CONDOR SERIES 20

Relational Database Management System for the HP Series 100 Personal Office Computers

User's Manual

CONDOR COMPUTER CORPORATION
P.O. Box 8318
Ann Arbor, Michigan 48107

1st Edition April 1982
2nd Edition February 1983

45550-90001

Printed in US

Preface

This manual describes the Condor Series 20 Database Management System for the HP Series 100 Personal Office Computers. It is written for the business manager or professional, and no prior familiarity with database management systems or concepts is assumed. The early chapters provide a conceptual overview of database management, and a step-by-step introduction to database creation and use. Experienced readers may want to skim these chapters, and concentrate on the later chapters and appendices.

This manual does assume familiarity with the general operation of an HP 125 or HP 120 system, as described in the HP Series 100 Owner's Manuals. If you are not familiar with HP Series 100 operation and terminology, you should review this manual before proceeding.

Condor Series 20 is available in three levels. Level 1 (20-1) provides a powerful set of commands for creating, managing and manipulating individual databases. Level 2 (20-2) adds relational commands for interrelating data from several different databases. Level 3 (20-3) adds advanced report writing and database indexing capabilities. This manual describes the full set of available Level 2 commands. Level 3 commands are included in a 20-3 manual section. Features not available to the Level 1 user are clearly noted. Upgrade products are available for Level 1 users who later desire the full capabilities of Level 2 and for Level 2 users who later desire the full capabilities of Level 3.

Disclaimer

CONDOR COMPUTER MAKES NO REPRESENTATIONS OF WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. FURTHER, CONDOR COMPUTER CORPORATION RESERVES THE RIGHT TO REVISE THIS PUBLICATION AND TO MAKE CHANGES FROM TIME TO TIME IN THE CONTENT HEREOF WITHOUT OBLIGATION OF CONDOR COMPUTER TO NOTIFY ANY PERSON OF SUCH REVISION OR CHANGES.

Table of Contents

Chapter 1 Database Concepts

- What is a Database?..... 1-2
- Database Structure..... 1-4
- What Does Condor Series 20 Do?..... 1-5
- Database Forms..... 1-5
- How Does Series 20 Work?..... 1-6
- Chapter Summary..... 1-7

Chapter 2 Getting Started with CONDOR Series 20

Commands Used in This Chapter
DEFINE
ENTER
LIST
PRINT
SORT
SYSTEM

- Installing Condor Series 20..... 2-1
- Starting Condor Series 20..... 2-4
- Building Your First Database..... 2-4
- Database Commands..... 2-7
- The Current Disc Drive..... 2-8
- Exiting Series 20..... 2-9
- Chapter Summary..... 2-10

Chapter 3 Defining a Database

Commands Used in This Chapter
DEFINE
- Naming the Database..... 3-2
- Naming Data Items..... 3-2
- Designing the Database Form..... 3-4
- Choosing the Disc to Store the Database..... 3-5
- Creating the Database Form..... 3-6
- Defining Data Items..... 3-7
- Maximum and Minimum Values..... 3-9
- Default Values..... 3-10
- Chapter Summary..... 3-12

Chapter 4 Entering and Updating Data

Commands Used in This Chapter
ENTER
UPDATE
- Data Entry..... 4-1
- Data Entry Softkeys..... 4-3
- Range Checking..... 4-5
- Default Data Values..... 4-5
- The Auto-Default Feature..... 4-6
- The Auto-Repeat Feature..... 4-6
- Updating Data..... 4-7
- Search Conditions..... 4-10
- Deleting Unwanted Records..... 4-11
- Chapter Summary..... 4-12

Chapter 5 Simple Inquiries and Reports

Commands Used in This Chapter
DISPLAY
LIST
PRINT
SORT
STAX
TABULATE
TITLE
- Simple Database Inquiry..... 5-1
- The DISPLAY Command..... 5-2
- Approximate Matches in Search Conditions..... 5-4
- Database Report Formats..... 5-5
- Sorting a Database..... 5-6
- Screen Format Report..... 5-7
- Screen Format Reports on the Screen..... 5-9

- Columnar Reports..... 5-10
- Columnar Reports with Statistics..... 5-11
- Columnar Reports on the Screen..... 5-14
- Summary Reports..... 5-14
- Statistical Reports..... 5-16
- Printing Summary and Statistical Reports..... 5-17
- Report Titles..... 5-17
- Chapter Summary..... 5-19

Chapter 6 Relational Database Capabilities

Commands Used in This Chapter

APPEND
CHANGE
COMBINE
COMPARE
COMPUTE
DELETE
ENTER
JOIN
LIST
POST
PROJECT
SAVE
SELECT

- The RESULT Database..... 6-1
- Selecting Database Records..... 6-2
- Saving the RESULT Database..... 6-3
- Combining Two Databases..... 6-5
- The APPEND Command..... 6-5
- The COMBINE Command..... 6-6
- The JOIN Command..... 6-7
- Projecting a Database..... 6-9
- Database Arithmetic..... 6-11
- The COMPUTE Command..... 6-12
- The CHANGE Command..... 6-12
- The POST Command..... 6-13
- Comparing Two Databases..... 6-16
- Deleting Database Records..... 6-18
- Chapter Summary..... 6-20

Chapter 7 Database Utilities

Commands Used in This Chapter

COPY
DATE
DBMS
DESTROY
DIC
DIR
EMPTY
LOGDISK
RENAME
SET
SYSTEM
$

- Changing Discs..... 7-1
- Listing Database Names..... 7-2
- Listing Disc Contents..... 7-3
- Changing the Name of a Database..... 7-3
- Copying a Database..... 7-3
- Destroying Unwanted Databases..... 7-4
- Emptying a Database..... 7-4
- Setting the Current Date..... 7-5
- Printing Reports to a Data File..... 7-5
- Setting Series 20 Operating Characteristics..... 7-6
- Starting Condor with DBMS command..... 7-7
- Exiting Series 20..... 7-8
- Running Non-Series 20 Programs..... 7-8
- Chapter Summary..... 7-9

Chapter 8 Using Databases with Other Applications

Commands Used in This Chapter

READ
SET PRINTER
WRITE

- Printing Reports to a File..... 8-1
- Read and Write Commands..... 8-2
- Word/125 Mailing Lists..... 8-3
- Word/125 Documents..... 8-4
- Plotting Database Data..... 8-5
- Databases and User Application Programs..... 8-6
- Exchanging Data with "Host" Computers..... 8-7
- Using Data From Word/125 Documents..... 8-9
- Reading Visicalc Data Into a Database..... 8-10
- Chapter Summary..... 8-11

Chapter 9 Restructuring and Reorganizing Databases

Commands Used in This Chapter

COPY
DEFINE
DESTROY
ENTER
FORMAT
HELP
READ
REORG
UPDATE
WRITE

- Database Structure..... 9-1
- The RESULT Database..... 9-2
- Other Database Files..... 9-2
- Manipulating Database Files..... 9-3
- Revising a Screen Form..... 9-3
- Printing Data Item Definitions..... 9-4
- Revising Data Item Definitions..... 9-5
- Changing Maximum, Minimum and Default Values.... 9-6
- The Data Dictionary..... 9-7
- Alternate Database Descriptions..... 9-9
- The Reorganizing a Database..... 9-13
- Reorganization using READ/WRITE..... 9-14
- Chapter Summary..... 9-16

Chapter 10 Database Command Procedures

Commands Used in This Chapter

ABORT
EDIT
HELP
RESTART
RUN
*ECHO
*END
*GET
*IF
*LET
*MESSAGE
;

- Command Procedures..... 10-1
- Command Procedure Processing..... 10-3
- Comments in Command Procedures..... 10-4
- Echoing Command Procedure Execution..... 10-5
- Messages in Command Procedures..... 10-5
- Command Procedure Parameters..... 10-5
- Conditional Processing..... 10-6

- Ending or Aborting a Procedure..... 10-7
- Changing Parameter Values..... 10-7
- Command Procedure Example..... 10-8
- Restarting a Command Procedure..... 10-10
- Editing Command Procedures..... 10-10
- HELP Screens..... 10-12
- Aborting a HELP Screen..... 10-14
- Chapter Summary..... 10-14

Chapter 11 Transaction Processing with CONDOR Series 20

- General Ledger Application Example..... 11-1
- Defining the Example Databases..... 11-2
- Entering Data into the Example Databases..... 11-3
- Entering Unique Records..... 11-4
- Entering Matching Records..... 11-6
- Posting Transactions to a Master Database..... 11-7
- Transaction Processing Help Screens..... 11-9
- Example Command Procedure Files..... 11-9
- Daily Transaction Processing..... 11-11

Chapter 12 Writing Reports*

Commands Used in This Chapter
REPORT

- Aspects Controlled in Formatting a Report..... 12-1
- Setting Up Your Condor 20-3 Discs..... 12-2
- About Using the Report Writer..... 12-2
- Invoking the Report Writer by Way of the Menu... 12-4
- Sample Warehouse Report..... 12-5
- Creating a Report Specification..... 12-7
- Creating a Report Form..... 12-8
- Indicating Line Printing Specifications..... 12-10
- Item-Type Specifications..... 12-12
- Printing Format..... 12-14
- Totals, Subtotals, Averages, Min, and Max..... 12-14
- Computed Item Equations..... 12-15
- Section Breaks..... 12-16
- Use of Form Feeds..... 12-16
- Report Specification Naming..... 12-16
- Fine Tuning the Report Specification..... 12-17
- Revising a Report Specification..... 12-18
- Describing a Report Specification..... 12-19
- Printing a Report..... 12-19
- Using Report Writer Without Going Through Menu.. 12-21
- Using the Report Writer in a Command Procedure.. 12-21
- Report Writer Specifications and Helpful Hints.. 12-22
- Maximum Number of Specifications Allowed..... 12-22
- Reports Wider than 80 Columns..... 12-24
- Line Spacing/Positioning..... 12-25
- Chapter Summary..... 12-27

Chapter 13 Indexing Your Database*

Commands Used in This Chapter
INDEX
- Reasons to Index a Database..... 13-1
- Setting Up or Re-Creating a Database Index..... 13-2
- Indexing a New Database..... 13-2
- Indexing an Existing Database..... 13-3
- Re-Creating an Index..... 13-4
- Duplicate Keys..... 13-4
- Updating an Index..... 13-5
- Invalidating an Index..... 13-5
- Commands that Affect Indexed Databases..... 13-6
- Chapter Summary..... 13-7

Glossary of Terms

Appendix A Series 20 Command Descriptions

Appendix B CONDOR Series 20 Specifications

Appendix C Series 20 File Formats

Appendix D Example Databases Used in This Manual

Appendix E CONDOR Series 20 Command Quick Reference

Appendix F Creating a Report with Report Writer*

Index

(*) Found only in the Condor 20-3 Manual Section

CHAPTER 1

DATABASE CONCEPTS

Condor Series 20 is a powerful and easy to use database management system for the HP Series 100 Personal Computers. In almost any business situation where you might say "I need to keep track of ....", Series 20 can help you keep track faster and more effectively.

With Condor 20, you can store important information, and then manipulate it in many useful ways. For example, you might track important customers and their orders. On demand, you could look up information about individual customers, print customer reports alphabetically or by geography, or combine customer information with order data to summarize orders by customer, by sales region, or by product line. You can transfer that information to Word/100 or Graphics/100 to create documents or plot color charts. And that is just the beginning of Series 20's capabilities!

Applications of Condor 20 are as varied as the different types of information that you use every business day. Condor 20 can track personal information, such as client names and addresses. A department can use it to keep track of office equipment, or employee time cards. You can use it to track spare parts inventory, or a stock of office supplies. And it can form the foundation for a complete transaction processing application.

This manual introduces the capabilities of Condor Series 20, step-by-step. Throughout the manual, the names Condor Series 20, Condor 20, and Series 20 will be used interchangably. The early chapters will help you "get going" quickly, with a few commands that let you build and use simple databases. Later, you can move through the manual to learn more about Condor 20's features, as you need them. This chapter begins by discussing a few fundamental database concepts.

WHAT IS A DATABASE?

First of all, what is a "database"? A "database" is an organized collection of related information that can be retrieved and manipulated on demand. Chances are that you already use many non-computerized "databases" every day, without calling them databases.

One of the best examples of a database is a secretary's card file. The card file keeps track of people -- their names, addresses and phone numbers, etc. Figure 1-1 shows a typical card from a card file. Notice how the card file fits the database definition: it is **organized** alphabetically, it stores **related, highly structured information**, and it is used to **retrieve information** many times each day.

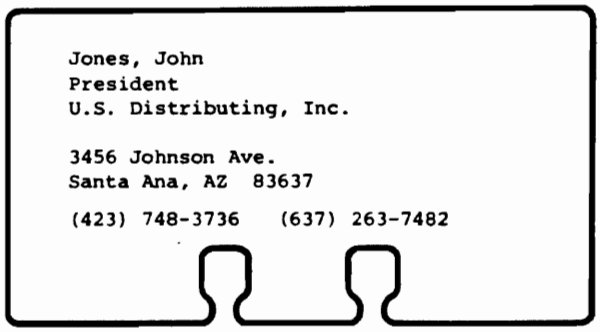
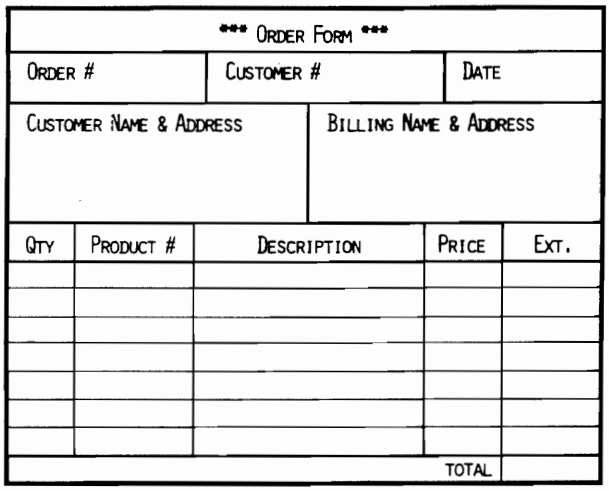


Figure 1-1. A Typical Card from a Card File

Another familiar database is the clerk's file folder holding the day's orders. Each order is recorded on a single order form, on which is written all the pertinent information about the order: customer name, address, product ordered, quantity, price, totals, etc. Figure 1-2 shows a typical order form from this "ORDERS database."



*** ORDER FORM ***

ORDER #	CUSTOMER #	DATE
CUSTOMER NAME & ADDRESS	BILLING NAME & ADDRESS	

QTY	PRODUCT #	DESCRIPTION	PRICE	EXT.
			TOTAL	

Figure 1-2. A Typical Order Form

Figure 1-3 shows another typical business document, a regional sales report. Could this data be viewed as a database? Yes! And you will often examine the information in Series 20 databases with printed reports like this one.

***** REGIONAL SALES REPORT FOR APRIL, 1982 ******

REGION.NAME	ACTUAL.MTD	QUOTA.MTD	ACTUAL.YTD	QUOTA.YTD
CANADA	2445.00	2400.00	14253.56	15000.00
MID-ATLANTIC	9475.45	10000.00	48394.39	50000.00
MIDWEST	6738.46	7000.00	41200.00	35000.00
MOUNTAIN	4393.00	4500.00	29382.45	27500.00
NEW ENGLAND	5238.34	5000.00	23057.39	26750.00
N. PACIFIC	7636.34	7000.00	53627.34	50000.00
S. PACIFIC	5346.34	5000.00	37463.34	38000.00
SOUTHERN	1927.78	2000.00	9372.92	10000.00
Total	43200.71	42900.00	256751.39	252250.00

Figure 1-3. Regional Sales Report

Actually, the row/column report is just the "card file" format of the first example, in disguise. Imagine that the data in each row of the report was written on a separate index card and then filed away in a card file. The data remains the same as in the report, of course, but the information is now in a "database" much more like the other two examples.

A card file is one of the best and simplest ways of thinking about a database. Card files and a "CARDFILE" database will be used often as examples in this manual.

DATA BASE STRUCTURE

How is a database organized? What is its structure and form? The CARDFILE database provides an excellent example to answer these questions.

A database is organized as a collection of **records**, just as a card file is organized as a collection of cards. Each card, for example, is one record. The card stores all the information related to one person -- name, title, address, firm, etc. The records correspond one-to-one with the people they represent. This will be true of most databases that you will use with Series 20; there will be a **one-to-one correspondence between database records** (the "index cards") **and some physical or logical entity** -- a customer, an order, a product, a sales region, etc.

Each CARDFILE record also has its own individual structure. It is made up of several pieces of information -- a name, an address, etc. We call these individual pieces of information **data items.** Each CARDFILE record contains **exactly the same** data items as the others. This will also be true for **every** Series 20 database: every record in the database contains exactly the same data items as the others. Figure 1-4 summarizes the structure of a database -- a database is comprised of records; each record contains the same data items as the others.

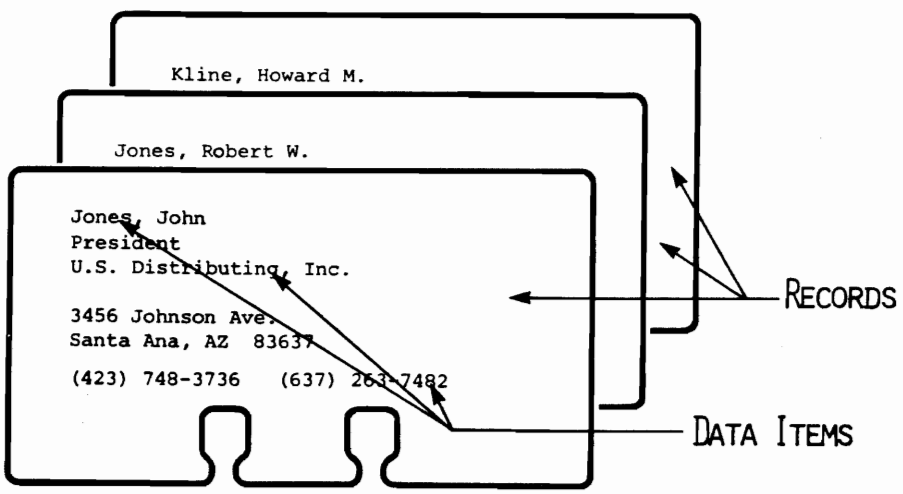


Figure 1-4. Records and Data Items

WHAT DOES CONDOR SERIES 20 DO?

