEY LENGTH EXCEEDED

The combined length of all specified sort fields exceeds 80 characters.

Reduce the total size of the sort field or use the OS/3 sort.

ESC145 UNABLE TO LOAD MODULE XXXXXXXX, ERROR CODE = NNN

An error has occurred during an attempt to load an ESCORT module.

If the error is known to be fatal to the ESCORT session, the following message is also displayed.

MEMORY MANAGEMENT ERROR REQUIRES IMMEDIATE TERMINATION, XMIT

If recovery of the ESCORT session can be attempted, this message is displayed instead:

PRESS TRANSMIT TO RETURN TO THE COMMAND MENU.

ESC150 ALPHA TO NUMERIC/NUMERIC TO ALPHA IMPLIED MOVE INVALID

You may not move an alphanumeric field to a numeric field or a numeric field to an alphanumeric field during implied moves for CREATING or EXTENDING clauses.

Correct the program and attempt to recompile it.

ESC151 SYMBOL XXXXXXXX SUBSCRIPT MUST BE 'N', 'B', OR 'P' TYPE FIELD

A symbol used as a subscript must identify a numeric, binary, or packed field.

Correct the program and attempt to recompile it.

ESC152 NUMERIC NOT ALLOWED IN ALPHANUMERIC COMPUTING/COMPARE

An alphanumeric clause may not include specification of numeric fields or literals.

Correct the program and attempt to recompile it.

ESC153 SYMBOL XXXXXXXX ALPHANUMERIC TYPE FIELD NOT ALLOWED IN SUBTOTALING TOTALING

Only numeric field types may be specified for SUBTOTALING or TOTALING clauses.

Correct the program and attempt to recompile it.

ESC154 SYMBOL XXXXXXXX SUBTOTALING TOTALING OF NONSUBSCRIPTED ARRAY FIELDS NOT ALLOWED

Only individual elements of arrays may be specified for SUBTOTALING or TOTALING.

Correct the program and attempt to recompile it.

ESC155 MASTER FILE HAS NO KEY FOR WORKSTATION PROMPT

The structure specified for use with the master file must include a key to be used for identifying desired records from the workstation.

Correct the program and attempt to recompile it.

ESC156 SYMBOL XXXXXXXX DECIMAL DIGITS IN PACKED SUBSCRIPT FIELD

Packed fields are allowable for subscripting only when the specified length of the decimal portion of the field is 0. For example: Z specified as 5P0 is valid for use as a subscript; Z1, specified as 5P2 is not.

Correct the program and attempt to recompile it.

ESC157 SYMBOL XXXXXXXX SUBSCRIPT REFERENCE TO NONARRAY SYMBOL ENTRY

Only elements of arrays may be referenced by subscripting.

Correct the program and attempt to recompile it.

ESC158 SECONDARY INPUT KEY HAS NO MATCHING SYMBOL IN INPUT/MASTER

The symbol used for a secondary input file key must be defined in the master or input file structure.

Correct the program and attempt to recompile it.

ESC159 VALID ITEMS FOR THIS CLAUSE ARE ITEMS AND LITERAL EXPRESSIONS

This clause may include only field names from a structure and literal expressions.

Correct the program and attempt to recompile it.

ESC160 SYMBOL XXXXXXXX ALPHA FIELD NOT ALLOWED IN COMBINATION WITH UNARY VALUES

Only numeric type fields may be specified with unary values.

Correct the program and attempt to recompile it.

ESC161 SYMBOL TABLE/LITERAL TABLE LIMITS EXCEEDED

Too many symbols or literals have been specified in this program.

Correct the program and attempt to recompile it.

ESC162 SYMBOL XXXXXXXX WORKAREA INITIALIZATION FIELD NOT IN WORKAREA STRUCTURE

1. Only fields defined in the structure specified for the WORKAREA may be initialized in the WORKAREA; or

2. A field specified in the WORKAREA structure is also defined in the master or input file structure.

Correct the program and attempt to recompile it.

ESC163 SYMBOL XXXXXXXX ARRAY ENTRY IS NOT SUBSCRIPTED

References to fields defined as arrays must include subscripts to identify individual array elements.

Correct the program and attempt to recompile it.

ESC164 SYMBOL XXXXXXXX REPEAT COUNT NOT COMPATIBLE FOR IMPLIED MOVE

When creating an input file, the repeat counts must be compatible between input and output fields.

Correct the program and attempt to recompile it.

ESC165 SYMBOL XXXXXXXX DECIMAL DIGITS IN 'N' or 'B' TYPE WORKAREA INITIALIZATION

Only integer values are valid in numeric and binary type fields.

Correct the program and attempt to recompile it.

ESC166 INVALID SUBSCRIPT IN WORKAREA INITIALIZATION

The subscripted WORKAREA initialization is invalid. For example: A(1)=1, A(2)=2, and A(3)=3 are valid. But A(I)=5 and A(2*3)=4 are invalid.

Correct the program and attempt to recompile it.

ESC167 *WARNING* OUTPUT TO PRINTER OVERLAP NUMERIC FIELDS ARE EXPANDED

The OUTPUT TO PRINTER clause is used to customize printer output. When data is moved from the input files to the output buffer, the fields are edited via the structure edit codes. However, the positions are not changed so as not to disturb the user form positioning. If the fields now overlap, this message is displayed.

To avoid the problem, allow for the edited (expanded) field sizes in creating the printer output structure. Attempt to recompile the program.

ESC168 WORKSTATION INPUT DOES NOT PROCESS REDEFINE FIELDS

A structure to be used for workstation input may not include redefined fields on the key if the key is to be used as a record prompt.

Correct the program and attempt to recompile it.

ESC169 NO SORT KEY, USE BY CLAUSE OR MS KEY ON OUTPUT STRUCTURE

No sort key was specified. Sort keys must be identified by using the BY clause with fields in the master file structure, or by designating an MS field on the output file structure.

Correct the program and attempt to recompile it.

STRUCTURE

ERROR MESSAGES

ESC199 STRUCTURE DOES NOT EXIST ON SPECIFIED FILE

An attempt was made to delete a structure that does not exist on the file specified.

ESC200 LINE nn DUPLICATE FIELD NAME

The field name already appears in the structure.

ESC201 LINE nn ILLEGAL LENGTH FOR ALPHANUMERIC FIELD

1. An illegal character was used in the length.
2. An embedded blank was found in the length.
3. The length of an alphanumeric field is zero.
4. The length of an alphanumeric field is greater than 4092.

ESC202 LINE nn LENGTH OF FIELD CANNOT BE ZERO

The sum of the integer and decimal portion of a P field or a V field totals zero.

ESC203 LINE nn ILLEGAL DECIMAL LENGTH FOR 'V' TYPE DATA

V type data must have a decimal portion greater than zero, but less than or equal to 9.

ESC204 LINE nn ILLEGAL LENGTH FOR 'B' TYPE FIELD

1. An illegal character was found in the length.
2. An embedded blank was found in the length.
3. The length of a B field is zero.
4. The length of a B field is greater than 9.

ESC205 LINE nn TOTAL LENGTH OF NUMERIC FIELD EXCEEDS 15

The sum of the integer and decimal portions of a P or V field exceeds 15.