Condor Series 20 is an HP Series 100 software package that stores databases on magnetic discs instead of card files. Using Series 20, information can be stored more compactly, and more accurately than with manual databases. Inquiries or reports are available in a fraction of the time it would take to prepare them with a manual system. And information that would be too expensive or time-consuming to generate by hand is available in minutes, at the touch of a few keys.

Condor 20 makes an excellent "personal assistant" for a business manager or professional, helping to keep track of important information. It can also be an important tool for the secretary. And in the hands of a system designer, it can form the basis for an entire transaction processing application.

DATABASE FORMS

Every Condor 20 database has an associated **screen form** -- a format for presenting database information on the HP Series 100 screen. The form presents information from a single record of the database at a time. You create the form when you first define a new database to Condor 20. From then on, the form is used to enter, update and display information in the database.

There is nothing mysterious about a database form. A card file, for example, has a fixed, consistent format for each card. Figure 1-5 shows how this card format can easily become an HP Series 100 screen form for an "electronic CARDFILE" database.

```
**************************************************************
**                                                          **
**      ***** HP SERIES 100 CARDFILE DATABASE *****         **
**                                                          **
**    NAME :        Jones, John                             **
**    TITLE :       President                               **
**    FIRM :        U. S. Distributing, Inc.                **
**                                                          **
**    STREET :      3456 Johnson Ave.                       **
**    CITY :        Santa Ana                               **
**    STATE :       AZ        ZIPCODE :   83637             **
**                                                          **
**    BUS-PHONE :   (423) 748-3736                          **
**    HOME-PHONE :  (637) 263-7482    REV-DATE : 01/26/83   **
**                                                          **
**************************************************************
```

Figure 1-5. CARDFILE Screen Form

HOW DOES CONDOR SERIES 20 WORK?

How do you actually use Condor 20 to create and access a database? The first step is to make some decisions -- what to name the data base and what data items it will contain. With this information, you are ready to create a new database.

First, you must **define** the database. You will "paint" the database form on the HP Series 100 screen, using the editing keys on the keyboard. Condor 20 obtains much of the information it needs about the database directly from the form. It will also ask you for additional information. After you supply all the needed information, Condor 20 creates the database on the disc.

Next, you will **enter information** into the database. The form you have just created is used for data entry, record by record. You can also update records in the database, or delete records, all using the same, familiar form.

Later, you will want to **retrieve data** from the database. Again, the form can be used to display retrieved information on demand. Or, you may want to **print reports**, based on database contents.

You can use information from Condor 20 databases with other HP Series 100 applications, too. For example, you might plot a pie chart from an ORDERS database, include a database report as part of a memo, or generate a mailing list from a CARDFILE database.

After a while, you will have many different databases, and will want to **relate these databases to one another.** In this way, even complex reports and inquiries can be handled on demand, drawing on stored information to gain new insights or to look at a problem in a new way.

But why talk about Condor 20 when you're ready to use it yourself? Go on to the next chapter, and get started on building your first Condor Series 20 database!

CHAPTER SUMMARY

* A **database** is an organized collection of related information, like a card file or folder full of order forms.

* Series 20 maintains databases electronically on discs, with the benefits of **speed, accuracy** and **powerful, fast, flexible inquiry and reporting.**

* A Series 20 database is comprised of **records.** Each **record** is comprised of **data items,** and all records in a database have the same data items.

* Data from Series 20 databases can be exchanged with other HP 125 applications, such as Word Processing, Graphics and Data Communications.

* Each Series 20 database also has an associated **screen form,** which is used to enter, update and display database information.

CHAPTER 2

GETTING STARTED WITH CONDOR SERIES 20

In this chapter, you will install Condor Series 20 onto your HP Series 100 system, and build and use your first Series 20 database. By carefully following the instructions presented here, you should be using your own database in less than an hour.

INSTALLING CONDOR SERIES 20

The one or two flexible discs that are packaged with this manual contain the Condor 20 Database Management Software. Before you can use Condor on your HP Series 100, you must "install" Condor onto "work discs" which you will use in everyday operation. The installation procedure is fully described in the Owner's Manual that was shipped with your system.

```
**************************************************************
*                                                            *
*                  ***** CAUTION *****                       *
*                                                            *
*    Condor Series 20 is a large and advanced software       *
*    product.  When Condor 20-1 or 20-2 is installed on      *
*    a 5-1/4" or 3-1/2" flexible disc, it must be the        *
*    only application on the disc.  No other programs        *
*    or files except the CP/M operating system may be        *
*    present on the "work disc" on which Condor is           *
*    installed.  If you also purchased the 20-3 section      *
*    disc, you will have to install it onto a separate       *
*    disc.                                                   *
*                                                            *
*    On larger HP Series 100 discs (8" flexible discs or     *
*    Winchester discs), Condor can easily share a disc       *
*    with other applications.  Condor 20-1 requires          *
*    185 KBytes of disc space; 20-2 requires 229 KBytes;     *
*    20-3 requires 279 KBytes.                               *
**************************************************************
```

1. Installing your Condor 20-1 or 20-2 disc.

This process consists of the following two steps:

a. As detailed in your system Owner's Manual, create a work disc with only the HP Series 100 operating system.

b. Install your Condor application disc onto the work disc. The procedure for installing an application is also detailed in the system Owner's Manual.

When you see the welcome screen on the work disc, a softkey will show **"Condor 20-1"** or **"Condor 20-2"**, depending on which you have installed.

2. Installing your Condor 20-3 disc.

This same process as above is followed when installing onto 3-1/2" or 5-1/4" discs, or if you prefer to place 20-2 and 20-3 on separate 8" or Winchester discs.

a. Create a work disc with only the HP Series 100 operating system.

b. Install your Condor 20-3 application disc onto the work disc.

You may wish to place 20-2 and 20-3 together on the same 8" or Winchester disc. This will allow you to access all of the Condor capabilities from the same place. To place them together, the following procedure is used.

a. Ensure that there is at least 240K of empty space left on the disc which contains Condor 20-2. However, the Condor 20-3 portion will not use that much space after it is installed, because it will share some common files with Condor 20-2.

b. Install the Condor 20-3 application disc onto the same disc with 20-2. The procedure is the same as was used to install Condor 20-2. At the welcome screen, an additional softkey will show **"Condor 20-3"**. Pressing either of the "Condor 20-2" or "Condor 20-3" softkeys allows you to use all of the 20-2 and 20-3 commands.

AFTER YOU HAVE COMPLETED INSTALLATION OF SERIES 20, IT IS EXTREMELY IMPORTANT THAT YOU SAVE THE ORIGINAL SERIES 20 APPLICATION DISC IN A SAFE PLACE. The application disc is your backup disc in case you should accidentally damage your work disc(s).

STARTING CONDOR SERIES 20

To begin processing databases with Condor Series 20, insert the 20-1 or 20-2 work disc you have just created into the **A:** drive in your system. Insert a blank, **formatted** disc into the **B:** drive -- this disc will contain your first few databases. (If your HP Series 100 uses a Winchester disc, then you can ignore these two steps.)

Follow the normal system start up procedure (see the Owner's Manual that accompanied your system for details). The Welcome Menu will appear, and one of the softkeys will be labelled **"Condor 20-1"** or **"Condor 20-2"**. Press this softkey. In a few seconds, a copyright notice will appear, and then Condor will ask you for your first command by displaying the prompt:

A>>

You are now ready to begin using Series 20. You will follow this same procedure each time you start a session with Series 20.

Whenever you want to use the Report Writer or Index of Condor 20-3, start up your system with the above procedures, using your 20-3 work disc.

BUILDING YOUR FIRST CONDOR 20 DATABASE

As a "warm-up" exercise, we will build a simple Condor 20 database to keep track of birthdays. Don't worry if you don't understand every detail as you go through the exercise. By following the instructions carefully, you will discover some powerful Condor 20 features, and see how quickly you can apply Condor 20 to your own tasks.

Do the following:

1. Type **B: <RETURN>**
   This signals Series 20 that you will create a database on the **B:** drive. Series 20 will respond with the prompt **"B>>"**.

2. Type **DEFINE BIRTHDAY <RETURN>**
   The system will ask if you want to create a new form.

3. Type **Y**
   In a few moments, a screen for creating the new form will be displayed. It will be blank except for the softkeys at the bottom.

4. Type **<RETURN> <RETURN> [NAME]** _____
   Type **<RETURN> <RETURN> [BIRTHDATE]** _____
   Type 16 underscores (**important: not hyphens**) for NAME, and 8 underscores for BIRTHDATE.

5. Press the **END** function key.
   This ends form creation, and stores the BIRTHDAY form on the disc. The system asks if the database is to be defined.

6. Type **Y <RETURN>**
   The system will now prompt you for the definition of each data item in the form. In a few moments, it will display the prompt **> NAME:**, indicating that NAME is the first data item to be defined.

7. Type **AN <RETURN>**
   You have specified that NAME contains alphanumeric data. Next, the system will display **> BIRTHDATE:**, requesting a definition for the BIRTHDATE data item.

8. Type **J <RETURN>**
   You have specified that BIRTHDATE is a Julian (date) data item. Since this is the last data item, the system requests that you review the data definitions and asks if they are correct.

9. Type **Y <RETURN>**
   The system will indicate **Busy** while it creates the database. It will then ask if you want a printed copy of the data definitions. If you have a printer on your HP Series 100...

10. Type **Y <RETURN>**
    The system prints attributes for the database **BIRTHDAY**, including the number of data items, their definitions, and the total record length. A **B>>** prompt will be displayed, and you have finished creating your first Series 20 database. Congratulations!

ENTERING DATA INTO YOUR DATABASE

With the BIRTHDAY database now defined, you are ready to enter data. Data is entered directly into the form, record by record. Simply follow the steps below:

1. Type **ENTER BIRTHDAY <RETURN>**
   The system will display the BIRTHDAY form on the screen. The fields on the screen where data may be entered are highlighted in inverse video, and the cursor is positioned in the first field, ready for you to enter a name.

2. Type **John Doe <RETURN>**
   When you press the RETURN key, the cursor moves to the next field, ready for you to enter a birth date.

3. Type **10/15/45 <RETURN>**
   When you press the RETURN key, the form is complete, and a set of softkey labels appear. By pressing the appropriate softkey, you may return to revise the data in the form, abort the data entry, end the data entry, or continue by entering a new set of data for a new record.

4. Press the **Continue** softkey
   The data in the form is now added to the database. The form is cleared, ready to accept the next record.

5. Repeat steps 1-4 several times
   With each form you enter, use a different name, but repeat some of the birthdays, to make the later reports interesting. If you like, you may use the data listed below:

NAME	BIRTHDATE
Jane Doe	10/20/50
Paul Anderson	2/5/62
Bob Greene	10/20/55
Mary Roberts	2/5/59

6. Press the **End** softkey.
   When you have entered a few records, end data entry by striking the **End** softkey. The form will disappear, and the B>> prompt will be displayed. Data entry is now complete.

MANIPULATING DATA IN A DATABASE

Now that you have seen how easy it is to create and enter data into a Series 20 database, try some Series 20 commands for manipulating and reporting data:

1. Type **SORT BIRTHDAY BY BIRTHDATE <RETURN>**
   The system will print messages indicating progress of the sort operation. After completion, the database will be in ascending date sequence.

2. Type **LIST BIRTHDAY BY BIRTHDATE, NAME <RETURN>**
   The system will display a two-column report on the screen, listing NAME and BIRTHDATE for each record. The data appears in sequence by birth-birthdate.

3. Type **SORT BIRTHDAY BY NAME <RETURN>**
   This command will sort the database into ascending sequence by (first) name.

4. Type **PRINT BIRTHDAY BY BIRTHDATE, NAME <RETURN>**
   This time, the report will appear on the system printer, the data is in sequence by name, and the columns are in the opposite order.

This exercise has given only a brief introduction to the data manipulation and reporting power of Series 20. The following chapters describe in detail the Series 20 capabilities for entering, updating, manipulating and reporting information.

DATABASE COMMANDS

Each time Series 20 displays the prompt:

A>> or **B>>** or **C>>** etc.

it is waiting for a **command** from you. Commands are the way that you tell Series 20 what to do; each of the 41 available commands performs its own unique function. Most Series 20 commands have the same format:

verb database-name qualifying-information [option]

VERB -- tells what action is to be taken; ENTER, PRINT, SORT, etc.

DATABASE-NAME -- tells what database to act upon

QUALIFYING -- varies depending on the command;

INFORMATION tells which data items to list, how to sort a database, etc.

[OPTION] -- selects different command options, such as whether information is displayed on the screen or printer

Examples:

DEFINE BIRTHDAY
(define the database named BIRTHDAY)
LIST BIRTHDAY
(list the database contents on screen)
UPDATE BIRTHDAY WHERE NAME IS JOHN*
(update the database record for JOHN DOE)
TABULATE BIRTHDAY BY DATE [P]
(print database statistics by date, on the printer)

Condor 20 commands can be up to 128 characters long. If you type a command longer than one line, Series 20 automatically displays the last part of the command on a separate line. You must only press the **RETURN** key when you have completely finished typing the command. If you make a mistake while typing a command, use the **BACKSPACE** key to move the cursor back to the character in error, and then retype the command from that character onward. The arrow keys **cannot** be used to edit Series 20 commands.

THE CURRENT DISC DRIVE

The letter before the ">>" in Series 20's command prompt tells you which disc is the "current disc" on which Series 20 is working. Unless you specifically tell Series 20 otherwise, it will always process databases on the current disc drive. You can change the "current disc" by typing the name of another disc drive following the command prompt.

Examples:

To change the current disc to drive B:, type:

B: <RETURN>

when the **A>>** prompt appears.

To switch drive **C:**, type:

C: <RETURN>

when the **A>>** prompt appears.

You can also ask Condor 20 to process a database that is **not** on the current disc. Just give the name of the disc drive just before the name of the database when you type in a command.

Examples:

LIST D:BIRTHDAY
(list the **BIRTHDAY** database on disc drive D:)

LIST B:BIRTHDAY
(list the **BIRTHDAY** database on drive B:)

LIST BIRTHDAY
(list the **BIRTHDAY** database on the current disc drive)

EXITING SERIES 20

When you are finished using Series 20, and wish to return to the Welcome Menu, type the command:

SYSTEM <RETURN>

After a few seconds, the HP Series 100 Welcome Menu will reappear. This is the way you should always exit Condor Series 20, to be certain that your databases are maintained in an orderly manner.

NEVER EXIT CONDOR 20 BY PERFORMING A HARD-RESET ON YOUR SYSTEM OR TURNING OFF THE SYSTEM POWER. DOING SO MAY RESULT IN LOSS OF DATA AND/OR LOSS OF YOUR DATABASES.

CHAPTER SUMMARY

* Condor 20 is installed on "work discs", like all HP Series 100 applications.

* To start Series 20, press the **"Condor 20-x"** softkey on the Welcome Menu.

* Condor 20 uses simple, english-like commands to define databases, to enter data into them, to sort them, print them, etc.

* Unless instructed otherwise, Series 20 uses the **current disc** for all database processing. You can change the current disc to a different drive by typing the drive name in response to the command prompt.

* Databases not on the current disc drive can be processed by including the drive name before the database name in Series 20 commands.

* The **SYSTEM** command exits Series 20 and returns you to the Welcome Menu. You must **always** exit Series 20 this way; never by performing a hard reset while Series 20 is in operation.

CHAPTER 3

DEFINING A DATABASE

Before you can use a database, you must first **define** the database for Series 20. That is, you must design the database form, and provide information on each data item in the database. You have already defined one simple database (the BIRTHDAY database) in the warm-up example of the previous chapter. This chapter describes the steps for defining a new database in more detail.

The general procedure for defining a database is:

1). Choose a name for the new database.

2). Decide what data items the database will contain; choose a name for each data item.

3). Design a screen form for the database. The form will be used to enter, update and display database information on the screen.

4). Decide on which disc the database will be stored.

5). Use the **DEFINE** command to paint the database form on the HP 125 screen.

6). Continuing with the **DEFINE** command, describe each data item to Series 20 in detail.

When you have completed these six steps, your database will be defined, and ready to accept data. The following sections describe each step in detail. (The example databases used in this Chapter, and throughout the manual, are fully described in Appendix D, for reference.)

NAMING THE DATABASE

The name that you choose for a new database should be descriptive and easy to remember. The name may have from one to eight alphabetic or numeric characters.

Example:

Here are some good examples of database names:

```
CARDFILE -- an electronic CARDFILE database
ORDERS   -- a database of customer orders
PRODUCTS -- a database of product information
REGIONS  -- a sales region database
CLIENTS  -- a database of client information
GLJOURNL -- a General Ledger journal database
CHTACCTS -- a Chart of Accounts database
```

NAMING DATA ITEMS

The most important decision when planning a new database is deciding what data items it will contain. You **can** add data items to a database later, but it is **much, much easier** to make certain that you have included all the needed data items at the beginning. Remember, each record in the database will usually correspond to a physical entity -- a customer, a part, an order, etc. Ask yourself "What information is important to remember about this entity?. What information might you want to use to retrieve, compare, accumulate, or select database records later?" That information should be included in the database as one or more data items.

An individual data item is the smallest bit of information that you can access in a database. If you were to store a client address as a single data item, for example, then you could not break the address down into street, city, and state later. You could not, for example, sort the database by state. If this type of sorting or selecting was important to your application, you would split the address into separate street, city and state data items.

Each data item in the database has its own **data item name.** Data item names can be up to fifteen characters long. Try to use short, descriptive names, since you will be typing them often as you use the database.

```
*************************************************************
*                  ***** CAUTION *****                      *
*                                                           *
*   The "special" characters "*", "?", "#", "<",            *
*   ">", "[", "]", and the single and double quote          *
*   characters all have special meaning in certain          *
*   Series 20 commands.  If you use these characters        *
*   in data item names, they may lead to confusion when     *
*   using these commands.  Avoid them if at all possible.   *
*   Data item names that contain embedded blanks (such as   *
*   NEW DATE) should also be avoided if at all possible.    *
*   It is better to "artificially" eliminate the blanks     *
*   with a period or hyphen (e.g. use NEW.DATE).  If        *
*   you must use embedded blanks, the data item name must   *
*   always be enclosed in quotes when used in Series 20     *
*   commands.                                               *
*************************************************************
```

Here are some examples of good data item names:

```
ADDRESS        -- for a street address
ZIP.CODE       -- for a zip code
PROMDATE       -- for promised date
ORDERDATE      -- for order date
```

Example:

The CARDFILE database will consist of the following data items:

NAME -- the name of the person who is described in this CARDFILE record. Up to 20 characters will be allowed.

TITLE -- the person's job title. Up to 20 characters will be allowed.

FIRM -- the name of the company the person represents. 30 characters allowed.

STREET -- the street address. 30 characters.

CITY -- the city. 20 characters.

STATE -- the state. 2 characters.

ZIPCODE -- the zip code. 5 characters.

BUS-PHONE -- a 10-character business phone no.

HOME-PHONE -- a 10-character home phone no.

REV-DATE -- revision date for this information. 8 characters, in mm/dd/yy format.

DESIGNING THE DATABASE FORM

The database form determines how database information is displayed on the HP Series 100 screen. The form is used to enter data into the database, to update the data, and to display selected records in response to a database inquiry. For each data item, the form will contain the **data item name**, followed by a **data entry field** that is used to enter or display the data for that data item.

For simple databases, it is often most effective to simply list the data items items down the left hand side of the screen, one data item to each row. If a paper business form is already being used in your firm to process the data in a database, you should design the screen form to resemble the paper form as closely as possible, for ease of use. Otherwise, arrange the data items in the most logical sequence on the display.

Example:

The CARDFILE database form should closely resemble the type-written card from the card file. One possible design is shown in Figure 3-1.

```
**************************************************************
**                                                          **
**          *****  HP SERIES 100 DATABASE  *****            **
**                                                          **
**    [NAME]:      _____________________                    **
**    [TITLE]:     _____________________                    **
**    [FIRM]:      ___________________________              **
**                                                          **
**    [STREET]:    ___________________________              **
**    [CITY]:      _____________________                    **
**    [STATE]:     __      [ZIPCODE]:  _____                **
**                                                          **
**    [BUS-PHONE]:  _______________                         **
**    [HOME-PHONE]: _______________  [REV-DATE] ________    **
**                                                          **
**************************************************************
```

Figure 3-1. CARDFILE Screen Form

Each data item name in the database **must** appear somewhere on the form, enclosed in square brackets (see example above). Series 20 searches the form for these square brackets to determine the data item names for the database.

Following each data item name on the form is a **data entry field** for that data item. This field is used to enter or display data for that particular data item. The field must be long enough to accommodate the largest data item value that will be entered or displayed (e.g. the longest customer name, or the largest dollar amount). Data entry fields are indicated on the form by a series of underscore characters (_).

Other information on the form (i.e. all the information except the data item names and the data entry fields) is simply treated as "part of the form". This information may be informative, or may make the form more readable, but it is purely cosmetic in function. The asterisks and title of the CARDFILE example form are examples of good "form cosmetics".

With experience, you will be able to design simple forms directly on the HP Series 100 screen. For the first few forms, it is probably a good idea to design the form on paper first.

CHOOSING THE DISC TO STORE THE DATABASE

Generally, you will place a new database on a disc other than the one that contains your work copy of Condor Series 20 software. The reason is simple -- Series 20 is a rather large program, and in most cases, there will not be enough space remaining on the Series 20 work disc to store a database of any significant size. This is especially true if your system uses 5-1/4" flexible discs.

On dual floppy disc systems, you will normally place your work disc with Condor Series 20 software in drive A:, and store the database on a flexible disc in drive B:. On Winchester disc systems, the database is often located on the winchester disc B:, C:, or D:.

Be certain that the disc you have chosen to store your database is formatted before attempting to use it (see directions for Disc Formatting in the **Owners Manual**). Label the disc, and insert it into the proper drive for processing.

Regardless of which drive will contain the database, you should usually select that drive as the "current drive" for ease of operation.

Example:

The sample CARDFILE database will be stored on a flexible disc dedicated to that purpose. The disc is labelled "CARDFILE" with a felt tip pen, and is inserted into disc drive B. Series 20 is loaded, and displays the prompt: **A>>**

In response, type: **B: <RETURN>**

Series 20 responds with the prompt:

B>>

indicating that drive B: is now the current drive.

CREATING THE DATABASE FORM

With planning for the database complete, and the proper disc selected, you are ready to define the database to Series 20. The **DEFINE** command is used to define new databases as shown in the steps below:

1. Type **DEFINE database-name <RETURN>**
   The system will ask if you wish to create a new form.

2. Type **Y <RETURN>**
   The screen will blank, ready for form input.

3. Type in the database form.
   Use the arrow keys on the keyboard to move anywhere on the screen, and "paint" the form on the screen by typing characters where they belong. Position titles, headings, data item names and data entry fields as desired. To help in positioning fields, you can display a grid on your screen by pressing the **Grid** softkey. To erase the grid, press the **Refresh screen** softkey. If you make a mistake, use the cursor keys or the **BACKSPACE** key to move to the error and correct it by over typing on the screen.

   Remember to enclose data item names in square brackets ([]). Data entry fields are indicated as a sequence of underscores (_) (Note: **not** hyphens!). Underscores for a single data entry field must be consecutive, with no intervening blanks or other characters. Other characters may come between the data item name and the data entry field, however.

4. Press the **End** softkey.
   This softkey ends form creation and automatically stores the form on the current disc. If you are dissatisfied with the form, and do not wish to save it, press the **Abort** softkey instead of **End**.

   The system will ask if you want to define a new database.

5. Type **Y <RETURN>**
   You will now enter data item definitions for each data item on the form.

Example:

Enter the CARDFILE database form with the following steps:

Type **B: <RETURN>**
This sets drive B: as the current drive.

Type **DEFINE CARDFILE <C/R>**
The screen will blank, ready for you to "paint" the form. It should appear as shown in Figure 3-1.

DEFINING DATA ITEMS

The data item names for a new database all appear on the database form, enclosed in square brackets. However, the form provides only a limited about of information about each data item -- its name, and its length (indicated by the number of underscores in the data entry field). The form does not tell whether the data item is alphabetic or numeric, whether it is a dollars-and-cents value, or a date, etc. The next step is to define each data item, in detail, to Series 20.

Series 20 can handle ten different types of data item:

CODE	TYPE	MINIMUM	MAXIMUM
A	Alphabetic	1 character	127 characters
AR	Alphabetic-Required	1 character	127 characters
AN	Alphanumeric	1 character	127 characters
ANR	Alphanumeric-Required	1 character	127 characters
$	Dollars	-$21,483,736.47	+$21,483,736.47
$R	Dollars-Required	-$21,483,736.47	+$21,483,736.47
J	Julian Date	01/01/00	12/31/99
JR	Julian Date-Required	01/01/00	12/31/99
N	Numeric	-2,148,373,647	+2,148,373,647
NR	Numeric-Required	-2,148,373,647	+2,148,373,647

The data type of each data item is used to check data for correctness when you are entering data into a database. For example, Series 20 will not allow you to enter a name into a dollars field, or enter a dollars-and-cents quantity into a date field. Appendix B describes in detail, for each data type, what data may be entered.

Fields that are "required" cannot be skipped over during data entry; Series 20 will force you to enter data in these fields. Fields that are not "required" may be skipped when entering data.

After you have finished entering the database form, Series 20 will prompt you, data item by data item, for the **type** of data that that the data item represents. Enter the appropriate code (from the table above), to indicate the type of data. Press **<RETURN>** after typing each code. Series 20 will take the default (built-in) minimum and maximum values for each type of data, and display them on the screen. (You will see how you may select other minimum, maximum and default values in the next section).

Example:

Continuing from the previous example, Series 20 will ask for the data type of each data item in the CARDFILE database. Respond as follows:

NAME:	**ANR**	<RETURN>
TITLE:	**AN**	<RETURN>
FIRM:	**AN**	<RETURN>
STREET:	**AN**	<RETURN>
CITY:	**AN**	<RETURN>
STATE:	**AN**	<RETURN>
ZIPCODE:	**N**	<RETURN>
BUS-PHONE:	**AN**	<RETURN>
HOME-PHONE:	**AN**	<RETURN>
REV-DATE:	**JR**	<RETURN>

After you press the RETURN key each time, Series 20 will fill in the default minimum, maximum, and default values for the data item, based on its type. For the CARDFILE database, these values will serve our purposes well.

Series 20 will next ask if the data item definitions are correct. Type **"N"** if a data item definition is incorrect, and repeat the procedure. If all the definitions are correct, type **"Y"**. The system will create the database on the disc, displaying a "Busy" message until it is finished.

Finally, Series 20 will ask if you want a printed copy of the definitions. Type **Y** to get a printed copy. The database is now defined; you may enter data into the database.

If you later discover that there is an error in the database form, or in the data item definitions, you can use the **FORMAT** or **DEFINE** commands to edit the incorrect information. Chapter 9 describes how to edit the form and definitions, as well as several advanced features of these commands.

MINIMUM AND MAXIMUM VALUES

When data is being entered into a database, Series 20 checks each data item value, as it is entered, against the minimum and maximum values allowed for that data item. If the entered data falls outside the acceptable range, an error message is displayed, and the data must be re-entered correctly.

The minimum and maximum values have slightly different meanings, depending on the type of data item:

* For **Alphabetic** or **Alphanumeric** data items, they refer to the minimum number of characters and the maximum number of characters that may be entered in the field.

* For **Julian** (date) data items, they refer to the earliest and latest date that may be entered in the field.

* For **Numeric** and **Dollars** data items, they refer to the smallest amount (including negative numbers) and largest amount that may be entered in the field.

For most fields, Series 20 will automatically pick the correct minimum and maximum values to use (its defaults). These are:

* For **Alphabetic** or **Alphanumeric** fields, a minimum of zero (0) characters and a maximum equal to the number of character positions in the field

* For **Julian** data items, the earliest and latest dates that Series 20 can represent.

* For **Numeric** or **Dollars** data items, the largest negative amount and largest positive amount that can be stored by Series 20 in a field that size.

However, there will be data items where you want to restrict the entered data to a much narrower range. For these data items, you enter the data item **type**, and the minimum and maximum values to be used, separated by commas, when Series 20 prompts you for the data item definition.

Example:

Suppose that the STATE data item on the CARDFILE database **must** have a 2-character State abbreviation entered. You can specify a minimum of 2 characters, maximum of 2 characters, and required field, with the data item definition:

STATE: **AR, 2, 0, 2 <RETURN>**

Example:

Suppose that an AMOUNT data item must be greater than or equal to $0, but less than $5000. These minimum and maximum values can be set with the definition:

AMOUNT: **$, 0.00, 5000.00**

DEFAULT VALUES

If a data item in a database has one value that is more common than all the others, or has a value that should be "filled in" on the form when no value is available, you should define that value as the **default** value for that data item. Later, when you are entering data into the database, you can merely strike the **RETURN** key when you come to the data entry field for that data item, and Series 20 will **automatically** insert into place on the form the default value. You specify a default value by listing it after the minimum and maximum values, separated by a comma, when Series 20 asks for a data item definition. The examples show several typical situations where "default" values can be used.

Example:

In the CARDFILE database, suppose that the value "N/A" is to appear in the TITLE data item whenever a title is not available. You can define "N/A" to be the default value for TITLE with the data item definition:

TITLE: **AN, 0, 0, 20, "N/A"**

(Remember, the Ø and 2Ø specify a minimum of zero and maximum of 2Ø characters for the title). With this definition, pressing the **RETURN** key when entering data into the TITLE field will display "N/A" in the field automatically.

Example:

Suppose the AMOUNT data item in a database is frequently zero, and we want to be able to enter a zero simply by pressing the **RETURN** key when it comes time to enter the AMOUNT field. The definition:

AMOUNT: **$,,,Ø.ØØ**

will make Ø.ØØ the default amount. Notice that if the automatically-selected minimum and maximum values are satisfactory, you can "skip over" those parts of the data item definition by simply typing the commas, as shown here. (If it weren't for the commas, the Ø.ØØ would be taken as a minimum value.)

For additional information on default values for data items and how they are used in data entry, see Chapter 4.

CHAPTER SUMMARY

* Defining a database is the first step in using a new database.

* A database is defined with the **DEFINE** command:

  DEFINE database-name

* The database form is painted on the HP 125 screen with the cursor keys, editing keys, and typewriter keyboard. Data item names are enclosed in square brackets, and data entry fields are indicated with a series of underscores.

* Each data item is further defined by telling Series 20 what type of data it contains. Data items can be alphabetic, alphanumeric, numeric, dollars-and-cents, or dates.

* A data item can be required or optional. If it is required, Series 20 will force the user to enter a value for that data item when data is entered.

* Minimum and maximum values may be optionally specified for each data item. Series 20 will not allow data outside this range to be entered into the database.

* Default values may be optionally specified for each data item, to speed data entry.

CHAPTER 4

ENTERING AND UPDATING DATA

After a database has been defined, you may enter data into it. Until the data is entered, the database is empty. This chapter describes the use of the **ENTER** command to enter new records into a database. The **UPDATE** command, which is used to modify data that is already stored in a database, is also described.

DATA ENTRY

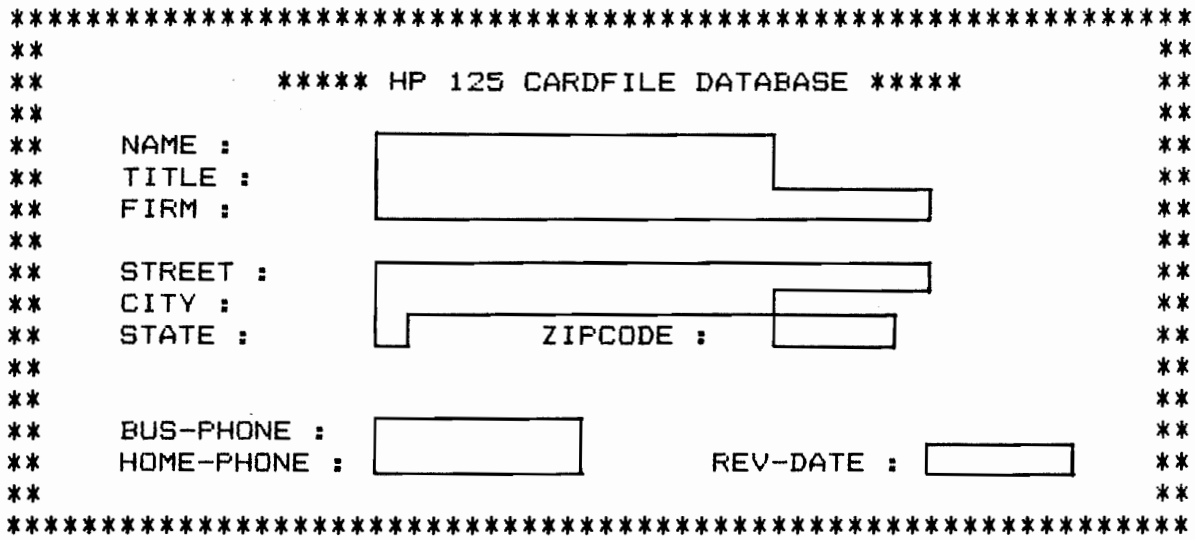
Entering data into a Series 20 database is as simple as filling out an ordinary business form. Just as you might use a pen to "fill in the blanks" of a paper form (such as an order form or purchase order), you use the HP 125 keyboard to "fill in the blanks" of a Series 20 database form.

To enter information into a database, type the command:

ENTER database-name

The form for the database will be displayed on the screen. The name of each data item in the database will appear on the form, followed by an inverse-video field. This "data entry field" is the place where you will "fill the blank" with data.

Figure 4-1 shows the database form for the example CARDFILE database:



4-1. CARDFILE Data Entry Form

To fill in the form, just type! When you reach the end of a field, the cursor will automatically move to the next field. You can also use the HP 125 editing keys to edit data in the form:

TAB key -- moves forward to the next field in the form

BACK TAB key -- moves backward to the previous field in the form

RETURN key -- same as TAB key; moves forward to the next field

BACKSPACE key -- backspaces one character

Arrow keys -- move the cursor up, down, left and right within the form

Normally, you will enter data into the form, field by field, until the entire form is completed. You can signal completion of the form in several different ways:

-- Typing into the last character position of the last field in the form.

-- Pressing the **<TAB>** or **<RETURN>** key to exit the last field of the form

-- Pressing the **END** softkey, which is always displayed while entering data into a database form

DATA ENTRY SOFTKEYS

When the form is completed, Series 20 will display a new set of softkey labels on the screen. Figure 4-2 shows these softkeys.