ESC206 LINE nn DECIMAL PORTION ILLEGAL WITH DATA TYPE USED

A decimal portion appears after the type for an N or B field.

ESC207 LINE nn ILLEGAL DATA TYPE SPECIFIED

1. A data type other than A, N, B, P, or V was specified.
2. No data type was specified.

ESC208 LINE nn ILLEGAL COMBINATION OR DUPLICATE KEY FOUND

1. The specified key already appears in the structure.
2. An index key was specified in the same structure with a matching key.
3. A matching sorted key and a matching unsorted key were specified in the same structure.

ESC209 LINE nn ILLEGAL START POSITION SPECIFIED

The start position specification is greater than 4092.

ESC210 LINE nn ILLEGAL KEY SPECIFIED

A key other than K1-K5, MU, or MS was specified.

ESC211 LINE nn EDIT CODES NOT ALLOWED WITH ALPHA FIELD

You cannot specify an edit code in an alphanumeric field.

ESC212 LINE nn DUPLICATE EDIT CODE SPECIFIED

The same edit code is already specified for the field.

ESC213 LINE nn ILLEGAL EDIT CODE SPECIFIED

An edit code other than Z, $, *, ,, –, or / was specified.

ESC214 LINE nn ILLEGAL REPEAT COUNT SPECIFIED

The repeat count specification is greater than 255.

ESC215 LINE nn REPEAT COUNT NOT PERMITTED WITH KEYS

A repeat count is specified in a field containing a key.

ESC216 LINE nn STRUCTURE LENGTH GREATER THAN 4092

The position of the field in the record exceeds 4092.

ESC217 LINE nn Z EDIT CODE NOT PERMITTED WITH (*) or ($)

The Z edit code cannot be used with the * or $ edit codes because these codes imply zero suppression.

ESC218 LINE nn ILLEGAL FIELD LENGTH USING '/' EDIT CODE

The length of the specified field must be 4 or 6 to use the / edit code.

ESC219 LINE nn ILLEGAL DATA TYPE USED WITH (/) EDIT CODE

Only numeric (N) data may be used with the / edit code.

ESC220 LINE nn BAD SYMBOL OR EMBEDDED BLANK IN FIELD NAME

1. Blanks cannot precede or be included in field names.

2. A character other than alphabetic or numeric was found in a field name.

ESC221 LINE nn FIELD NAME CANNOT BE RESERVED WORD

The field name cannot be an ESCORT reserved word.

ESC222 LINE nn FIRST CHARACTER OF NAME MUST BE ALPHABETIC

Field names must begin with alphabetic characters.

ESC223 LINE nn MORE THAN 200 ENTRIES IN STRUCTURE

A maximum of 200 structure statements is permitted in a structure.

ESC224 LINE nn ILLEGAL DECIMAL PORTION SPECIFIED

1. A decimal portion greater than 9 was specified for a P or V field.

2. The remainder of the length/type field was not blank following the decimal portion.

ESC225 LINE nn ILLEGAL INTEGER PORTION FOR 'P' OR 'V' TYPE

1. An illegal character was detected in the length of the field.

2. A blank was found in the length of the field.

ESC226 LINE nn OTHER CODES NOT PERMITTED WITH (/) EDIT CODE

The / edit code cannot be used in combination with any other edit code.

ESC227 LINE nn 'S' PREFIX USED ONLY WITH 'N' AND 'V' DATA

Only data types that store the sign as an overpunch can use the S prefix.

ESC228 LINE nn INDEX KEY CANNOT EXCEED 80 CHARACTERS

A MIRAM key cannot be longer than 80 characters.

ESC229 ESCORT LIBRARY LOCKED - SAVE OPERATION CANCELLED

Another user is using the ESCORT library file.

Retry the command.

ESC230 UNABLE TO WRITE STRUCTURE SUCCESSFULLY TO FILE

If saving a structure, the ESCORT library file cannot be accessed for writing. Otherwise, the session file cannot be accessed for writing.

ESC231 SPECIFIED STRUCTURE DOES NOT EXIST

The structure specified does not exist on either the ESCORT library file or the session file.

ESC232 ILLEGAL STRUCTURE NAME ENTERED

1. An embedded blank appears in the structure name.

2. An illegal character appears in the structure name.

3. The first character of the structure name is not alphabetic.

STRUCTURE
ERROR MESSAGES

ESC233 STRUCTURE NAME CANNOT BE A RESERVED WORD

The structure name specified was an ESCORT reserved word.

ESC234 SPECIFIED STRUCTURE DOES NOT EXIST ON ESCORT SESSION FILE

The SAVE STRUCTURE command was attempted on a structure that was not found on the ESCORT session file.

ESC235 SELECTION NOT CURRENTLY IMPLEMENTED

A selection made from the structure menu is not implemented for this release.

ESC236 ILLEGAL SELECTION HAS BEEN MADE

The selection made does not correspond to the selections on the structure menu.

ESC241 NO STRUCTURES EXIST ON SESSION FILE

An attempt was made to display or print structure names from the session workfile, but no structures were found.

ESC243 ILLEGAL LENGTH FOR (N) FIELD

1. An illegal character was found in the length.
2. An embedded blank was found in the length.
3. The length of the N field is zero.
4. The length of the N field is greater than 15.

ESC245 INSERTION NUMBER GREATER THAN ENTRIES IN STRUCTURE

The insertion line number is greater than the number of entries currently in the structure.

ESC246 INSERTION NUMBER CANNOT BE ZERO

The insertion line number must be a valid line number greater than zero.

ESC266 NO STRUCTURES EXIST ON LIBRARY FILE

An attempt was made to display or print structure names from the ESCORT library file, but no structures were found.

ESC005 RECOVERY/SAVE NAME NOT FOUND IN CATALOG

The filename you specified for session file save or recovery was not found in the system catalog. All files used in an ESCORT session must be cataloged.

ESC006 FATAL I/O ERROR ON RECOVERY/SAVE, OPERATION ABORTED

A fatal error has occurred during save or recovery of the session file you specified.

Assure that the needed volume is online. Then, if this is a save operation and a new file, check the VTOC of the volume to be sure there is sufficient space for allocation of the file. If both of the above are satisfactory, retry the save or recover operation. If the problem recurs, this file cannot be used for save or recovery.

ESC025 JOB XXXXXXXX CANCELLED; CANNOT FIND PROGRAM XXXXXXXX

No object program exists for program named.

Enter program named, or remove program from the job.

ESC026 JOB XXXXXXXX CANCELLED WHILE RUNNING PROGRAM XXXXXXXX

Program named had fatal error.

Correct program named.

ESC027 PROGRAM TOO LARGE

Source program exceeds main storage capacity allocated by the system.

If program exceeds main storage allocation by only a few characters (50 or less), you can reduce the size of the program by combining lines and eliminating indentation of lines and punctuation characters. Otherwise, you must reduce the size of the source program.

ESC028 LITERAL NOT ENCLOSED WITH DOUBLE QUOTES

Missing double quote character.

System automatically inserts double quote character after last character of literal. If this does not correct error, you must correct source program.