Figure 4-2. Data Entry Softkeys

Stop to review the data for accuracy at this point. Then, use the softkeys to select what will happen next:

Press the **PRINT** softkey to get a printed copy of the form and data. After the form is printed, the softkeys are displayed again, for another choice.

Press the **REVISE** softkey if the data is incorrect, and you wish to correct it. The data may be revised by moving the cursor to the character(s) in error, and overtyping. When all corrections have been made, press the **END** softkey, and the softkey choices in Figure 4-2 will be displayed once again.

Press the **DELETE** softkey if the data is incorrect, and you wish to "start again" with a clean form. The form will be cleared, and the data will **not** be added to the database. You may now re-enter the data.

Press the **CONTINUE** softkey if the data is correct, and you wish to continue entering data. The new record will be added to the database, and the form will be cleared. You may continue, entering the next record.

Press the **ABORT** softkey if the data is incorrect, and you wish to stop entering data now. The data in the form will **not** be added to the database. The screen will clear, and in a few seconds, the command prompt will appear.

Press the **END** softkey if the data in the form is correct, and this is the last record you wish to enter at this time. The new record will be added to the database, and the form will disappear from the screen. After a few seconds of "cleanup", the command prompt will appear, ready for your next command.

Example:

To enter data into the CARDFILE sample database defined in the last chapter, follow the following procedure:

Type **B:** in response to the A>> prompt
(this specifies drive B: as the current drive)
Type **ENTER CARDFILE <RETURN>**
(the CARDFILE form will appear on the screen.
Type **Jones, John <RETURN>**
(this will fill in the NAME field)
Type **President <RETURN>**
(this will fill in the TITLE field)
Type **U. S. Distributing, Inc. <RETURN>**
(this will fill in the FIRM field)
Type **3456 Johnson Ave. <RETURN>**
(the street address)
Type **Santa Ana <RETURN>**
(the city)
Type **AZ**
(the state -- note that the cursor automatically moves to the next field)
Type **83637**
(the zip code)
Type **4237483736**
(the business phone)
Type **6372637482**
(the home phone)
Type **04/02/82**
(the revision date)

This completes entry of the first record. The softkey labels of Figure 4-2 will appear at the bottom of the screen. Press **CONTINUE** to add the record to the CARDFILE database. The form will be cleared, waiting for other records to be entered. Try entering other data, and try the **REVISE, PRINT** and other softkeys. When you have entered four or five records, press the **END** softkey to complete data entry.

RANGE CHECKING

When a database form is defined, you may optionally specify a range of acceptable values for each data item. For example, you might restrict an amount to be less than $5000, or restrict a date to be between March 1 and June 30. As data is entered into the form, it is checked to be certain that it falls within the acceptable range. If you try to enter data that is outside the acceptable range, an error message is displayed, and the data is not accepted.

Normally, if you try to enter data that is outside of the acceptable range, you are making an error, and you will want to re-enter the data correctly. However, you can manually override the range checking, and force Series 20 to accept data that is outside the acceptable range. This is done by pressing the **RANGE OVERRIDE** softkey when the "data out of range" error message appears, and then re-entering the data. **YOU SHOULD USE THIS FEATURE WITH GREAT CAUTION, SINCE ENTERING OUT-OF-RANGE DATA MAY RESULT IN ERRORS WHEN DATABASE COMPUTATIONS OR COMPARISONS ARE ATTEMPTED LATER.**

DEFAULT DATA VALUES

When a database form is defined, you may optionally set up a "default" value for each data item on the form. Usually, the default will be the most commonly-entered value for that item. For example, if 90 percent of the people in the CARDFILE database will have the same city, state and zip code, it may be helpful to have these values set up as "default values".

When entering data, you may enter the default value for any field by pressing **<RETURN>** at the beginning of the data item field. The default value, if there is one, will be displayed in the field, and the cursor will move to the next field.

Example:

> Suppose the CARDFILE database form had been defined with default values for CITY (Santa Ana), STATE (AZ) and ZIPCODE (83637). Then, while entering the CARDFILE record in the preceding example, you could simply press **<RETURN>** when it came time to enter the city, state and zipcode fields. The default values would be automatically displayed and used for each of the three fields.

THE AUTO-DEFAULT FEATURE

In some data entry applications, it is useful to have the default value for each data item appear in the form **before** beginning data entry for each record. This allows you to see the defaults while entering data, and either accept them, or type over them to enter other data. This "AUTO-DEFAULT" feature may be selected at any time by pressing the **AUTO-DEFAULT** softkey that is displayed during data entry. Whenever the "AUTO-DEFAULT" feature is in operation, "Auto-Default" will appear at the bottom of the screen. Pressing the softkey again will turn off the "AUTO-DEFAULT" feature.

Example:

> If the CARDFILE form is defined with defaults for the CITY, STATE and ZIPCODE data items, then pressing the **AUTO-DEFAULT** softkey would cause these defaults to appear each time the form is displayed or cleared for entering a new record. Usually, the defaults will be the correct data, so you may simply skip the fields, saving data entry time. When the defaults are not the correct data for a particular record (e.g. a person is from Los Angeles), you may type over the default values to correct them.

THE AUTO-REPEAT FEATURE

In some data entry applications, much of the data that is entered for one record will be the exactly the same for the following record, and the one following, etc. In this case, it is very useful if the form is **not** cleared when preparing to enter the next record. Rather, the data from the previous record is allowed to remain in the form. Only those few data items that change from record to record need to be entered (by typing over the information from the previous record).

This feature of Series 20 is termed "AUTO-REPEAT", since the data from each data record is automatically repeated in the form for the next record. The "AUTO-REPEAT" feature can be selected at any time during data entry by pressing the **AUTO-REPEAT** softkey. "Auto-Repeat" will appear at the bottom of the screen, indicating the "AUTO-REPEAT" feature is in operation. To turn off the AUTO-REPEAT feature, press the softkey again.

Example:

Suppose your firm has just acquired a new customer, and thirty people from the firm are to be added to the CARDFILE database. All the people have exactly the same firm name, address, business phone, etc. The only data that varies from person to person is the name, title and home phone.

AUTO-REPEAT mode will be very useful here. Enter the first person into the database normally, and then press the **AUTO-REPEAT** softkey. For each of the other 29 people, you can enter the name and title, **TAB** to the home phone field, and you have finished data entry! Each of the other fields (firm, city, state, etc.) will still be filled with the data from the preceding record, which is the same for all the records. In this way, data entry time for the new records will be cut dramatically.

The AUTO-DEFAULT and AUTO-DEFEAT features are mutually exclusive -- both may not be selected at the same time. Pressing either the **AUTO-DEFAULT** or **AUTO-REPEAT** softkey will automatically turn off the other feature (if it was turned on).

UPDATING DATA

What happens when someone in the CARDFILE database has a new telephone number or changes address? The **UPDATE** command lets you retrieve the appropriate record from the database, display the data in the database form, revise the data as required, and then put the revised record back into the database.

To retrieve database records for updating, type the **UPDATE** command:

UPDATE database

Series 20 will ask you for the "search condition" for the update -- that is, "which records would you like to update?" The simplest form of search condition takes the form:

data-item IS value

Examples:

NAME IS "JONES, J."
(find the record for J. Jones)

FIRM IS "HEWLETT-PACKARD"
(find all of the records for HP employees)

REV-DATE IS 04/01/82
(find records revised on April 1)

CITY IS SEATTLE
(find the records for people from Seattle; notice that quotes are not needed when there are no embedded blanks)

Series 20 will search the database for records that match the search condition. If no records in the database match the search condition, Series 20 will print a message to that effect, and ask for another search condition.

If a record is found that matches the search condition, it will be displayed on the screen, in the database form. Note that more than one record may meet the search condition. (In the example above, perhaps more than one person in the CARDFILE database works for Hewlett-Packard). The first record found in the database that matches the search criteria is displayed.

At this point, you may review the data displayed in the form. The softkey labels in Figure 4-3 are displayed on the screen.

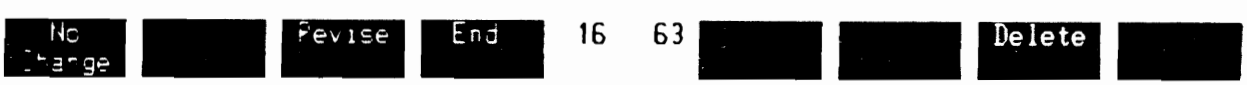


Figure 4-3. Update Softkey Labels

Use the softkeys to select what to do next:

Press the **NO CHANGE** softkey if you decide to leave the data unchanged, as it is displayed in the form.

Press the **REVISE** softkey if you wish to edit or modify the data in the form. You may then use the HP 125 editing keys to edit the displayed data, character by character, until it is correct. When the data is correct, press the **END** softkey. The softkeys in Figure 4-2 will be displayed, and you may print the revised record, revise it further, delete the record, etc. using these keys. Pressing the **CONTINUE** softkey at this point continues the UPDATE operation.

Press the **DELETE** softkey to delete the displayed record from the database completely. The record will be marked as "deleted", and the data in the record will no longer be accessible. Use the **DELETE** softkey with caution! If a record is deleted accidentally, you will have to re-enter all the data; there is no way to recover the deleted information.

Press the **END** softkey to stop the UPDATE operation. If you have made revisions to the data in the form, the revised record will be replaced in the database. The form will disappear from the screen, and Series 20 will ask for another selection criteria.

If more than one record in the database meets your search condition, pressing the **CONTINUE** or **DELETE** softkey automatically brings the **next** record meeting the search condition into the form. This continues, record by record, until no more records meet the search condition. Series 20 will then ask you for another search condition.

If you wish to perform another search, and update another set of records, enter a new search condition, and repeat the steps above. If you are finished updating the database, press the **RETURN** key when Series 20 asks for a new search condition. The screen will clear, and the command prompt will request the next command.

Example:

Suppose that the telephone number for U.S. Distributing (whose President was added to the CARDFILE database in the previous example) has changed. We wish to change it in the database:

Type **B:** in response to the A>> prompt
(select drive B: for processing)
Type **UPDATE CARDFILE**
(Series 20 will ask for a search condition)
Type **FIRM IS "U. S. Dist*"** (the form will appear, with the record of a U.S. Distributing employee displayed)
Press the **REVISE** softkey
(to revise the record)
TAB to the **BUS-PHONE** field
(change the number by typing over it)
Press the **END** softkey
(indicating the end of the revisions)
Press the **CONTINUE** softkey
(the revised record is replaced in the database)

Repeat the procedure with the next record displayed in the form. Experiment with the **REVISE**, **DELETE** and **NO CHANGE** softkeys as well.

SEARCH CONDITIONS

Search conditions for the **UPDATE** command can become quite complex. In addition to testing for a match with a certain value, we can ask for records with a greater than/less than relationship as well. The table lists the tests that can be done to select records:

Relationship	Code	Meaning
Equals	**EQ**	Search for data item values that are identical to the specified value
Less Than	**LT**	Search for data item values that are less than the specified value
Less Than or Equals	**LE**	Search for data item values that are less than or equal to the specified value
Greater Than	**GT**	Search for data item values that are greater than the specified value
Greater Than or Equals	**GE**	Search for data item values that are greater than or equal to the specified value
Inequality	**IS NOT**	Search for data item values that are not equal to the specified value

In addition to these relationships, compound search conditions can be created by combining simple search conditions with an "AND" or "OR". However, "AND" and "OR" conditions cannot be combined in a single search condition. Extensive abbreviation is also possible; see Appendix A for complete details.

Examples:

NAME LT "B"
(selects records with names that begin with "A"

REV-DATE GE 02/01/82
(selects records revised since February 1, 1982)

FIRM IS "HEWLETT-PACKARD" AND DATE GT 02/01/82
(selects records that meet both criteria)

As the examples show, quotes (") may be used in search conditions to clarify the search for alphabetic data. Quotes **must** be used when the data item value being searched for includes embedded spaces or special characters. Thus, most names, addresses, etc. must be enclosed in quotes when used in search conditions.

Experienced Series 20 users may "shortcut" the dialogue by including the search condition directly in the **UPDATE** command itself, following the word **"WHERE":**

Example:

```
UPDATE CARDFILE WHERE NAME IS "JOHNSON, R."
UPDATE CARDFILE WHERE DATE LT 01/01/82
UPDATE CARDFILE WHERE FIRM IS "HP" AND NAME IS "JONES,
JOHN"
```

Search conditions are used in many Series 20 commands. Whenever a search condition is used, the rules and examples given here apply.

DELETING UNWANTED RECORDS

You can use the **UPDATE** command to delete a small number of unwanted records from a database. Use the search condition to select the records to be deleted. Then, as each record appears in the form, choose either the **NO CHANGE** softkey to keep the record, or the **DELETE** softkey to delete it. Series 20-2 also provides a **DELETE** command for deleting large numbers of records automatically. This command is discussed in Chapter 6.

CHAPTER SUMMARY

* Database data is entered using the **ENTER** command. Data entry and update both use the database form to display data, a record at a time.

* As each record is entered, softkeys are used to:

  CONTINUE to the next record,

  PRINT the record,

  REVISE the record,

  ABORT the record, or

  END data entry

* Database update takes place through the **UPDATE** command.

* Records are selected for update through a **search condition.** The database is searched, and records matching the condition are displayed in the database form, a record at a time, for potential revision.

* The **UPDATE** command provides a **DELETE** softkey, which deletes the displayed record.

CHAPTER 5

SIMPLE INQUIRIES AND REPORTS

Series 20's inquiry and reporting features let you obtain the data you need from your databases, on demand. Series 20 can help you with everything from simple inquiries (e.g. "What is the address for U.S. Distributing?") to very complex reporting needs ("Please summarize sales trends by product line for our 20 largest customers over the last six months"). The next several chapters describe this full spectrum of inquiry and reporting capabilities. This chapter begins with the simplest inquires.

SIMPLE DATABASE INQUIRY

The simplest form of Series 20 database inquiry occurs when all the information to answer a question is contained in a single database. In the CARDFILE database, examples would include:

What is John Jones' address and phone number?
Which clients live in Arizona?
Who are the salesmen from U.S. Distributing?

You already know one simple way of answering this kind of question -- with the **UPDATE** command of the last chapter! To pose a simple inquiry:

1) Type **UPDATE database-name**
   (Select the database that answers the inquiry)

2) Enter a **search condition**
   (Use the search condition to pose your question)

3) **Review** the displayed record(s)
   (Your answer comes in the form of a database record that matches the search condition, displayed in the database form.)

4) Press the **NO CHANGE** softkey
   (If several records match the search condition, the NO CHANGE softkey brings each record successively into the form for viewing.)

5) Press the **END** softkey
   (When you have viewed all the records, the **END** softkey will complete the inquiry.)

6) Enter another search condition
   (Series 20 gives you the opportunity for another inquiry in the same database. If you do not want to make another inquiry, press **RETURN**.

Example:

To see all the salesmen from U. S. Distributing, enter the commands:

UPDATE CARDFILE

when Series 20 asks for a search condition, type:

TITLE IS Salesman AND FIRM IS "U. S. Distributing"

The CARDFILE record for each person meeting the search condition will be displayed on the screen, in succession.

THE DISPLAY COMMAND (Level II)

For Level II users, the **DISPLAY** command provides the same inquiry functions as the **UPDATE** command, without the danger of accidentally changing the database contents. The command is entered as:

DISPLAY database

Like the **UPDATE** command, the **DISPLAY** command searches the database for all records that meet a "search condition". Series 20 asks for the search condition immediately after you enter the **DISPLAY** command. Identical rules apply for search conditions for both the **DISPLAY** and **UPDATE** commands. Here are some examples of search conditions:

NAME IS "JONES, JOHN"
(John Jones' form would be displayed)

STATE IS AZ
(Several records would qualify, and you could browse through them)

STATE IS NOT AZ
(The opposite inquiry from the preceding example)

ZIPCODE LT 02999
(Zipcodes less than 02999 are selected)

STATE IS NY AND FIRM IS "W.E.I."
(Select people from W.E.I. in New York)

FIRM IS "W.E.I." OR FIRM IS "R.X.I."
(Select W.E.I. or R.X.I. employees)

As the examples show, you have a great deal of flexibility to tailor the database inquiry to your exact needs. Search conditions can also be combined with **AND** and **OR**, to build up more complex search criteria. (Note: **AND** and **OR** cannot both be used in the same search condition).

After you enter the search condition, Series 20 will display the database form on the screen, and begin searching the database for records matching the condition. The first qualifying record is displayed in the form, and the softkey labels in Figure 5-1 are displayed.

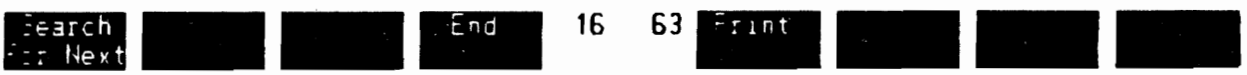


Figure 5-1. Display Command Softkeys

Use the softkeys to select what will happen next:

Press the **SEARCH FOR NEXT** softkey to continue searching the database; the next record matching the search condition will be found and displayed in the form.

Press the **PRINT** softkey to print the form and the displayed record on the system printer.

Press the **END** softkey to stop the search.

When the **END** softkey is pressed, or when Series 20 can find no more records that match the search condition, the screen is cleared. The command prompt appears, and Series 20 is ready to accept the next command.

APPROXIMATE MATCHES IN SEARCH CONDITIONS

In the examples thus far, search criteria have been based on an **exact match**, character by character between data item values and the value supplied in the search criteria. What happens if we wish to retrieve the CARDFILE record for a person named Robertson, but have forgotten his first name? Using an exact match, we cannot -- if we enter a command searching for **NAME IS "ROBERTSON"**, and the name is stored as **"ROBERTSON, A."** in the database, no match will be found, and the record will not be retrieved.