ESC029 CANNOT SAVE PROGRAM WITHOUT A NAME

Attempting to save a program without a name.

Retry command, using a program name.

ESC030 NO SOURCE LANGUAGE ENTERED - COMMAND CANCELLED

No source language entered when using the ENTER or CHANGE command. If error occurred when using the CHANGE command, you probably forgot to move the cursor.

Enter source language and retry command.

ESC031 CANNOT FIND SOURCE PROGRAM ON WORKFILE - COMMAND CANCELLED

This error is caused by one of the following:

1. Changing a source program without a name and no RE$IDENT program on session file
2. Saving a source program that does not exist on session file. Program name is probably misspelled.

Retry command, using a program name (1) or using a program name that exists on the session file.

ESC032 CANNOT FIND OBJECT PROGRAM ON WORKFILE - COMMAND CANCELLED

This error is caused by one of the following:

1. Running an object program without a name and no RE$IDENT object program on the session file
2. Saving an object program that does not exist on the session file. Program name is probably misspelled.

Retry command, using a program name (1) or using a program name that exists on the session file.

ESC033 CANNOT FIND PROGRAM - COMMAND CANCELLED

Source or object program does not exist on session file or in ESCORT library. Program name is probably misspelled.

Retry command, using new program name.

ESC034 CANNOT WRITE SOURCE PROGRAM TO ESCORT LIBRARY

ESCORT library is full.

Allocate more main storage for ESCORT library.

ESC035 CANNOT WRITE OBJECT PROGRAM TO ESCORT LIBRARY

ESCORT library is full.

Allocate more disk file space for ESCORT library.

ESC036 CANNOT WRITE SOURCE PROGRAM TO WORKFILE

Session file is full. Too many ENTER and CHANGE commands were used in the current session.

Terminate current session and restart another session.

ESC037 CANNOT READ SOURCE PROGRAM FROM WORKFILE - COMMAND CANCELLED

Source program has been found on workfile, but unable to read from disk.

Reenter source program again; then execute original command and if same error occurs, contact service personnel.

ESC038 CANNOT READ SOURCE PROGRAM FROM LIBRARY - COMMAND CANCELLED

Source program has been found on library, but unable to read from disk.

Reenter source program again and save; then execute the original command and if same error occurs, contact service personnel.

ESC039 CANNOT FIND JOB NAME ON WORKFILE - COMMAND CANCELLED

Attempting to save a job that does not exist on the session file. Job name is probably misspelled.

Retry command, using a new or corrected job name.

ESC040 CANNOT FIND JOB - COMMAND CANCELLED

No job exists by name being used - probably misspelled job name.

Retry command again using new job name.

ESC041 CANNOT WRITE JOB TO WORKFILE - COMMAND CANCELLED

Session file is full. Too many ENTER and CHANGE commands are used in the current session.

Terminate current session and restart another session.

ESC042 CANNOT WRITE JOB TO LIBRARY - COMMAND CANCELLED

ESCORT library is full.

Allocate more disk file space for ESCORT library.

ESC043 CANNOT READ JOB FROM WORKFILE - COMMAND CANCELLED

Job has been found on workfile, but unable to read from disk.

Reenter job again; then execute original command and if same error occurs, contact service personnel.

ESC044 CANNOT READ JOB FROM LIBRARY - COMMAND CANCELLED

Job has been found on library but unable to read from disk.

Reenter job again and save; then execute original command and if same error occurs, contact service personnel.

ESC045 CANNOT SAVE JOB WITHOUT A NAME - COMMAND CANCELLED

Attempting to save a job without a name.

Retry command, using a job name.

ESC046 MEMORY NOT AVAILABLE - PROGRAM NOT SAVED

The object program is too large for the current main storage allocated; system automatically allocates more main storage, but none is available at this time.

Retry command when system is not so busy.

ESC047 EXCEEDED LIMIT OF FILE NAMES

You have exceeded the maximum number of files (36) allowed per session. Remove files not being used. Then reenter new file names.

ESC048 FILE NAME NOT FOUND

The file name cannot be found in the password table.

Enter the file name into the password table for this session.

ESC049 NO PASSWORDS HAVE BEEN ENTERED

No passwords have been entered for this session.

Enter passwords.

ESC050 NO MEMORY AVAILABLE

Space for the password table cannot be allocated.

Retry the command when the system is not so busy.

ESC117 RECOVERY/SAVE REQUIRES PASSWORD ENTER READ OR WRITE PASSWORD XXXXXX

The file you have specified for session file save or recovery is password protected. Enter the necessary password for this file in the space provided.

ESC119 RECOVERY/SAVE PASSWORD ERROR

An incorrect password was entered for the file you specified for session file save or recovery. Your request cannot be processed.

ESC229 ESCORT LIBRARY LOCKED - SAVE OPERATION CANCELLED

Another user is using the ESCORT library file.

Wait a few minutes and retry command.

ESC244 ESCORT LIBRARY FILE LOCK

Another user is using the ESCORT library file.

Wait a few minutes and retry the command.

ESC247 SCREEN FORMAT ERROR

An error occurred while attempting to read or write a screen format.

Tutorial mode processing cannot continue; you are returned to command mode.

ESC248 SESSION FILE READ ERROR

An error occurred while attempting to read a program from the session workfile.

Tutorial mode processing cannot continue; you are returned to command mode.

ESC249 SESSION FILE WRITE ERROR

An error occurred while attempting to write a program on the session workfile.

Tutorial mode processing cannot continue; you are returned to command mode.

ESC250 ESCORT LIBRARY WRITE ERROR

An error occurred while attempting to save a program on the ESCORT library file.

Tutorial mode processing cannot continue; you are returned to command mode. Then enter program mode and attempt to save your program.

ESC251 NO BUFFER AVAILABLE

Tutorial mode attempted to acquire a buffer from the system but none was available.

Select one of the following to perform the operation that was interrupted:

- Save a program – Enter program mode and select the SAVE PROGRAM option from the program commands menu.
- Enter a structure – Try again to create your program (and enter your structure) in tutorial mode.

If the problem recurs, try again when the system is not so busy.

ESC252 FILENAME ALREADY USED

The file name you just keyed in was previously used in this program and is illegal for use here.

Enter the name of a file that has not already been used in this program.

ESC253 NO DATA RECEIVED - PLEASE REENTER

A function key was pressed that is not supported by tutorial mode. Data was expected, but any data entered may not have been transmitted.

If you want to call a tutorial supported function by using a function key, press the correct function key. Otherwise, reenter your data and press the XMIT key.

ESC254 STRUCTURE NAMED ALREADY EXISTS

The alternate structure you chose to create for use with your file already exists.

Select another structure option or choose another name for the alternate structure you will create.

ESC255 STRUCTURE NAMED DOES NOT EXIST

The alternate structure you chose to use with your file does not exist.

Select another structure option or name an existing alternate structure for use with your file.

ESC256 NAME INVALID AS ALTERNATE

You have entered the name of the structure associated with your file name when the use of an alternate structure was requested.

Select the option to use the structure associated with your file name or enter a valid alternate structure name.

ESC257 PROGRAM ALREADY SAVED - RESELECT

You have already saved the program you just created.

Make a different selection from the menu.

ESC258 ESCORT LIBRARY READ ERROR

An error has occurred while attempting to read a program from the ESCORT library file.

Tutorial mode processing cannot continue. You will be returned to command mode.

ESC008 SYSTEM ERROR XXXX OCCURRED ACCESSING FILE XXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX

A system error occurred while ESCORT was trying to access a data file.

See Appendix A of OS/3 System Messages Programmer/Operator Reference, UP-8076 (current version) for an explanation of the error code. Check to see that the hardware resources needed to access this file are available.

ESC009 DM nn TYPE nn OCCURRED ACCESSING FILE

An error condition was returned by data management on a file access.

Refer to the OS/3 system messages programmer/operator reference, UP-8076 (current version) for an explanation and corrective action.

ESC010 PASSWORD PROTECTION ERROR, COULD NOT OPEN FILE XXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXX

The passwords specified do not allow access to the file.

Supply the correct passwords and retry.

ESC011 FILE NOT CATALOGED, COULD NOT OPEN XXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX

The file name specified was not found in the catalog. All data files referenced in ESCORT programs must be cataloged.

ESC012 FILE NOT FOUND OR COULD NOT BE ALLOCATED, COULD NOT OPEN XXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

The specified file does not exist, or there is no file space to allocate the file.

Correct the file name if misspelled, or make space available for file allocation. Retry.

ESC013 FILE OPENED WITHOUT WRITE PASSWORD, COULD NOT WRITE TO XXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXX

The write password was not supplied, and the file was opened with read only access.

Supply write password and retry.

ESC014 FILE TYPE IS NOT MIRAM, COULD NOT OPEN XXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXX

All data files referenced in ESCORT programs must be MIRAM data files.

ESC015 DUPLICATE KEY ERROR OCCURRED IN ACCESSING FILE XXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXX

An attempt was made to write a record with the same key as an existing record.

ESC016 KEY CHANGE ERROR OCCURRED ACCESSING FILE XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXX

A key change during an update was attempted in a key that does not allow changes.

ESC017 RECORD HAS NO RCB, COULD NOT DELETE FROM FILE XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Delete operations can be performed only on records with RCBS.

ESC019 KEY MATCH NOT FOUND ACCESSING FILE XXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXX

A record with the specified key was not found.

ESC020 ESCORT INTERNAL ERROR IN ESC$DFOO MODULE

An internal logic error has occurred within ESCORT.

Contact your local Sperry Univac representative.

ESC021 COULD NOT UPDATE/DELETE RECORD, NO PREVIOUS READ FROM FILE XXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

Consecutive update/delete operations were attempted without intervening read.

ESC022 COULD NOT ACQUIRE DEVICE REQUIRED TO OPEN FILE XXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXX

The device or volume required to open the specified file was not available. The file could not be opened.

Contact the system operator to determine why the device or volume was not available.

RUNTIME PROCESSOR
ERROR MESSAGES

ESC173 SYSTEM MEMORY NOT AVAILABLE

A buffer was requested from the system, but none was available.

Attempt to recompile the program when the system is not so busy.

ESC174 INPUT/MASTER KEY HAS NO MATCH

The key value for the input file does not match any key values for the master file.

Correct the program.

ESC175 WORKSTATION I/O ERROR

An unrecoverable I/O error occurred while trying to execute an ENTER, FROM WS, DISPLAY, or OUTPUT TO WS clause.

ESC176 ARITHMETIC RESULT OVERFLOW

The receiving field in a record is not large enough to contain the results of the current arithmetic calculation.

Enlarge the field, or reduce the size of the calculation.

ESC178 SUBSCRIPT EXCEEDS ARRAY MAXIMUM

During program execution, a subscript has exceeded the maximum repeat count for an array.

ESC179 STACK OUTSIDE OF ALLOWABLE RANGE

This is an internal ESCORT error. Contact your local Sperry Univac representative.

ESC180 INVALID PROGRAM OBJECT CODE

This is an internal ESCORT error.

Contact your local Sperry Univac representative.

ESC181 INVALID FUNCTION KEY

The only valid function key for run-time processing is F4.

ESC190 ERROR OCCURRED WHILE ATTEMPTING TO USE THE SCREEN FORMAT FOR INPUT DATA

ESCORT has encountered a screen format error.

A screen containing additional information is displayed with this message.

ESC191 ERROR OCCURRED WHILE ATTEMPTING TO USE THE SCREEN FORMAT FOR CHANGE DATA

ESCORT has encountered a screen format error.

A screen containing additional information is displayed with this message.

ESC192 ERROR OCCURRED WHILE ATTEMPTING TO USE THE SCREEN FORMAT FOR OUTPUT TO WS

ESCORT has encountered a screen format error.

A screen containing additional information is displayed with this message.

ESC193 ERROR OCCURRED WHILE ATTEMPTING TO USE THE SCREEN FORMAT FOR OUTPUT TO WS

ESCORT has encountered a screen format error.

A screen containing additional information is displayed with this message.

ESC194 KEYS INCONSISTENT WITH KEY POSITION/LENGTH ON FILE

The key position or length specified in the file structure does not match the key position or length on the file as found when the file was opened.

ESC195 FILE XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXX STRUCTURE SIZE GREATER THAN MAXIMUM RECORD SIZE

The structure you have used with the specified file is larger than the records in the file.

Correct the structure to be no larger than the records, or use another structure of correct size and format for the file.

Appendix F. SYSGEN Requirements and System Considerations

F.1. SYSGEN REQUIREMENTS

You must specify the following SYSGEN options when you use ESCORT:

System 80	Series 90	Options	
		SYSGEN	Hardware
X	X	Maximum timer services	
X	X	Resident tasking support	
X	X	Spooling	
X	X	File lock	
	X	CDI	
	X	Shared code support	
	X	Dynamic buffer management	
	X	ISKEY	
	X		Featured instruction set

If you use the EXPREGION and RESBUFSIZ keyword parameters, do not specify EXPREGION=0 and RESBUFSIZ=(some large value). This causes an error when used with ESCORT. Instead, specify EXPREGION=20000 and use the default for RESBUFSIZ (500 bytes).

F.2. EFFICIENCY RECOMMENDATIONS

1. Make sure ESCORT modules are resident. Generate the system that way or specify it during IPL.

2. Place files carefully. Avoid the RES pack, and place COBOL work files and ESCORT files on different packs, if possible.

3. Avoid using the console.

F.3. CODING RESTRICTIONS

It is not possible to code ESCORT programs using the General Editor (EDT) because EDT uses variable-length records and ESCORT uses fixed-length records.

Appendix G. Custom Formatting Using Structures

The following example (Figure G-1) uses custom structures to custom-format output. The example reads a file, prints a heading on each page of output, lists detail lines, and prints a subtotal for each customer.