Series 20 solves this problem with its **approximate match** feature. Anywhere that you might enter a character as part of a search condition, you may also enter one of two "wild card" characters -- a question mark ("?") or an asterisk ("*"). These characters function just like "wild cards" in a card game -- **they will match any other character.** The question mark and asterisk work in slightly different ways:

> **Question Mark:** A question mark matches any other **single character** in the same position. For a match to occur, the number of question marks must be exactly the same as the number of matching characters.
>
> **Asterisk:** An asterisk matches any **character or group of characters** in the same position. A single asterisk can match one or many matching characters.

Examples:

Search condition: **NAME IS RICHARDSON???**

Matches:
RICHARDSON, J
RICHARDSON,JB

Does not match:
RICHARDSON, ANDREW
RICHARDSON, J.B.

Search Condition: **NAME IS RICHARDSON***

Matches:
RICHARDSON, J
RICHARDSON,JB
RICHARDSON, ANDREW
RICHARDSON, J.B.

Characters following an asterisk are ignored. For example,

Search Condition:	**NAME IS RICH*EW**
Matches:	RICHARDS, EW RICHARDSON, EW RICH, ANDREW
And also matches:	RICHARDS, E.W. RICHELIEU

An asterisk in the first position will match every record in the database.

Approximate matches can be used with both the **DISPLAY** and **UPDATE** commands, as well as several other Series 20 commands. In fact, with only a few exceptions (noted under individual commands in Appendix A), an approximate match can be used anywhere in a Series 20 command! Approximate matches can be very useful when searching for names, street names, etc., where exact spelling may be unknown. They are also useful for selecting records based upon just part of a data item (e.g. select the "VP-Sales", "VP-Finance", "VP-Manufacturing", etc. by searching for **TITLE IS "VP*"**).

DATABASE REPORT FORMATS

Simple database inquiries may be effectively answered by selecting records and viewing them in the database form, one record at a time. But what happens if you need to look at data from many different records at once? A printed report, organized into the familiar rows and columns, is often the most convenient way of looking at this type of information.

With Series 20's extensive reporting capabilities, you can organize data the way you need it, generating many different kinds of reports. Reports can be printed on the printer, or displayed on the screen, on demand. There are five basic Series 20 report formats:

Screen Format:	Records are printed in the database form; one record per form.
Columnar Format:	Records are printed in a row/column report; one row per record; each column displays one data item.
Columnar Format with Statistics:	Columnar format, with totals, subtotals, averages, etc.

Summary Format: Selected data items are tabulated (totalled, counted, etc.); summary statistics are printed.

Statistical Report: Summary statistics (totals, counts, averages) are printed for the entire database.

Examples of each type of report, and instructions on how to print it, are given in the sections that follow. There are three steps to generating any report from a Series 20 database:

1). Make sure that the database contains all the data needed for the report. (Chapter 6 tells how to combine data from different databases to generate more complex reports).

2). Sort the database into the order in which you want it to appear in the report (using the **SORT** command.)

3). Print the actual report. (Using the **PRINT, LIST, TABULATE, STAX** and/or **TITLE** commands)

SORTING A DATABASE

Databases are sorted into order for a report using the **SORT** command. The database records are sorted based on the values of the data items named in the **SORT** command:

SORT database BY data-item1, ... data-item32 [option]

Up to 32 data items can be used for sorting (for example, to sort a database by state, and within each state by city, and within each city by firm name, etc.). If more than one data item is used, the first data item listed becomes the major sort item, with each successive data item being sorted within the preceding items. An ascending sort is normally used (A before B, 1 before 2, etc.). To sort into descending order instead, the **[D]** option of the SORT command is used (see the examples).

Examples:

SORT CARDFILE BY NAME
(Sorts into ascending order by name)

SORT CARDFILE BY FIRM, STATE, CITY
(Sorts into ascending order by firm, and within each firm, by state, and within each state, by city)

SORT CARDFILE BY DATE [D]
(Sorts into descending order by date, placing most recent dates first)

The **SORT** command may require some time to complete the sorting operation, especially on large databases. Progress messages are displayed on the screen as the sort operation proceeds.

Note: While sorting a database, Series 20 requires temporary working space. The **SORT** command obtains this space by creating a temporary disc file **on the current disc drive.** This file will be equal in size to the database being sorted, and is automatically erased when the sort is complete.

If you are sorting a large database, you should check to make sure there is enough space on the current disc drive to accommodate this working space. If there is not enough space, insert a blank disc into a spare disc drive, and make that drive the current disc drive before beginning the sort. After sorting is complete, you can return to your previously-used current disc drive.

SCREEN FORMAT REPORT

Printing a report in screen format is very simple. First, be certain the database is sorted into the order you wish for printing. Then, enter the command:

PRINT database

Records from the database will be printed in the database form, on the system printer.

Example:

Entering the commands:

```
SORT CARDFILE BY NAME
PRINT CARDFILE
```

will produce the screen format report in Figure 5-2.

```
**********************************************************
**                                                      **
**          ***** HP 125 CARDFILE DATABASE *****        **
**                                                      **
**    NAME :        Anderson, Harold E.                 **
**    TITLE :       Mfg. Rep.                           **
**    FIRM :        Arizona Test Equipment              **
**                                                      **
**    STREET :      47 Harrison St.                     **
**    CITY :        Middletown                          **
**    STATE :       AZ      ZIPCODE :   84749           **
**                                                      **
**                                                      **
**    BUS-PHONE :   4634837473                          **
**    HOME-PHONE :  9384737483      REV-DATE :  04/20/82  **
**                                                      **
**********************************************************

**********************************************************
**                                                      **
**          ***** HP 125 CARDFILE DATABASE *****        **
**                                                      **
**    NAME :        Cartwright, Mary                    **
**    TITLE :       VP - Sales                          **
**    FIRM :        Interna[illegible] Freight C[illegible]  **
```

Figure 5-2. Screen Format Report

The softkey labels of Figure 5-3 are displayed during printing of the report, and allow control over the printing operation:

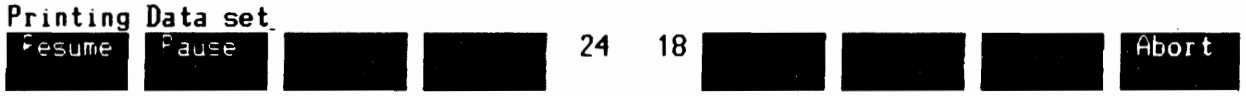


Figure 5-3. Report Control Softkeys

Press the **PAUSE** softkey to temporarily halt the printer. (The printer will complete the current record before stopping.)

Press the **RESUME** softkey to start printing again after pressing the **PAUSE** softkey.

Press the **ABORT** softkey to abort the report.

******* CAUTION *******

Care must be taken to make sure the your HP 125 is connected to a printer before entering any print routine. If you print to a non-existent HP-IB printer (Serial Printer or Thermal printer) you will have to hard reset the system. If you print to a non-existent serial printer (Letter Quality, or Internal Printer) control will be passed back to Condor.

SCREEN FORMAT REPORTS ON THE SCREEN

If you want to display a screen format report on the HP 125 screen instead of printing it on the system printer, use the **LIST** command:

LIST database

The report will look exactly the same as it would if printed on the printer. However, only a small part of the report can be displayed on the HP 125 screen at a time. The **LIST** command will display each record of the report on the screen, and then pause with the softkeys labels of Figure 5-4 displayed:

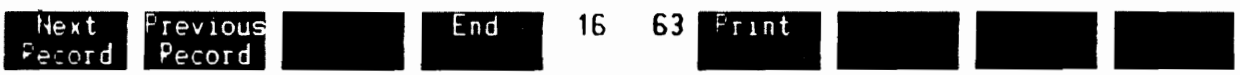


Figure 5-4. Screen Report Control Softkeys

Choose the next step by pressing the appropriate softkey label:

Press the **NEXT RECORD** softkey to see the next record of the report.

Press the **PREVIOUS RECORD** softkey to see the preceding record of the report

Press the **PRINT** softkey to print the currently displayed record on the system printer

Press the **END** softkey to end the report.

If you accidentally press **NEXT RECORD** when the last record of the report is displayed, or press **PREVIOUS RECORD** when the first record of the report is displayed, the **LIST** command responds with the softkeys in Figure 5-5.

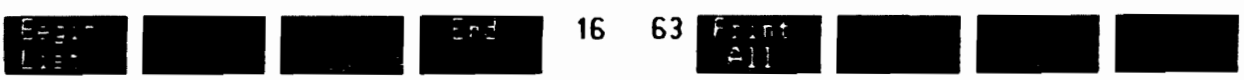


Figure 5-5. Screen Report Control Softkeys

These softkeys provide the following choices:

Press the **BEGIN LIST** softkey to begin the report, starting with the first record

Press the **PRINT ALL** softkey to print a report of all the records on the system printer. Note that this has the same effect as the **PRINT** command for screen format reports.

Press the **END** softkey to end the report.

COLUMNAR REPORTS

Columnar reports are the common row/column reports that are used in business every day. To print a columnar report, first sort the database into the proper order for the report. Then enter the PRINT command:

PRINT database BY data-item1, data-item2, ...

Records from the database will be printed in sequence, each record appearing as one row of the report. The columns of the report are the selected data items from the **PRINT** command, appearing left to right, as they are listed in the command.

Example:

We want to print a report of the CARDFILE database, sorted geographically. The report is to show the STATE, CITY, FIRM and NAME of each person in the database. Entering:

```
SORT CARDFILE BY STATE, CITY
PRINT CARDFILE BY STATE, CITY, FIRM, NAME
```

will produce the columnar report in Figure 5-6.

STATE	CITY	FIRM	NAME
AZ	Middletown	U. S. Distributing, Inc.	Williams, Jeff
AZ	Middletown	Arizona Test Equipment	Harris, Bill
AZ	Middletown	Arizona Test Equipment	Jameson, Paula
AZ	Middletown	Arizona Test Equipment	Anderson, Harold E.
AZ	Phoenix	Arizona Test Equipment	Smith, John W.
AZ	Santa Ana	U. S. Distributing, Inc.	Jones, John
AZ	Santa Ana	U. S. Distributing, Inc.	Harrison, Jacob
AZ	Tucson	International Freight Co.	Robertson, Michael
AZ	Tucson	International Freight Co.	Kline, Paul J.
AZ	Tucson	International Freight Co.	Cartwright, Mary

Figure 5-6. Columnar Report

While the report is printing, the softkey labels in Figure 5-3 provide the same printing control as described under "Screen Format Reports" above.

The headings above each column in the report will be the data item names. You can prevent these headings from being printed by specifying the **[X]** option of the **PRINT** command.

Example:

```
PRINT CARDFILE BY STATE, CITY, FIRM, NAME [X]
```

COLUMNAR REPORTS WITH STATISTICS

Often, a columnar report will be more useful if the report contains totals, subtotals, averages, etc. Series 20 will automatically compute and print these statistics, if you request them in the **PRINT** command.
The statistics that can be computed are:

TOTAL -- The total of a column is printed
MIN -- The minimum value in a column is printed
MAX -- The maximum value in a column is printed
AVERAGE -- The average of a column is printed
STAX -- All the above statistics are printed

Statistics can be accumulated for one or more columns in a report. More than one type of statistic can also be requested for a given column in the report.

To generate a columnar report with statistics, first sort the database into the proper order for printing. Then, enter the **PRINT** command:

```
PRINT database BY data-item1, data-item2, ...
AND COMPUTE statistic  data-item3, data-item4 ...
```

Remember, enter the command as one long command--only press **<RETURN>** at the end of the command.

Examples:

To total the orders in an **ORDERS** database, enter:

```
PRINT ORDERS BY ORDER.NO, DATE, AMOUNT AND COMPUTE TOTAL
AMOUNT
```

The report in Figure 5-7 will be generated:

ORDER.NO	DATE	AMOUNT
56473624	12/01/81	456.56
35264537	12/01/81	56.45
64736453	12/02/81	50.00
56384637	12/03/81	47.75
46374673	12/03/81	100.00
Total		710.76

Figure 5-7. Sample Columnar Report

The **"AND COMPUTE..."** phrase of the **PRINT** command can request several different statistics, and name several different data items for each different type of statistic. For example:

```
PRINT ORDERS BY ORDER.NO, AMOUNT, ORDER.DATE, TAX
AND COMPUTE TOTAL AMOUNT, TAX AVERAGE AMOUNT
```

will produce the report in Figure 5-8.

ORDER.NO	AMOUNT	DATE	TAX
56473624	456.56	12/01/81	56.56
35264537	56.45	12/01/81	6.45
64736453	50.00	12/02/81	.00
56384637	47.75	12/03/81	7.75
46374673	100.00	12/03/81	10.00
Total	710.76		80.76
Average	142.15		

Figure 5-8. Sample Columnar Report with Statistics

In addition to totaling columns at the end of a report, you can request that subtotals be computed during the report. Subtotals will be printed each time the value of a specified data-item changes, when you add the phrase:

SUBTOTAL USING data-item

to the end of a **PRINT** command.

Example:

```
PRINT ORDERS BY ORDER.NO, DATE, AMOUNT, TAX
AND COMPUTE TOTAL AMOUNT, TAX SUBTOTAL USING DATE
```

will produce the report in Figure 5-9.

ORDER.NO	DATE	AMOUNT	TAX
56473624	12/01/81	456.56	56.56
35264537	12/01/81	56.45	6.45
Subtotal			
		513.01	63.01
64736453	12/02/81	50.00	.00
Subtotal			
		50.00	.00
56384637	12/03/81	47.75	7.75
46374673	12/03/81	100.00	10.00
Subtotal			
		147.75	17.75
Total		710.76	80.76

Figure 5-9. Sample Columnar Report with Subtotals

If you request subtotals in the **PRINT** command, the subtotals will normally be **non-cumulative.** You can request **cumulative** subtotals, by specifying the **[A]** option on the command. As with columnar format reports, the **[X]** option will suppress column headings.

Example:

```
PRINT ORDERS USING ORDER.NO, DATE, AMOUNT, TAX
AND COMPUTE TOTAL AMOUNT SUBTOTAL USING DATE [A]
```

will print the report as above, but with each day's order subtotal showing cumulative order dollars for the month, rather than the subtotal for that day only.

As you can see, PRINT commands can be quite lengthy. Extensive abbreviation is possible. See the description of the PRINT command in Appendix A for details.

COLUMNAR REPORTS ON THE SCREEN

Columnar format reports, with or without statistics, can be displayed on the HP 125 screen as well as printed on the printer. To produce the columnar reports on the screen, use the **LIST** command instead of the **PRINT** command:

```
LIST  database  BY data-item1, data-item2, ...
```

The report will appear on the HP 125 screen exactly as it would on the printer. All the options of the **PRINT** command for columnar format reports and columnar format reports with statistics also apply to the **LIST** command.

SUMMARY REPORTS

In the previous section, you saw how to print subtotals on a columnar report. If you are only interested in subtotal or statistical information (and not the detail report itself), then a Summary Report can be used.

Summary reports are generated with the **TABULATE** command:

```
TABULATE database  BY data-item1, data-item2 ...
AND COMPUTE  statistic  data-item3, data-item4  [option]
```

This command displays, on the screen, the statistics requested, for the data items listed. The data in the database **must be sorted** into sequence by data-item1, data-item2, etc. before the **TABULATE** command is given.

Example:

To tabulate how many people in the CARDFILE database are from each city and state, enter:

```
SORT CARDFILE BY STATE, CITY
TABULATE CARDFILE BY STATE, CITY
```

The resulting report is shown in Figure 5-10.

STATE	CITY	{Count}
AZ	Middletown	4
AZ	Phoenix	1
AZ	Santa Ana	2
AZ	Tucson	3
CA	Los Angeles	1
CA	San Francisco	1
CN	Toronto	1
CO	Denver	1
GA	Atlanta	1
IL	Chicago	1
MA	Boston	1
NY	New York	1
Total		18

Figure 5-10. Sample Summary Report

Example:

To tabulate orders by day, enter:

```
TABULATE ORDERS BY DATE AND COMPUTE TOTAL AMOUNT
```

The resulting report is shown in Figure 5-11.

DATE	AMOUNT {Subtotal}
12/01/81	513.01
12/02/81	50.00
12/03/81	147.75
Total	710.76

Figure 5-11. Sample Summary Report

The statistics available through the **TABULATE** command are the same as those for the **PRINT** command (**TOTAL, MIN, MAX, AVERAGE**). As with the columnar reports, the **[A]** option can be used to tabulate cumulative subtotals.

STATISTICAL REPORTS

If summary statistics for an entire database are needed, a statistical report will provide all the necessary information. The statistical report is requested with the **STAX** command:

STAX database BY data-item1, data-item2 ... [option] ..

A variety of statistical information can be compiled through the options of the command:

[A] -- calculate averages only for selected data items
[C] -- calculate countnt only for selected data items
[M] -- calculate only the minimums & maximums
[T] -- calculate totals only for selected data items
[P] -- send the output to the printer

Examples:

To display total and average order statistics on your screen, enter:

STAX ORDERS BY AMOUNT [T] [A]

To print the same statistics to your printer, enter:

STAX ORDERS BY AMOUNT [T] [A] [P]

and generate the report in Figure 5-12.

```
                AMOUNT

Total
                710.76
Average
                142.15
```

Figure 5-12. Sample Statistical Report

If no option is specified, all the available statistics are printed for each data item.

PRINTING SUMMARY AND STATISTICAL REPORTS

The **TABULATE** and **STAX** commands normally display their results on the screen. To print the results on the printer instead, use the **[P]** options of the commands. For example:

```
TABULATE ORDERS BY ORDER.DATE AND COMPUTE TOTAL AMOUNT
[P]
```

Prints the report of Figure 5-12 on the printer instead of the display.