Explanation of example: system utility fields

$LINE and $PAGE, the system utility fields specified, control the spacing and page breaks in the example. You can test these fields and set them to different values. They increment automatically. They have a value of zero initially and must be set to a value other than zero to start the internal counter. In the program, $PAGE is initialized to 1 in the IF FIRSTIME clause. $LINE is set to 1 during the first cycle of the program (when $LINE equals 0) or when the line count is greater than 50. Setting $LINE to 1 causes a top-of-page and printing of, in this case, the heading, beginning on the line specified.

Custom structures

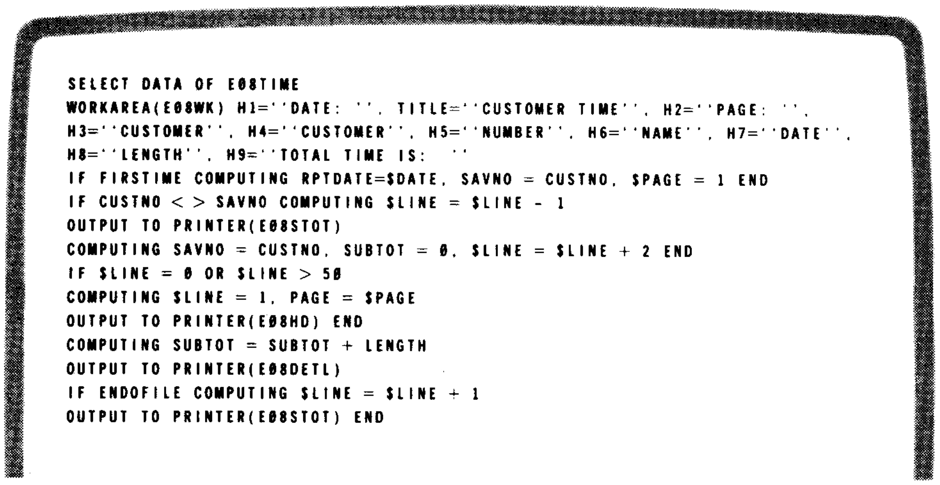
The structures E08HD, E08DETL, and E08STOT are custom structures. E08HD uses dummy fields (the FIL fields) to space the data. In E08DETL and E08STOT, the starting position for the field is specified when the structure is entered, eliminating the need for the dummy fields. This method allows fields with edit codes and V or P type numeric fields to be expanded.

Allocating temporary storage

Initializing fields

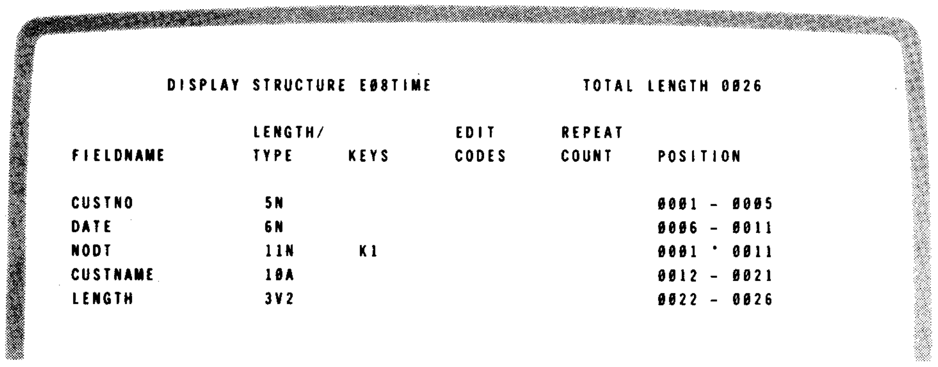
The WORKAREA clause allocates temporary working storage space and initializes values of data fields not defined in the master or input files. All fields are initialized to blanks (alphanumeric fields) or zeros (numeric fields). When a field is to have a value other than blank/zero, setting it equal to that value initializes the field. In this example, the structure E08WK contains those fields not found in E08TIME that are referenced in the custom structures for output, and fields used for comparison and calculations. The title and column headings are also initialized here. Values for the other fields are computed later in the program.

Note that the key of the file is redefined; that is, a starting position was specified when the structure was created, and another field (or fields) occupies the same position.



```
SELECT DATA OF E08TIME
WORKAREA(E08WK) H1=''DATE: '', TITLE=''CUSTOMER TIME'', H2=''PAGE: '',
H3=''CUSTOMER'', H4=''CUSTOMER'', H5=''NUMBER'', H6=''NAME'', H7=''DATE'',
H8=''LENGTH'', H9=''TOTAL TIME IS:  ''
IF FIRSTIME COMPUTING RPTDATE=$DATE, SAVNO = CUSTNO, $PAGE = 1 END
IF CUSTNO < > SAVNO COMPUTING $LINE = $LINE - 1
OUTPUT TO PRINTER(E08STOT)
COMPUTING SAVNO = CUSTNO, SUBTOT = 0, $LINE = $LINE + 2 END
IF $LINE = 0 OR $LINE > 50
COMPUTING $LINE = 1, PAGE = $PAGE
OUTPUT TO PRINTER(E08HD) END
COMPUTING SUBTOT = SUBTOT + LENGTH
OUTPUT TO PRINTER(E08DETL)
IF ENDOFILE COMPUTING $LINE = $LINE + 1
OUTPUT TO PRINTER(E08STOT) END
```

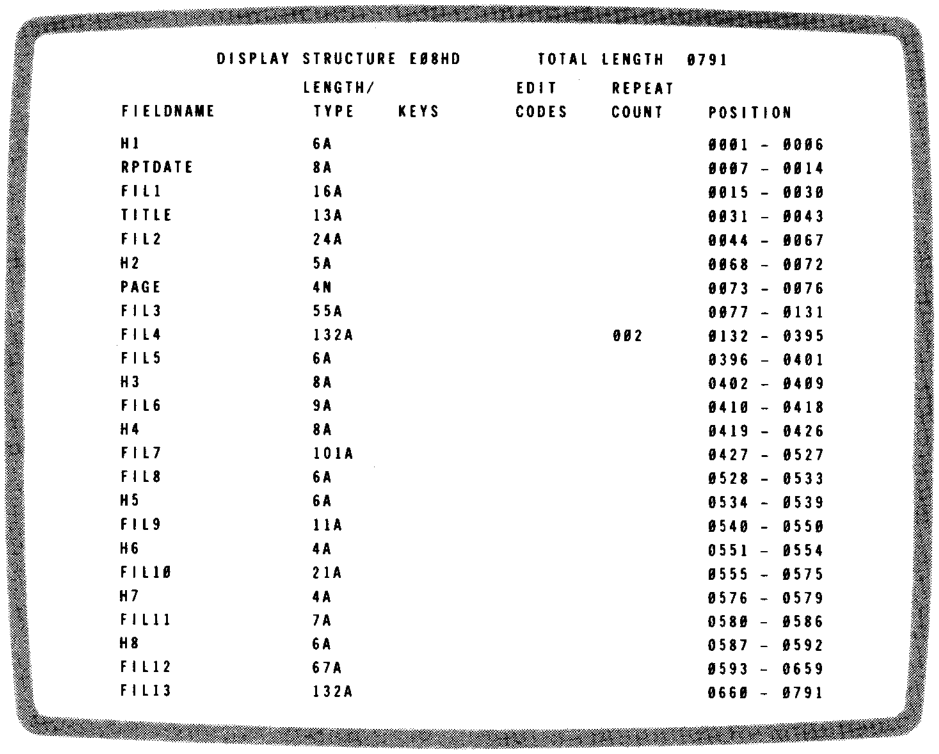
a. Display of program E08TIME



```
          DISPLAY STRUCTURE E08TIME            TOTAL LENGTH 0026

                   LENGTH/              EDIT      REPEAT
FIELDNAME          TYPE       KEYS      CODES     COUNT     POSITION

CUSTNO              5N                                      0001 - 0005
DATE                6N                                      0006 - 0011
NODT                11N       K1                            0001 ' 0011
CUSTNAME            10A                                     0012 - 0021
LENGTH              3V2                                     0022 - 0026
```

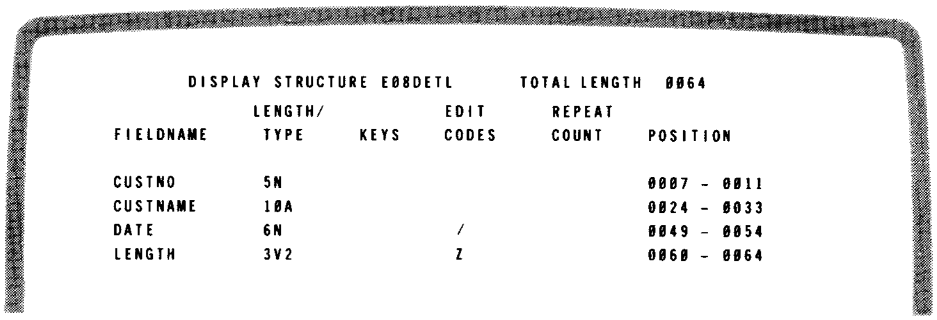
b. Structure E08TIME

Figure G–1. Example of Custom Formatting Using Structures (Part 1 of 6)



```
          DISPLAY STRUCTURE EØ8HD        TOTAL LENGTH  Ø791
                      LENGTH/                EDIT    REPEAT
FIELDNAME             TYPE      KEYS         CODES   COUNT     POSITION

H1                    6A                                       ØØØ1 - ØØØ6
RPTDATE               8A                                       ØØØ7 - ØØ14
FIL1                  16A                                      ØØ15 - ØØ3Ø
TITLE                 13A                                      ØØ31 - ØØ43
FIL2                  24A                                      ØØ44 - ØØ67
H2                    5A                                       ØØ68 - ØØ72
PAGE                  4N                                       ØØ73 - ØØ76
FIL3                  55A                                      ØØ77 - Ø131
FIL4                  132A                           ØØ2       Ø132 - Ø395
FIL5                  6A                                       Ø396 - Ø4Ø1
H3                    8A                                       Ø4Ø2 - Ø4Ø9
FIL6                  9A                                       Ø41Ø - Ø418
H4                    8A                                       Ø419 - Ø426
FIL7                  101A                                     Ø427 - Ø527
FIL8                  6A                                       Ø528 - Ø533
H5                    6A                                       Ø534 - Ø539
FIL9                  11A                                      Ø54Ø - Ø55Ø
H6                    4A                                       Ø551 - Ø554
FIL1Ø                 21A                                      Ø555 - Ø575
H7                    4A                                       Ø576 - Ø579
FIL11                 7A                                       Ø58Ø - Ø586
H8                    6A                                       Ø587 - Ø592
FIL12                 67A                                      Ø593 - Ø659
FIL13                 132A                                     Ø66Ø - Ø791
```

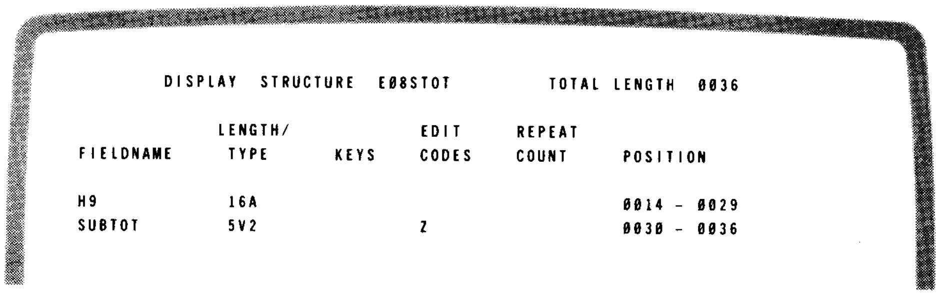
c. Structure EO8HD



```
          DISPLAY STRUCTURE EØ8DETL      TOTAL LENGTH  ØØ64
                  LENGTH/              EDIT      REPEAT
FIELDNAME         TYPE       KEYS      CODES     COUNT     POSITION

CUSTNO            5N                                       ØØØ7 - ØØ11
CUSTNAME          1ØA                                      ØØ24 - ØØ33
DATE              6N                     /                 ØØ49 - ØØ54
LENGTH            3V2                    Z                 ØØ6Ø - ØØ64
```

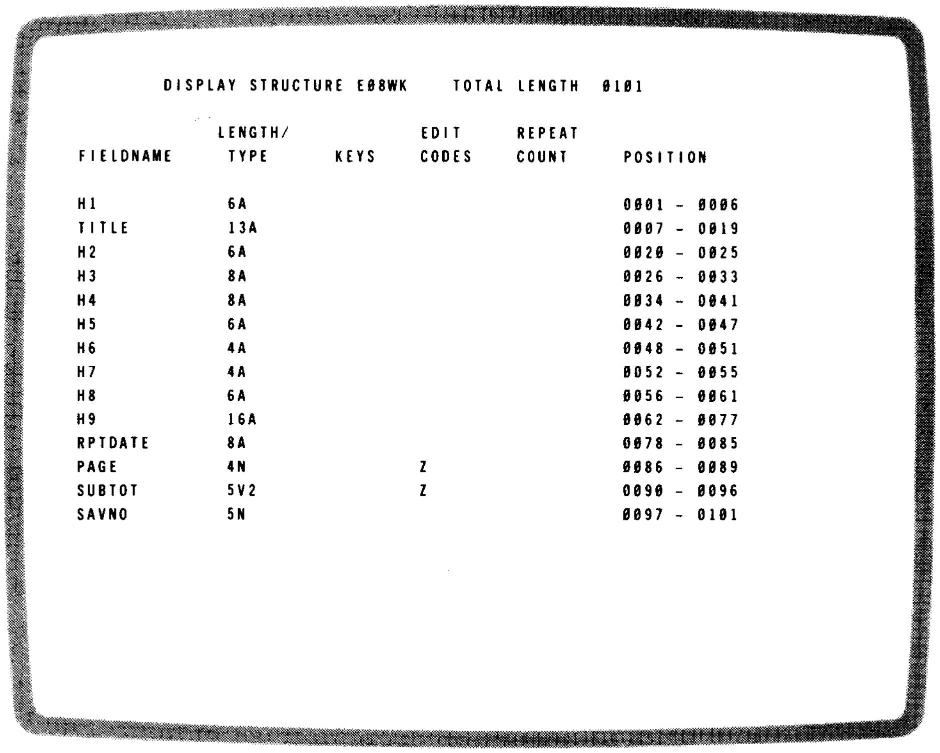
d. Structure EO8DETL

Figure G–1. Example of Custom Formatting Using Structures (Part 2 of 6)



```
        DISPLAY  STRUCTURE  E08STOT        TOTAL LENGTH  0036

                LENGTH/                EDIT     REPEAT
FIELDNAME       TYPE        KEYS       CODES    COUNT      POSITION

H9              16A                                        0014 - 0029
SUBTOT          5V2                    Z                   0030 - 0036
```