In addition to displaying reports on the screen and printing them on the printer, the **TABULATE** command can place a summary report into a **database.** Specifying the **[S]** option of the **TABULATE** command causes it to create a special database, whose records correspond to the rows of the Summary Report, and whose data items correspond to the report's columns. The database generated is the **RESULT** database, which is described fully in Chapters 6 and 9.

REPORT TITLES

Titles can be printed on Series 20 reports by using the **TITLE** command:

```
TITLE  "...text..."
```

The text (enclosed in quotes) will appear at the top of each page of the report when it is printed.

Example:

```
TITLE "Outstanding Orders Summary Report"
```

More elaborate titles, with page numbering, dating, multi-line titles, and pagination, are available through various command options:

P or **PAGE** means print page n, where n is the current page
L or **LINE** means carriage return-line feed
D or **DATE** means print date set by the Condor DATE command
T or **TOP** means do form feed after nn lines
S or **SKIP** means skip a line, i.e. insert blank line
nn = number of lines between titles (default is 66)

For example,

```
TITLE "...text...",PAGE,"...text..."
```

causes the page number to appear in the midst of the title line, in exactly the position where the word **PAGE** appears in the command.

```
TITLE "...text...",DATE,"...text..."
```

similarly causes the date to appear in the midst of the title line.

```
TITLE "...text1...",LINE,"...text2..."
```

causes a line feed after printing text1, for a two-line title.

```
TITLE TOP, nn
```

causes a top-of-form after **nn** lines. Use this option to control the number of printed lines per page of the report -- from 2 to 99 lines per page are allowed.

```
TITLE SKIP, nn
```

causes a skip of **nn** lines between every line. This allows you to perform double spacing (when **nn** = 2), triple spacing (when **nn** = 3), etc.

The various options of the **TITLE** command can be combined in a single command, leading to intricate control over report titling.

Examples:

The commands:

```
TITLE DATE,"  Daily Order Report  ",PAGE
TITLE TOP, 66, "Order Analysis",LINE,"By Product Line"
```

produce the titles shown in Figure 5-13

```
Ø1/26/83  Daily Order Report  Page 1

Order Analysis
By Product Line
```

Figure 5-13. Report Title Examples

CHAPTER SUMMARY

* The **UPDATE** and **DISPLAY** commands are used for simple database inquiry; each uses a search criteria and displays the matching records in the database form

* Screen-format reports display each record in the database form, and are produced by the commands:

  PRINT database
  LIST database

* Columnar reports display data in row/column format, and are produced by the commands:

  PRINT database BY data-item1, ...
  LIST database BY data-item1, ...

* Statistics (totals, averages, etc.) may be added to columnar format reports with the clause:

  AND COMPUTE statistic data-item ...

* Statistical and summary reports suppress the report detail, and provide summary information only. They are produced with the:

  TABULATE database BY data-item
  STAX database BY data-item

* The **TITLE** command is used to control printed report formats, and to print report titles.

CHAPTER 6

RELATIONAL DATABASE CAPABILITIES

For many questions that arise in daily business use, a simple inquiry or report, like those in the last chapter, will provide the needed information. For this kind of inquiry, all the information needed to provide an answer is located within a single database. But you will also have inquiries that require information from several different databases to be combined, related and compared to arrive at an answer.

Series 20 excels at this kind of complex database manipulation. It is a **relational** database management system, which means that data from many different databases can be related and combined on demand. These relational database management capabilities are the topic of this chapter. Some of the commands are available only with Condor 20-2 or 20-3. These commands will be noted.

THE RESULT DATABASE

Many of the database commands in this chapter combine data from two databases to create a third database. This database contains the **results** of the database operation (such as a merge, selection, etc.), and is appropriately named the **RESULT** database. Figure 6-1 shows the role of the **RESULT** database:

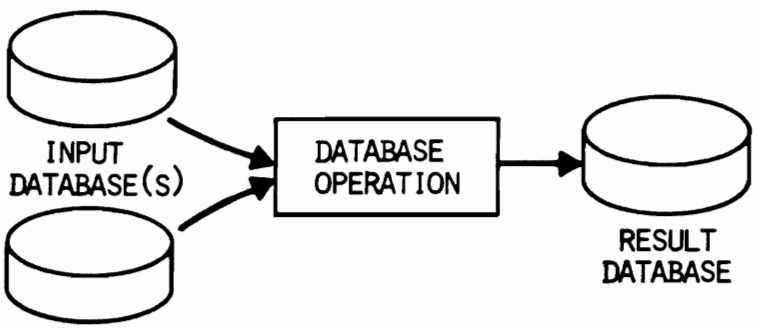


Figure 6-1. The RESULT Database

The **RESULT** database is always created on the **current disc drive.** It is **automatically** created by commands that produce a **RESULT** database.

Creating the RESULT database on a disc requires that a "Data Dictionary" already exists on that disc. This will automatically be true if any database has been defined or copied onto the disc. Chapter 9 fully explains the Data Dictionary and its function.

SELECTING DATABASE RECORDS

One of the most common types of inquiries is "show me all the records that meet the following criteria ...". For example, you may want to select all the people in the **CARDFILE** database who live in Texas; or select all parts in an **INVENTORY** database where stock-on-hand is less than six months' supply.

This type of selection is done with the **SELECT** command:

SELECT database

The **SELECT** command is very similar to the **UPDATE** and **DISPLAY** commands. Like those commands, it asks for a search condition, and searches the database for records that match the condition. But with the **SELECT** command, the records that match the search condition are placed in the **RESULT** database, so that you may process them further. Figure 6-2 illustrates the SELECT command.

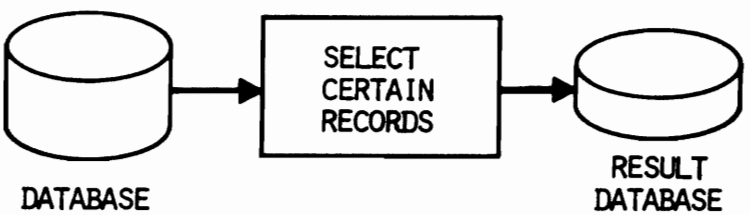


Figure 6-2. The SELECT Command

You can shorten the **SEARCH** command dialogue by including the search condition in the command, following the word **"WHERE"**:

SELECT database WHERE search-condition

Example:

SELECT CARDFILE WHERE STATE IS AZ
(records for Arizona residents are copied to the RESULT database)

PRINT RESULT BY NAME, CITY, STATE
(print the RESULT database)

The rules for search conditions in the **SELECT** command are the same as those for the **DISPLAY** and **UPDATE** commands (see Chapter 4).

SAVING THE RESULT DATABASE

Whenever you use a command that creates a **RESULT** database, the former contents of the **RESULT** database are replaced and lost. If you want to preserve the **RESULT** database, and return to it for later processing, you must save its contents. The **SAVE** command lets you save the **RESULT** database under another name:

SAVE database

The **RESULT** database is copied into the database named in the command. Figure 6-3 illustrates the effect of the **SAVE** command.

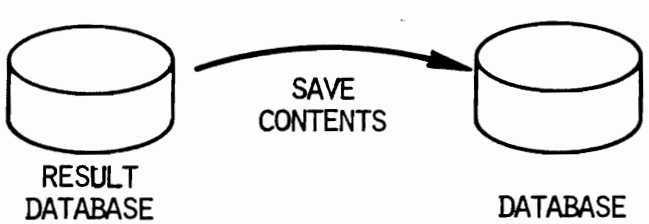


Figure 6-3. The SAVE Command

The **SELECT** and **SAVE** commands are often used together for complex inquiries that ask "what if" questions. A common inquiry technique is to successively narrow the search condition until an acceptably small number of records is found, then review the records selected. The following example illustrates this data-base inquiry technique.

Example:

Suppose we wish to search an EMPLOYEE database, to find employees eligible for the position of Manager of Engineering. Candidates should meet the following qualifications:

Mandatory conditions:
BS MS degree
Management experience

Desirable conditions:
MSEE degree
Not currently at or above salary level
5 or more years experience with company

First find those matching the mandatory conditions:

```
SELECT EMPLOYEE WHOSE DEGREE IS BS* OR DEGREE IS MS*

Data Base: EMPLOYEE 2820 Records.
Busy
Total records in Result Set = 152
```

Two **SELECT**s must be used because **AND** and **OR** cannot be combined in the same search condition:

```
SELECT RESULT WHOSE POSITION > 1999

Data Base: RESULT 152 Records
Busy
Total records in Result Set = 51
```

We assume position codes 2000 and above are management. Note that the RESULT database is used as the source of the search, and is overwritten with the result of the search. Candidates meeting the mandatory conditions are saved in case succeeding searches yield too few candidates and we wish to backtrack:

```
SAVE POTENTL
SELECT POTENTL WHOSE SALARY < 40000

Data Base: POTENTL 51 Records
Busy
Total records in Result Set = 30
```

Close, let's try for MSEE's:

```
SELECT RESULT WHOSE DEGREE IS MSEE

Data Base: RESULT 30 Records
Busy
Total records in Result Set = 2
```

Probably the best candidates. Find out who they are:

```
LIST RESULT BY NAME, DEPT
```

Now, backtrack to get secondary candidates:

```
SELECT POTENTL WHOSE SALARY < 40000 AND DEGREE IS BS?E

SELECT RESULT WHOSE HIRE.DATE < 6/1/76

LIST RESULT BY NAME, DEPT
```

As the example shows, the **SELECT/SAVE** combination allows powerful ad hoc inquiry, which can explore many different paths before reaching its conclusion.

COMBINING TWO DATABASES

Data from two databases can be combined together into a single database, if both databases have the same structure (identical number of fields and record sizes). For example, clients who attended each of two different seminars may have been registered into two different databases, and you can combine the two databases to analyze the entire group of attendees.

Three commands permit the combining of records in two databases: **APPEND**, **COMBINE**, and **JOIN**.

THE APPEND COMMAND

The **APPEND** command adds the records of one database to another:

```
APPEND  database1  database2
```

Both databases must have an **identical** structure; i.e. they must contain the same number of data items, in the same order, with the same lengths. The records of database 2 are appended (added) to database1. Database2 remains unchanged by the command. Figure 6-4 illustrates the command.

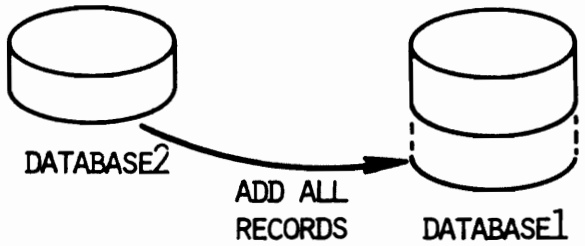


Figure 6-4. The APPEND Command

Example:

Orders for the day are contained in an ORDERS database. At the end of the day, these order records must be added to the other order records for the month, in the month-to-date orders database (MTDORDER), from which month-to-date reports are generated. The command:

```
APPEND MTDORDER ORDERS
```

appends the day's orders to the month-to-date database.

THE COMBINE COMMAND (Condor 20-2 or 20-3 only)

The **COMBINE** command is similar to the **APPEND** command, but it leaves the data in both databases undisturbed. Instead, databases are combined into the **RESULT** database. Figure 6-5 illustrates the **COMBINE** command:

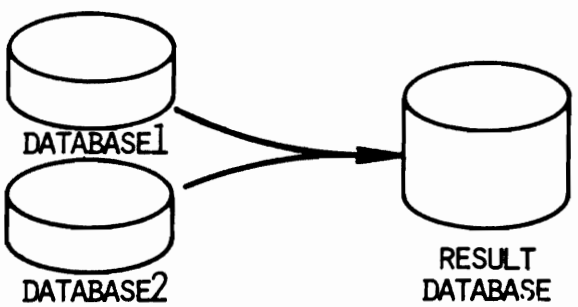


Figure 6-5. The COMBINE Command

The **COMBINE** command has the same form as the **APPEND** command:

COMBINE database1 database2

Again, both databases to be combined must have the **identical** structure. When the command is complete, the **RESULT** database will have the data from both databases, and the database form and data item definitions from database1.

Example:

People who attended two recent seminars were registered using into two databases, named SEMINAR1 and SEMINAR2. Now, you want to generate statistical reports on total seminar attendance. The command:

COMBINE SEMINAR1 SEMINAR2

generates a **RESULT** database, merging the two databases, for analysis.

THE JOIN COMMAND (Condor 20-2 or 20-3 only)

The **APPEND** and **COMBINE** commands combine databases with identical structure (i.e. both databases have the same data items, which are defined the same way). The **JOIN** command is used to combine selected data from two databases which do **not** have the same structure.

Consider the following example:

> Regional order data is collected during the month in a REGIONS database, which also includes the Region Manager's name. At month-end, you wish to generate a mailing to the region managers, detailing their order performance. The REGIONS database does not contain a mailing address for the Region Managers. However, the CARDFILE database **does** contain their addresses. To generate the mailing, you need to combine the address from the CARDFILE database with the order data from the REGIONS database.

The **JOIN** command is designed for exactly this purpose.

The **JOIN** command operates by matching selected data items from the two databases to be combined. These data-items are called the "matching items", and are listed in the command:

JOIN database1 database2 MATCHING data-item, ...

Example:

```
JOIN REGIONS CARDFILE MATCHING NAME
     (data from the REGIONS and CARDFILE databases
      will be combined)
```

The JOIN command searches both databases, looking for a record in database1 and a record in database2 whose "matching items" match. (In the example, the command searches for a CARDFILE record and a REGIONS record that both contain the same NAME.) Each time the **JOIN** command finds such a pair of records, a new record is written to the **RESULT** database. The new record contains all the data items of database1 **and** all the data items of database2. (The matching fields, since they have the same value in both databases, are not repeated twice, but will only appear once in the **RESULT** database.)

Figure 6-6 illustrates the JOIN command.

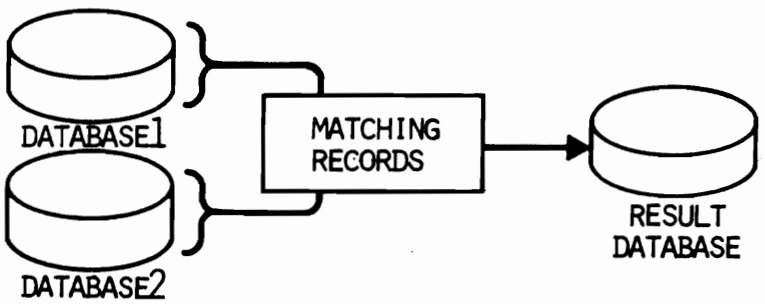


Figure 6-6. The JOIN Command

In the example above, the **RESULT** database would contain all the data items from the CARDFILE database (Street, City, Home Phone, etc.) and all the data items from the **REGIONS** database (Name, Order Dollars, etc.). Some of the data items may not be needed for subsequent processing. The **PROJECT** command (described in the next section) can be used to discard unneeded data items.

Example:

A report showing monthly orders by product and product line is required. The **TABULATE** command has already been used to create a **RESULT** database with one record per product, containing the required statistics. But product names and the product line code for each product is contained in a separate **PRODUCTS** database. The command:

JOIN RESULT PRODUCTS MATCHING PRODUCT.NO

will produce a new **RESULT** database that includes the required information. The required report can be generated from this database.

PROJECTING A DATABASE (Condor 20-2 or 20-3 only)

When processing large databases, the time it takes to process Series 20 commands depends on **how many records** are in the database, and on **how much information** is contained in each database record. Large records require more time to process (and consume more space on the disc) than small records.

Often, only a few of the data items in a database are needed to answer an inquiry. The **PROJECT** command can be used to select only those data items, eliminating the others from the record. The resulting database (with fewer data items) is written to the **RESULT** database. Figure 6-7 illustrates the effects of the **PROJECT** command:

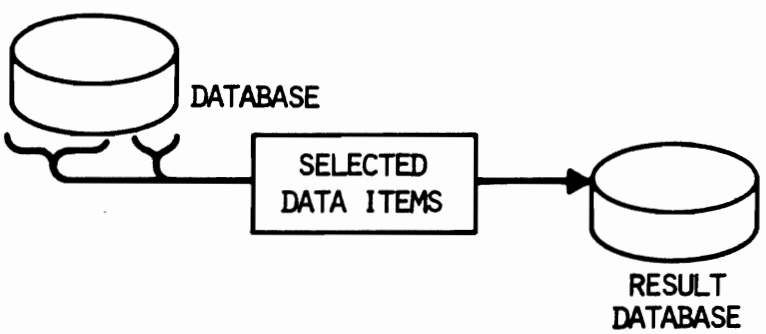


Figure 6-7. The PROJECT Command

The data items that are to be included in the **RESULT** database are named in the **PROJECT** command:

PROJECT database BY data-item1, data-item2, ...

Example:

An **ORDERS** database contains quite a lot of information about each order -- Firm Name, Ship-to and Billing Addresses, etc. To product statistical reports on order dollars, order types, and order dates, it will be much more efficient to work with a database containing only that information. The command:

PROJECT ORDERS BY AMOUNT, DATE, ORDER.TYPE

creates a (much smaller) **RESULT** database that contains only the data items needed for the analysis. Subsequent **SELECT**, **SORT** and **PRINT** commands will require less disc space and operate faster as a result.

Often several **JOIN** and **PROJECT** commands will be used in sequence to produce a **RESULT** database that is needed for a report. The following example illustrates this type of inquiry.

Example:

A report showing all the orders for the day, by customer, is required. The report is to show, for each order, the order number, the customer name, the product ordered, and the order amount. Subtotals are to be computed for each customer.