e. Structure E08STOT



```
        DISPLAY STRUCTURE E08WK     TOTAL LENGTH  0101

                LENGTH/                EDIT     REPEAT
FIELDNAME       TYPE        KEYS       CODES    COUNT      POSITION

H1              6A                                         0001 - 0006
TITLE           13A                                        0007 - 0019
H2              6A                                         0020 - 0025
H3              8A                                         0026 - 0033
H4              8A                                         0034 - 0041
H5              6A                                         0042 - 0047
H6              4A                                         0048 - 0051
H7              4A                                         0052 - 0055
H8              6A                                         0056 - 0061
H9              16A                                        0062 - 0077
RPTDATE         8A                                         0078 - 0085
PAGE            4N                     Z                   0086 - 0089
SUBTOT          5V2                    Z                   0090 - 0096
SAVNO           5N                                         0097 - 0101
```

f. Structure E08WK

Figure G–1. Example of Custom Formatting Using Structures (Part 3 of 6)

```
DATA     OF    E08TIME

CUSTNO     DATE     CUSTNAME       LENGTH

00001     800110     AAAAAAAAAA     010.00
00001     800120     AAAAAAAAAA     010.00
00001     810101     AAAAAAAAAA     010.00
00002     800105     BBBBBBBBBB     010.00
00002     800115     BBBBBBBBBB     010.00
00003     800109     CCCCCCCCCC     010.00
00004     800121     DDDDDDDDDD     010.00
00004     800122     DDDDDDDDDD     010.00
00004     800325     DDDDDDDDDD     010.00
00004     800712     DDDDDDDDDD     010.00
00004     801101     DDDDDDDDDD     010.00
00004     810106     DDDDDDDDDD     010.00
00005     810623     EEEEEEEEEE     010.00
00006     810213     FFFFFFFFFF     010.00
00006     810304     FFFFFFFFFF     010.00
00006     810325     FFFFFFFFFF     010.00
00007     801028     GGGGGGGGGG     010.00
00007     801205     GGGGGGGGGG     010.00
00007     801210     GGGGGGGGGG     010.00
00008     810414     HHHHHHHHHH     010.00
00008     810513     HHHHHHHHHH     010.00
00009     810101     IIIIIIIIII     010.00
00009     810303     IIIIIIIIII     010.00
00009     810401     IIIIIIIIII     010.00
00010     800504     JJJJJJJJJJ     010.00
00010     810302     JJJJJJJJJJ     010.00
00010     810509     JJJJJJJJJJ     010.00
00010     810601     JJJJJJJJJJ     010.00
00010     810602     JJJJJJJJJJ     010.00
00010     810603     JJJJJJJJJJ     010.00
00010     810604     JJJJJJJJJJ     010.00
00010     810605     JJJJJJJJJJ     010.00
00010     810606     JJJJJJJJJJ     010.00
00010     810607     JJJJJJJJJJ     010.00
00010     810608     JJJJJJJJJJ     010.00
00010     810609     JJJJJJJJJJ     010.00
00010     810610     JJJJJJJJJJ     010.00
```

g. List of data for program E08TIME

Figure G–1. Example of Custom Formatting Using Structures (Part 4 of 6)

```
DATE: 82/03/05             CUSTOMER TIME              PAGE:0001

        CUSTOMER    CUSTOMER         DATE       LENGTH
        NUMBER      NAME

         00001       AAAAAAAAAA       80/01/10     10.00
         00001       AAAAAAAAAA       80/01/20     10.00
         00001       AAAAAAAAAA       81/01/01     10.00

                  TOTAL TIME IS:     30.00

         00002        BBBBBBBBBB      80/01/05     10.00
         00002        BBBBBBBBBB      80/01/15     10.00

                  TOTAL TIME IS:     20.00

         00003        CCCCCCCCCC      80/01/09     10.00

                  TOTAL TIME IS:     10.00

         00004        DDDDDDDDDD      80/01/21     10.00
         00004        DDDDDDDDDD      80/01/22     10.00
         00004        DDDDDDDDDD      80/03/25     10.00
         00004        DDDDDDDDDD      80/07/12     10.00
         00004        DDDDDDDDDD      80/11/01     10.00
         00004        DDDDDDDDDD      81/01/06     10.00

                  TOTAL TIME IS:     60.00

         00005        EEEEEEEEEE      81/06/23     10.00

                  TOTAL TIME IS:     10.00

         00006        FFFFFFFFFF      81/02/13     10.00
         00006        FFFFFFFFFF      81/03/04     10.00
         00006        FFFFFFFFFF      81/03/25     10.00

                  TOTAL TIME IS:     30.00

         00007        GGGGGGGGGG      80/10/28     10.00
         00007        GGGGGGGGGG      80/12/05     10.00
         00007        GGGGGGGGGG      80/12/10     10.00

                  TOTAL TIME IS:     30.00
```

h. Printout for program E08TIME, Page 1

Figure G–1. Example of Custom Formatting Using Structures (Part 5 of 6)

```
DATE: 82/03/05          CUSTOMER TIME            PAGE: 0002

     CUSTOMER   CUSTOMER
     NUMBER     NAME            DATE        LENGTH

      00008      HHHHHHHHHH      81/04/14    10.00
      00008      HHHHHHHHHH      81/05/13    10.00

             TOTAL TIME IS:    20.00

      00009      IIIIIIIIII      81/01/01    10.00
      00009      IIIIIIIIII      81/03/03    10.00
      00009      IIIIIIIIII      81/04/01    10.00

             TOTAL TIME IS:    30.00

      00010      JJJJJJJJJJ      80/05/04    10.00
      00010      JJJJJJJJJJ      81/03/02    10.00
      00010      JJJJJJJJJJ      81/05/09    10.00
      00010      JJJJJJJJJJ      81/06/01    10.00
      00010      JJJJJJJJJJ      81/06/02    10.00
      00010      JJJJJJJJJJ      81/06/03    10.00
      00010      JJJJJJJJJJ      81/06/04    10.00
      00010      JJJJJJJJJJ      81/06/05    10.00
      00010      JJJJJJJJJJ      81/06/06    10.00
      00010      JJJJJJJJJJ      81/06/07    10.00
      00010      JJJJJJJJJJ      81/06/08    10.00
      00010      JJJJJJJJJJ      81/06/09    10.00
      00010      JJJJJJJJJJ      81/06/10    10.00

             TOTAL TIME IS:    130.00
```

i. Printout for program E08TIME, Page 2

Figure G–1. Example of Custom Formatting Using Structures (Part 6 of 6)

Glossary

A

array
Adjacent fields of similar data (e.g., the months in a year) that occur several times and have the same length for each occurrence (e.g., 3-letter representations for each month: JAN, FEB, MAR, etc).

C

character string operators
Words that let you set conditions on a field by specifying an alphanumeric string of data. For example, IF FIELDA STARTS WITH "DOG".

clauses
The part of program statements that lets you describe to ESCORT the specific operations you want to perform on your data file.

commands
The part of ESCORT that lets you tell ESCORT what operation you want to perform on your programs, structures, and jobs.

compile
The method by which ESCORT checks the syntax of your program statements and translates them into a language the computer can understand.

consolidated data management
The data access method used by ESCORT that is set by your system administrator at system generation time.

constants
Those values assigned to data fields that do not change during program execution.

D

data
Facts that are organized in a formalized manner and are suitable for communication and processing.

direct entry method
The program mode method that lets you key program statements directly onto a blank screen.

E

edit codes
A set of symbolic editing codes you designate in your structure that perform certain functions on numeric fields.

F

field
The smallest unit of an ESCORT file. Each field holds an individual piece of data.

field name
A 1- to 8-alphanumeric-character designation you assign to identify each field of a record in your file.

file
A group of related records taken as a whole.

H

HELP
The ESCORT facility that explains the function of the segment in which you request help.

I

index keys
Keys that designate fields in your structure that you use to sort your data into meaningful order. These also designate the matching criteria when defining the file to the system. See also *sort keys*.

indexed file
A file organized in a logical manner with data records and a set of record keys that point to each data record.

interactive program mode
The ESCORT operating mode you'll use after you learn ESCORT by using interactive tutorial mode.

interactive tutorial mode
The operating mode that teaches you how to use the ESCORT language.

J

job
A group of related programs that are executed sequentially.

K

keys
(See *sort keys* and *index keys*.)

L

LENGTH/TYPE
A structure statement parameter that designates how long and what type of data fields are in your record.

library
An area on your disk where ESCORT programs, structures, and jobs are stored following a SAVE operation.

logical operator
Words you use to link one or more conditional statements. The two logical operators are OR and AND.

LOGON command
The OS/3 interactive services command that lets you gain access to the workstation and the system.

M

matching criteria
Structure fields designated as index key fields or matching fields.

matching sorted
A type of matching criteria you use for an unkeyed file. It tells the system the file is arranged in ascending order according to that data field.

matched unsorted
A type of matching criteria you use for an unkeyed file. It tells the system the file is not arranged in any particular order.

MIRAM (multiple indexed random access method)
The file type used by ESCORT.

N

nonindexed file
A file organized sequentially. This means the records appear on the file in the order you enter them.

P

program
A collection of program statements and clauses that perform operations on your data files.

program statement
One of five statements that describe generally the operation you want to perform on your files.

R

record
A group of data fields relating to a specific item.

relational operator
A set of arithmetic type operators that regulate the entry, change, or selection of your data.

repeat count
The structure parameter that lets you specify the number of times an array field is repeated in the record.

reserved words
All words used in the ESCORT language. This means you can't use them as file or field names, but you can use them as constants or in data fields.

RE$IDENT (program or job)
The default name that ESCORT assigns when you enter or change an unnamed program or job.

S

screen format coordinator
The component of screen format services that coordinates the transfer of input and output data between your program and your form.

screen format generator
The component of screen format services that allows you to create screen forms from the workstation.

screen format services
The OS/3 component that allows you to create customized input and output forms to use with your ESCORT program.

screen menu method
The program mode method that lets you use a series of menus to build your program.

session file
A disk file where everything you enter or change is contained. This file is only accessible during your current session and afterwards only if you name and save it.

sort keys
Keys that designate fields in your structure that you designate to sort your data into meaningful order (e.g., alphabetically, by account number, etc).

start position
The structure statement parameter that lets you designate where in the record you want a certain field or fields to appear.

structure
The collection of structure statements through which you tell ESCORT what your file looks like by defining each field in the record with a structure statement.

structure statement
The statement through which you define a field in a record by specifying the field name, length, and type.

system bulletin
The bulletin that gives you information about the operation of your system. It is displayed after you key in the LOGON parameters.

U

utility fields

A group of fields supplied and used by the system that perform often used operations.

V

variable name qualifiers

Qualifiers that tell the system what file structure a specified field is from. This is helpful when you have more than one structure with the same field name referenced in your program.

Index

Term	Reference	Page
Direct entry method		
advantages	1.2	1-6
	4.1	4-1
intended user	4.1	4-1
selection	1.2	1-5
	4.1	4-1
Direct entry of clauses		
choosing	4.22	4-58
how to access	4.22	4-58
Direct entry of program statements		
choosing direct entry of statements	4.22	4-58
direct entry example	4.22	4-58
how to access	4.22	4-58
keyin format	4.22	4-59
program length	4.22	4-58
screen format	4.22	4-58
DISPLAY clause		
format	7.21	7-25
parameters	7.21	7-25
use	7.21	7-25
DISPLAY JOB command		
named jobs	5.26	5-18
purpose	5.26	5-18
unnamed jobs	5.26	5-18
DISPLAY JOB NAMES command	5.27	5-18
DISPLAY PASSWORDS command	5.36	5-22
DISPLAY PROGRAM command		
named programs	5.5	5-5
purpose	5.5	5-5
unnamed programs	5.5	5-5
DISPLAY PROGRAM NAMES command		
E$AF	5.6	5-5
library types	5.6	5-5
purpose	5.6	5-5
DISPLAY STRUCTURE command		
action	5.16	5-11
purpose	5.16	5-11
DISPLAY STRUCTURE NAMES command	5.17	5-12

Term	Reference	Page
E		
Edit codes		
function	6.7	6-13
symbols and functions	6.7	6-13
	Table 6-1	6-13
END clause	7.22	7-26
ENTER DATA program statement		
basic format	7.4	7-5
basic statement	7.5	7-6
clauses	7.5	7-6
complete format	7.4	7-5
	A.3	A-2
example	7.5	7-6
example program	7.5	7-7
example structures	7.5	7-6
format	7.5	7-6
parameter uses	7.5	7-6
parameters	7.5	7-6
related clauses	7.3	7-4
use	7.3	7-4
ENTER DATA program statement (program mode)		
creating a new data file	4.5	4-6
data entry methods	4.4	4-6
extending an existing data file	4.6	4-11
ENTER DATA program statement (tutorial mode)		
creating a data file	3.5	3-4
options	3.4	3-4
options selected	3.4	3-4
purpose	3.4	3-4
ENTER JOB command		
named jobs	5.28	5-18
purpose	5.28	5-18
unnamed jobs	5.28	5-18
ENTER OR CHANGE PASSWORDS command	5.37	5-23
ENTER PROGRAM command		
named programs	5.7	5-5
purpose	5.7	5-5
unnamed programs	5.7	5-5
ENTER STRUCTURE command		
action	5.18	5-12
enter structure screen	5.18	5-12
purpose	5.18	5-12

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

Cut along line.

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

CUT