Three separate databases hold all the required information:

The **ORDERS** database contains, for each order:

```
-- ORDER.NO       (Order number)
-- CUST.NO        (Customer number)
-- PRODUCT.NO     (Product ordered)
-- QUANTITY       (Quantity ordered)
-- UNIT.PRICE     (Unit Price)
-- AMOUNT         (Amount of purchase)
   - etc -
```

The **CUSTOMER** database contains, for each customer:

```
-- CUST.NO        (Customer number)
-- FIRM           (Company name)
-- NAME           (Name of contact)
   - etc -
```

The **PRODUCTS** database contains, for each product:

```
-- PRODUCT.NO     (Product number)
-- PRODUCT.LINE   (Product line code)
-- DESCRIPTION    (Product description)
    - etc -
```

The following sequence of commands will produce the desired report:

```
JOIN ORDERS CUSTOMER BY CUST.NO
    (Join these two databases to get a customer name
     for each customer number in the ORDERS database)

PROJECT RESULT BY ORDER.NO, FIRM, PRODUCT.NO, AMOUNT
    (Discard unneeded data items from the join operation
     with this PROJECT command)

JOIN RESULT PRODUCTS BY PRODUCT.NO
    (Join again, this time to get a description of each
     product ordered from the PRODUCTS database)

PROJECT RESULT BY ORDER.NO, FIRM, DESCRIPTION, AMOUNT
    (Again, discard unneeded data items to make the
     following SORT go as rapidly as possible)

SORT RESULT BY FIRM
    (Sort into order for printing)

TITLE "***** Daily Orders Summary *****"
    (Title the report)

PRINT RESULT BY ORDER.NO, FIRM, DESCRIPTION, AMOUNT
AND COMPUTE TOTAL AMOUNT SUBTOTAL USING FIRM
    (Finally, print the report)
```

DATABASE ARITHMETIC

Data item values within a database can be changed in many different ways. The **UPDATE** command (described in Chapter 4) allows you to change values, one-by- one, for selected records in a database. Updating an entire database in this way would be very tedious, however. The **COMPUTE** and **CHANGE** commands allow you to change a data item value throughout an entire database with one command.

THE COMPUTE COMMAND

The **COMPUTE** command calculates a new value for a data item in every record of a database. The data item to be computed, and the formula for computing its new value, are both given in the command:

COMPUTE database ST data-item = expression

For each record in the database, the **COMPUTE** command performs the computation indicated, and assigns the resulting value to the data-item. The expression in the command must be an algebraic expression using addition (+), subtraction(-), multiplication (*), and division (/).

Examples:

```
COMPUTE ORDERS ST  EXTENSION = QUANTITY * PRICE
COMPUTE ORDERS ST TAX = SUBTOTAL * 6/100
```

The individual variables that are added, subtracted, multiplied and divided must be either numeric constants (e.g. 100) or numeric other data items from the same record. **THE EXPRESSION IS ALWAYS EVALUATED FROM LEFT TO RIGHT, OPERATION BY OPERATION.** While parentheses may be used to improve readability, **PARENTHESES ARE COMPLETELY IGNORED** in the computation. For this reason, it is probably best to not include parentheses in expressions. Dollars-and-cents constants in the expression should always have a decimal point and two decimal places. Constants or data items without decimal points are treated as cents when combined with dollars-type data items (i.e. 1067=$10.67). Refer to the COMPUTE command in Appendix A if you would like more details.

THE CHANGE COMMAND (Condor 20-2 or 20-3 only)

The **CHANGE** command also changes the value of a data item in each record of a database. In this case, the new value is not computed, but is merely given in the **CHANGE** command itself. For example, you might want to change the date in each record of an ORDERS database to today's date. The **CHANGE** command lists the data item to be changed, and its new value:

CHANGE database ST data-item IS value ...

The word "ST" in the command is an abbreviation for "SUCH THAT".

Examples:

```
CHANGE ORDERS ST DATE IS Ø2/15/82
    (the date for all orders is set to Ø2/15/82)
CHANGE EMPLOYEE ST STATUS IS ACTIVE
    (every employee is set on active status)
CHANGE EMPLOYEE ST STATUS IS ACTIVE AND DATE = Ø2/15/82
    (two changes are made in one command)
CHANGE EMPLOYEE ST STATUS1 IS @STATUS2
    (Status1 is assigned the value of Status2)
```

As the examples show, several data items can be changed in a single **CHANGE** command simply by repeating the **"data-item IS value"** phrase with the word **"AND"**. An equals sign can be used an abbreviation for the word **"IS"**.

The last example shows how one data item can be assigned the value of another data item in the same record. The "at sign" (**@**) **is used to tell Series 2Ø that STATUS2 is a data item name, and that the value of this data item is to be assigned to the data item named STATUS1. (Note that if the @** were not used, the command would cause STATUS1 to be assigned the value "STATUS2" as a literal, seven-letter word.)

THE POST COMMAND

The **COMPUTE** and **CHANGE** commands perform their arithmetic within each individual database record. Data item values within the record determine the new value for the data item being changed. Sometimes, data item values must be changed based on information from **outside** the database.

For example, suppose we were accumulating regional month-to-date order statistics in a REGIONS database, with one record per sales region. As each individual order is received during the day, it is entered into an ORDERS database. Each order includes information identifying the sales region from which it came. How can the REGIONS database be updated at the end of the day to reflect the day's orders?

The **POST** command performs this updating function. Using the **POST** command, records from one database (usually referred to as the "transaction" database) can be **POSTED** to another database (usually referred to as the "master" database), altering the contents of the master database. In the example, we would **POST** the **ORDERS** database to the **REGIONS** database, updating regional order statistics.

Figure 6-8 illustrates the operation of the **POST** command.

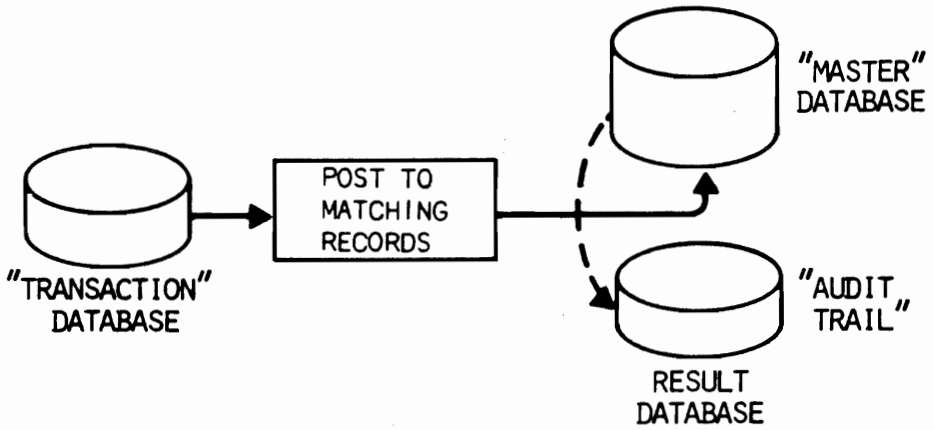


Figure 6-8. The POST Command

For each record in the "transaction" database, the **POST** command:

1. Finds the matching "master" database record, using one or more "matching fields". **THE MATCHING FIELDS MUST HAVE THE EXACT SAME STRUCTURE (same data type, length, etc.).** In our example, the "matching field" is the REGION, which determines which master database record will be posted.

2. **POST**s the "master" database record, as requested in the **POST** command. Several different data items in the "master" database can be updated with a single **POST** command.

The **POST** command can perform several different kinds of posting:

ADD -- the data item value from the "transaction" record is added to the corresponding data item in the "master" database record. You would use this option to accumulate total orders, for example.

SUBTRACT -- the data item value from the "transaction" record is subtracted from the corresponding data item in the "master" database record. You would use this option to process withdrawals from a total inventory count, for example.

REPLACE -- the data item value from the "transaction" record replaces the corresponding data item value in the "master" database record. You would use this option to replace the "last update" date in a master database, for example.

The databases, matching fields, posting operation, and data items for posting are all named in the **POST** command:

POST database1 database2 MATCHING data-item1, ...
posting-operation data-item2 ...

Database1 is the "master" database, and **database2** the "transaction" database for the posting operation. **Data-item1** specifies the matching field. The posting operation may be any of those listed above; the operation is carried out on the data-item(s) listed as **data-item2.**

Example:

Daily orders from the ORDERS database are to be posted to the REGIONS database. The command:

POST REGIONS ORDERS MATCHING REGION ADD AMOUNT

causes each order record to be matched against the correct regional record in the REGIONS database. The AMOUNT from the ORDERS record will be added to the AMOUNT in the REGIONS record. At the completion of the **POST** command, the REGIONS database is up-to-date with the day's new orders

Example:

Withdrawals from a PARTS inventory database are entered into a TRANSACT database during the day. At the close of business, the PARTS inventory must be updated, and reports generated for ordering new parts. The command:

POST PARTS TRANSACT MATCHING PART-NO SUBTRACT QUANTITY

causes each TRANSACT record to be matched with the master PARTS inventory record for the part withdrawn. The quantity on hand is decreased by the quantity of parts withdrawn. At the end of the **POST** command, the PARTS database has been updated to reflect the day's withdrawals.

You may use more than one matching field by listing several data-items, separated by commas, following the word **"MATCHING"** in the command. If several matching fields are listed, **all** the fields must match for the posting to occur. Similarly, the posting-operation may be applied to more than one data-item. You can also combine different posting options in the same command.

Examples:

POST PARTS WITHDRAW MATCHING PART-NO, BIN-NO SUBTRACT QUANTITY
(two matching data items)

POST PARTS WITHDRAW MATCHING PART-NO SUBTRACT QUANTITY, REPLACE DATE
(two posting operations)

The **POST** command creates a **RESULT** database as a by-product of the **POST**ing operation. Before a "master" database record is updated, it is copied to the **RESULT** database. The **RESULT** database thus serves as an audit trail of the **POST**ing operation.

The **POST** command must search the "master" database for **each** record of the "transaction" database, which can be a time-consuming task. The command will operate much more rapidly if both the "master" and "transaction" databases are sorted into sequence by matching fields before **POST**ing.

For more examples of the **POST** command, see the General Ledger example in Chapter 11.

COMPARING TWO DATABASES

Two databases can be compared to one another using the Series 20 **COMPARE** command. The compare command lists the individual data-items that are to be compared, record by record:

COMPARE database1 database2 MATCHING data-item, ...

The **COMPARE** command works in the following way:

1. A record of database1 is read
2. A search is made through database2, to find a record whose data-items match those of the database1 record. All the data-items listed in the **COMPARE** command must match in order for the database2 record to be considered a "match".
3. If a matching record is found, the **database1 record** is added to the **RESULT** database.

4. Steps 1-3 are carried out for each record in database1.

5. At the end of the **COMPARE** command, the **RESULT** database will contain those database1 records for which matching database2 records were found.

Figure 6-9 illustrates the **COMPARE** command.

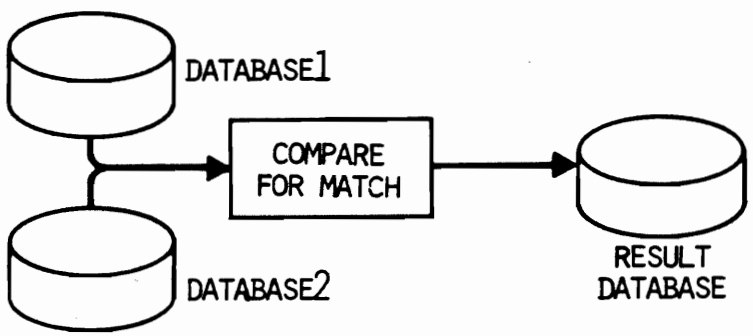


Figure 6-9. The COMPARE Command

Example:

Two databases of prospective customers have been produced from two different sources. We wish to compare the databases, to find duplicates. The command:

COMPARE LIST1 LIST2 MATCHING NAME, FIRM

will produce a **RESULT** database containing all the records in LIST1 that had a duplicate record in LIST2 (i.e. the same NAME and FIRM).

The **COMPARE** command can also be used to find the records in a database that do **NOT** match any records in another database:

COMPARE database1 database2 NOT MATCHING data-item, ...

The **COMPARE...NOT MATCHING** command follows exactly the same five-step process described for the **COMPARE** command. However, the records placed in the **RESULT** database will be the records from database1 for which **no** matching database2 record can be found.

Example:

Daily orders have been entered into an ORDERS database, and a data-item in each record tells from which sales region the order came. Before posting the REGIONS database with the ORDERS, we should insure that only valid regions have been entered into the ORDERS database records. The command:

COMPARE ORDERS REGIONS NOT MATCHING REGION

will place in the **RESULT** database any ORDERS record which does not have a matching region in the REGIONS database. These records should be corrected before posting.

Note that both databases in a **COMPARE** command must contain the data-items to be compared, and the data items must be defined the same way (data item name, type and size) in both databases. However, the other data items in the two databases may be quite different.

The order of the databases in the COMPARE command is also important. Note that

COMPARE ORDERS REGIONS MATCHING REGION

is not the same as

COMPARE REGIONS ORDERS MATCHING REGION

In the first example, records from the ORDERS database will be placed in the **RESULT** database; in the second, records from the REGIONS database will be placed in the **RESULT** database.

More examples of the **COMPARE** command can be found in the General Ledger example of Chapter 11.

DELETING DATABASE RECORDS (Condor 20-2 or 20-3 only)

Records of a database can be deleted one-by-one through the **UPDATE** command. However, if many records are to be deleted, this is a time consuming process. The **DELETE** command lets you delete many records from a database with a single command:

DELETE database WHERE search-condition

The command searches the database named for records that match a search condition. Only those records that match the search condition are deleted. The search-condition follows the same rules described for the search-conditions in the **UPDATE, DISPLAY** and **SELECT** commands (see Chapter 4)

The **DELETE** command has far-reaching effects, since it can delete many records at a time, and the data in deleted records is no longer accessible. Care should therefore be used when using the **DELETE** command.

Note that deleted records are logically deleted, but they will still occupy space in the database, and will still be included in the record count for the database. Sorting the database will collect together all the space occupied by the deleted records, making it available for other records, and decreasing the record count accordingly.

Examples:

DELETE CARDFILE WHERE REV-DATE LT 01/01/75
(deletes CARDFILE records not revised since 1974)

DELETE CARDFILE WHERE STATE IS MI
(deletes Michigan residents from the CARDFILE database)

CHAPTER SUMMARY

* The relational commands in Series 20 allow multiple databases to be related "on the fly" to respond to even complex inquiries.

* The **SELECT** command selects specific records from a database and places them in the **RESULT** database.

* The **SAVE** command saves the **RESULT** database for later use.

* The **APPEND** command adds the records of one database to another.

* The **COMBINE** command combines two databases into a **RESULT** database.

* The **JOIN** command matches records from two databases, based upon matching fields, and combines their data items in a **RESULT** database.

* The **PROJECT** command creates a **RESULT** database that contains only selected data items from the original database.

* The **CHANGE** command changes a data-item value in every record of a database.

* The **COMPUTE** command computes a new value for a data-item from an algebraic expression.

* The **POST** command posts "transaction" records to a "master" database, updating selected data items.

* The **COMPARE** command compares two databases, placing in the **RESULT** database the records that do or do not match.

* The **DELETE** command deletes selected records from a database.

CHAPTER 7

DATABASE UTILITIES

In everyday use, there are several "housekeeping" functions that are critical for managing and maintaining your databases. Functions such as database backup, copying a database from one disc to another, switching among several different discs with databases on them, etc., are an important, if mundane part of daily database use. This chapter describes the "utility" commands that help you perform these functions.

CHANGING DISCS

After a short while, it is likely that you will have databases on several different flexible discs. During a single session with Series 20, you may wish to process several of these databases. The **LOGDISK** command allows you to switch discs in an orderly fashion.

> ******* CAUTION *******
>
> You must NEVER remove a disc from the disc drive while Series 20 is processing a database command. REMOVING A DISC WHILE SERIES 20 IS READING OR WRITING DATA ON IT CAN RESULT IN LOSS OF DATA, AND MAY RESULT IN LOSS OF ALL THE DATABASES ON THE DISC

The **only** time you should remove and insert discs when using Series 20 is when the command prompt:

A>>, B>>, C>>, etc.

appears on the screen. When this prompt appears, Series 20 is waiting for a command from you, and is not accessing any of the discs.

At this point, you may remove a flexible disc, and insert another flexible disc in its place. After switching discs in this way, you must inform Series 20 that a new disc is present, by "logging in" the disc through the **LOGDISK** command:

LOGDISK drive:

Example:

You have removed the flexible disc in drive B:, and replaced it with another. Enter the command:

LOGDISK B:

to have Series 20 recognize the new disc.

Failure to properly "log in" the new disc can produce erroneous results and error messages from Series 20.

LISTING DATABASE NAMES

A listing of all the databases that are defined on a disc may be obtained through the **DIC**tionary command:

DIC drive:

This command may be especially useful to find the name of a database that you had forgotten, or to locate unused databases that should be destroyed. The names of all the defined databases on the requested drive will be displayed on the screen. If you wish to see a database listing for the current disc, you do not have to type the drive name in the command.

Examples:

DIC D:
(lists databases on drive D:)
DIC
(lists databases on the current disc)

The information displayed comes from the "Data Dictionary", and includes the database name ("title") for each database, along with the names of the corresponding form file, definition file, and data file. (Chapter 9 describes the Data Dictionary and the role of these component files).

LISTING DISC CONTENTS

The names of the files stored on a disc can be listed using the **DIR** ("directory" command:

DIR drivename:

This command performs the same function as the CP/M directory command. The names of the files on the requested disc drive are listed on the screen, in four columns. Both the filename and the extension, if any, are listed, for each file.

If the drivename is not given in the command, a directory listing for the current disc drive is displayed. Note that the directory command deals directly with **files** on the disc, and not with databases. Thus, each database stored on the disc will be listed as its component files (form file, data file, etc.). See Chapter 9 for a description of these files.

CHANGING THE NAME OF A DATABASE

The name of a database can be changed with the **RENAME** command:

RENAME newname = oldname

This command changes the name of the database, and renames all its associated files to reflect the new name.

Examples:

```
RENAME  CUSTOMER = CLIENTS
RENAME  MTDACCTS = DAYACCTS
```

COPYING A DATABASE

There are several reasons why it may be necessary to copy a database from one disc to another. Frequently-used databases stored on flexible discs should regularly be copied to fresh diskettes, to prevent data loss from disc wear. Critical databases should also be copied frequently, with the "backup" copy stored in a safe place. Or, a copy of a database may be produced on a flexible disc, to be mailed to another HP Series 100 user for processing.

A duplicate copy of a database is made with the **COPY** command:

COPY newdatabase = olddatabase

Examples:

```
COPY E:ORDERS = B:ORDERS
    (copy the orders database from drive B: to E:)
COPY E:ORDR29 = B:ORDERS
    (copy to a database with a different name)
COPY E:ORDR29 = E:ORDERS
    (duplicate database on the same disc)
```

DESTROYING UNWANTED DATABASES

When a database is no longer needed, you can destroy it with the **DESTROY** command:

```
DESTROY database
```

The **DESTROY** command frees all the disc storage that the database required, for use by other databases or files. After a database has been destroyed, the data in the database is no longer accessible. The form and data item definitions for the database are also destroyed, and are no longer accessible.

Example:

```
DESTROY MYDATA
    (destroys the database named MYDATA)
```

The **DESTROY** command has drastic effects! Series 20 will ask you to confirm that you actually want to destroy the database named in the command before carrying out the destruction. Use the command with care.

EMPTYING A DATABASE

From time to time, you may need to purge a database of all its contents, but keep the database form and data item definitions. For example, a database that contains month-to-date order statistics must be cleared of its data as part of the end-of-month processing, to prepare for the following month. You can "empty" a database of its data with the **EMPTY** command:

```
EMPTY database
```

The **EMPTY** command logically deletes all the records from a database. The data is no longer accessible following the command. However, unlike the **DESTROY** command, the database form and data item definitions for the database are preserved.

Example:

```
EMPTY ORDERS
    (deletes all records in the ORDERS database)
```

Because of its drastic effects, confirmation is required before the **EMPTY** command is carried out.

SETTING THE CURRENT DATE

Series 20 maintains a "current date" for printing the date in report titles, and for comparing and manipulating dates in databases. You may set the "current date" through the **DATE** command:

```
DATE  mm/dd/yy
```

This changes the "current date" to the date shown. You can also use the **DATE** command to view the current date, by entering:

```
DATE
```

without typing a new date. Series 20 will display its "current date", and then request a new date from you. At this point, you may either enter a new date, or enter a **<RETURN>**, in which case the "current date" remains unchanged.

Examples:

```
DATE  12/25/81
DATE  1/2/82
DATE
```

PRINTING REPORTS TO A DISC FILE

Normally, when you instruct Series 20 to print a report (using the **PRINT** command, for example), the report is printed on the system printer. The printer used is selected with the HP 125 configuration menu, as described in your system **Owners Manual**. You can ask Series 20 to print its reports to a disc file instead, with the command:

```
SET PRINTER  filename
```

After entering this command, all subsequent printed output will be written in the named file instead of being printed on the printer. To return printed output back to the printer, enter the command without a filename:

SET PRINTER

The **SET PRINTER** command may be especially useful for reports that must be edited (e.g. with Series 100/Word) before being printed. It also allows printed reports to be incorporated into documents, transmitted to a host system electronically, etc. See Chapter 8 for information on using the **SET PRINTER** command to pass information from Series 20 to other applications.

SETTING SERIES 20 OPERATING CHARACTERISTICS

Several features of Series 20 operation can be selectively switched on and off by the user, for convenience in database processing. These features are selected through the **SET** command, which has many different variations.

The **SET DATE** command controls how Series 20 will display and print dates. The available formats, and their corresponding commands, are:

SET DATE MDY -- dates appear as mm/dd/yy
SET DATE DMY -- dates appear as dd.mm.yy
SET DATE YMD -- dates appear as yy-mm-dd

The **SET ECHO** command controls the echoing of Series 20 commands and messages to the screen, especially during command procedure file processing. It has two forms:

SET ECHO ON -- commands and messages are displayed
SET ECHO OFF -- commands and messages are not displayed

The **SET ABORT** command controls whether a command procedure automatically aborts when an error occurs. Normally, the command procedure is aborted. (See Chapter 10 for a description of the Series 20 Command Procedure capability.) The **SET ABORT** command has two forms:

SET ABORT ON -- errors cause command procedures to abort
SET ABORT OFF -- command procedures continue despite errors

.e

The **SET AUTOSEL** command controls the system's search for its command files. Each time you enter a command, Series 20 searches for the corresponding "command file", which contains the HP 125 instructions needed to carry out the command. With the

"AUTOSELECTION" feature, Series 2Ø first searches the current drive, and failing to find the command file there, will search the master drive for the command file. Normally, the command file will be found on the master drive.

You may disable this feature (restricting the command file search to the current disc only) with the commands:

SET AUTOSEL ON -- Autoselection will occur
SET AUTOSEL OFF -- Autoselection will not occur

Normally, autoselection should be left ON. **Note:** you may override the entire command search by explicitly designating the drive on which the command file will be found as part of the command name (e.g. **C:LIST** executes the **LIST** command from the command file on drive **C:**).

The **SET MASTER** command allows you to set the "master drive" that will be used for the autoselection feature. Normally, the master drive is drive A:, which contains the Series 2Ø programs. Under unusual circumstances, you may switch to another master drive, with the command:

SET MASTER drive:

where **drive:** is the desired drive name.

STARTING SERIES 2Ø WITH A DBMS COMMAND

If you would like to start Series 2Ø without using the Welcome Menu you may use the **DBMS** command from the CP/M level. Simply wait for a CP/M prompt:

A>

and type,

DBMS

This will start Condor Series 2Ø just as if you had pushed the "Condor DBMS" softkey.

EXITING SERIES 20

After finishing a session with Series 20, you may exit with the **SYSTEM** command:

```
SYSTEM
```

This command causes an orderly shutdown of Series 20, and returns the system to CP/M control. If there is a WELCOME menu on the current disc, the WELCOME menu will be displayed, to choose the next application. Otherwise, the CP/M prompt:

```
A>
```

will appear, with the current disc indicated.

```
*****************************************************************
*                      ***** CAUTION *****                      *
*                                                               *
* Never leave Condor without using the SYSTEM command.  If      *
* you do, you may find that the edit keys on the upper two      *
* rows of your keyboard do not work properly.  To correct       *
* this, simply start Series 20 by by pushing the "Condor        *
* DBMS" softkey, and then exit Series 20 with the SYSTEM        *
* command.                                                      *
*****************************************************************
```

RUNNING NON-CONDOR 20 PROGRAMS

Occasionally, you may want to run a non-Series 20 program while in the midst of a Series 20 database management session. You may, of course, exit from Series 20, run the program, and then re-enter Series 20 control. However, you can also execute the program while under Series 20 control, by typing the program name, preceded by a dollar sign ($), in response to the Series 20 command prompt.

Examples:

```
$PIP      -- runs the CP/M utility PIP
$MYPROG   -- runs the application program MYPROG
$STAT     -- runs the CP/M program STAT
```

This capability is limited to small programs that can fit into HP Series 100 memory along with Condor's main program. Large programs that make use of all or nearly all the available user space on the HP Series 100 must be executed by exiting Condor.

CHAPTER SUMMARY

* Several utility commands provide "housekeeping" functions for Series 20 databases.

* The **LOGDISK** command is used when switching discs, to "login" the new disc.

* The **DIC** command lists all the currently-defined databases on a disc.

* The **DIR** command lists the files on a disc.

* The **RENAME** command is used to change the name of a database.

* The **COPY** command copies databases.

* The **DESTROY** command destroys a database, including its form and data item definitions

* The **EMPTY** command deletes all the records in a database.

* Series 20's current date is set with the **DATE** command.

* Other Series 20 operating characteristics are set with the **SET** command.

* The **SYSTEM** command exits Series 20, and returns to the Welcome Menu.

* CP/M programs can be run without exiting Series 20 by placing a dollar sign in front of their names, and entering the name as a command. (e.g. $PIP).

CHAPTER 8

USING DATABASES WITH OTHER APPLICATIONS

Condor databases can be used as a source of data for other HP Series 100 applications, from word processing to business graphics. For example, you might use a customer database to generate a customized mailing through Series 100/Word or Wordstar/100. You might use Series 100/Graphics to plot data from an orders database.

Data from other HP Series 100 applications can also be entered into Condor databases for processing and reporting. You might have a Basic program that generates data for database processing by Condor. Or, you may want to feed financial data from Visicalc/100 or MicroPlan/100 into Condor for tracking purposes.

This chapter shows how to exchange data with some of the other HP Series 100 applications.

PRINTING REPORTS TO A FILE

A simple way of passing data from Series 20 to other applications is by printing Series 20 reports into a disc file instead of printing them on the system printer. Once a report is stored in a disc file, it may be incorporated into a Series 100/Word or Wordstar/100 document, or transmitted to a host system via Link/125 or DSN/Link, etc. Printed output is directed to a disc file with the command:

SET PRINTER filename

All subsequent printed reports will be stored in the disc file instead of being printed on the printer. If the file extension is not specified in the filename, Series 20 will give it the extension **.PRT** (e.g. **SET PRINTER TEMP** goes to **TEMP.PRT**). To return to normal printing, enter the command:

SET PRINTER

The file is closed, and normal printing resumes.

Example:

To print a report to a disc file, for later transmission to a host system, enter the following command sequence:

```
SET PRINTER RPTFILE
    (the report will be stored in the file RPTFILE.PRT)
PRINT CARDFILE  BY NAME, FIRM, TITLE, DATE
    (the report is actually "printed" in the file)
SET PRINTER
    (return to normal printing)
```

The file RPTFILE.PRT can now be transmitted to a host via Link/125 or DSN/Link.

READ AND WRITE COMMANDS

The Series 20 **READ** and **WRITE** commands provide a more flexible way to exchange data between a database and other applications. Figure 8-1 shows how these commands work.

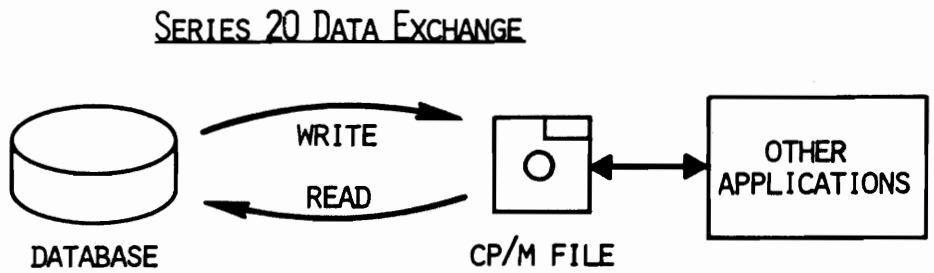


Figure 8-1. READ and WRITE Commands

The **WRITE** command **extracts** data from a database, and places it in a CP/M file. Once the data is in the file, applications such as Series 100/Word, Wordstar/100, Series 100/Graphics, MicroPlan/100, and your own programs can process it. Different HP Series 100 applications will require that the data file be created in slightly different formats, An option (which must be in upper-case) in the **WRITE** command controls the format that is used to write the file:

WRITE database output-filename [option]

The **READ** command takes data from a CP/M file, and enters it into a database. The data in the file may have come from a Series 100/Word document (or Wordstar non-document), a Visicalc/100 spread sheet, a Link/125 file transfer, your own programs, etc. Once entered into the database with the **READ** command, the data can be manipulated using all the other Condor commands.

the syntax of the READ command is:

READ database input-filename

Specific instructions for using databases with other HP Series 100 applications are given below. For complete details on the **READ** and **WRITE** commands and their file formatting options, see Appendix A and Appendix C.

SERIES 100/WORD AND WORDSTAR/100 MAILING LISTS

You can extract database data to create a Series 100/Word mailing list through the **[M]** (for mailing list) option of the **WRITE** command. The data items in the database will become, item by item, the variable parameters (~1, ~2, etc.) that Series 100/Word MailMerge feature will merge into the form letter. Each record of the database will create one customized letter. You may need to use the **PROJECT** command to create a database that contains exactly the right data items, in exactly the right order, before you can **WRITE** the database for use by Series 100/Word. A Wordstar/100 mailing list may be created by using the **[B]** option in the WRITE command.

Example:

A mailing is generated to selected customers, customized with (1) the customer's name, (2) address, (3) city, (4) state and (5) zip code. The CARDFILE database provides the mailing list:

SELECT CARDFILE WHERE ZIPCODE GT 70000
(select records for mailing)

SORT RESULT BY ZIPCODE
(sort into order for bulk mailing)

PROJECT RESULT BY NAME, STREET, CITY, STATE, ZIPCODE
(select the data items needed for mailing list)

WRITE RESULT NEWLIST [M]
(the mailing list file is named NEWLIST; it can be used directly by Series 100/Word Mailmerge. To create the mailing list for mergine with Wordstar/100, the command is:

WRITE RESULT NEWLIST [B]

SERIES 100/WORD AND WORDSTAR/100 DOCUMENTS

Data from Condor databases can be incorporated in tabular form into memos and other documents prepared using Series 100/Word. The **[G]** option of the **WRITE** command is used to present database information in row/column format, exactly as it would appear in a columnar report. Data from each record appears in each row, data items from the record are displayed in columns, from left to right, and a double space separates columns from one another.

After the **WRITE** command has created its CP/M file, Series 100/Word can access the file through its **READ (R)** or **READ FOR INSERTION (RI)** commands. Note that no column headings are written to the output file by the WRITE command; if required, they may be edited into the document with Series 100/Word. Reading the information into a Wordstar/100 document works in the same manner, using Wordstar/100 **READ A FILE** softkey.

Example:

Summary data from the ORDERS database is included in a Series 100/Word or Wordstar/100 memo:

SORT ORDERS BY REGION
TABULATE ORDERS BY REGION AND COMPUTE TOTAL AMOUNT AVERAGE AMOUNT [S]
(Orders are tabulated by region, with the results stored in the **RESULT** database. Each record in the **RESULT** database contains a region name, a total of orders, and an average of orders.)

WRITE RESULT REPORT [G]
(Transfer data to a file named **REPORT**)

Now, leave Condor, enter Series 100/Word or Wordstar/100, and compose the memo. When the time comes to insert the table, use the **READ FILE** softkey in the word processor.

When prompted, give the filename: **REPORT**

The table of data from the database will appear in the document.

If you want a report with Condor report headings, subtotals, etc., use the **SET PRINTER filename** command described earlier in this chapter and read it into your word processor with the **READ FILE** command.

PLOTTING DATABASE DATA

Using Series 100/Graphics, data from a Condor 20 database can be plotted in a wide variety of pie, bar and linear charts. The **[G]** option of the WRITE command is used to organize database data into the row/column format required by Series 100/Graphics. For pie charts, the database to be plotted must contain only two data items per record -- the first gives the label for a piece of the pie, and the second gives the data value for that piece. For bar charts, the database must contain a label data item and up to five value items per record. Linear charts draw only their data from the database record; labels are added through the Series 100/Graphics menu.

Example:

A pie chart of orders by region is required, with data drawn from the ORDERS database:

SORT ORDERS BY REGION
TABULATE ORDERS BY REGION AND COMPUTE TOTAL AMOUNT [S]
(generate order statistics by region, and store them in the **RESULT** database)

WRITE RESULT DATAFILE.DAT [G]
(put statistics in the file **DATAFILE.DAT**)

Now, exit Condor and enter Series 100/Graphics. Select the **PIE CHARTS** softkey, and the pie charts menu is displayed.

Type **DATAFILE** in the field labelled **File:**

Press the **GET DATA** softkey

The data from the database will appear in the pie chart menu.

Example:

Test results have been stored in a TESTDATA database, and are to be used to generate a Series 100/Graphics line chart:

PROJECT TESTDATA BY MONTH, TEST1, TEST2, TEST3
(Select only month and data points as data items. Month must be designated by 1, 2, ..., 12, ...)

SORT RESULT BY MONTH
(put result in time sequence)

WRITE RESULT DATAFILE.DAT [G]
(place test results in a file)

Now exit Condor and enter Series 100/Graphics.
Select the **LINEAR CHART** softkey.

Type **DATAFILE** in the field labelled **File:**

Press the **GET DATA** softkey

The test data from the database will appear on the screen as data for the linear plot.

Bar Chart data (also row/column in organization) can be generated in the same way.

DATABASES AND USER APPLICATION PROGRAMS

Data from Condor 20 databases can be processed by user-written HP Series 100 programs, whether written in Basic, Cobol, Pascal, Fortran or other languages. Basic programs use the **[B]** option of the WRITE command to obtain database data in a usable format. Other languages typically prefer to process data in an "ASCII card image" format, which is selected with the **[R]** option.

Example:

Place all the data items from the CARDFILE database into a file for processing by a Basic program:

WRITE CARDFILE NEWFILE [B]

When data is stored using the **[B]** option, a Basic statement such as:

```
INPUT #1, A$, B$, C$, D$, E$, F$, G, H$, I$, J$
```

will read successive data items into the string and numeric variables shown. Series 20 does this by separating alphabetic data with quotes and commas and numeric data with commas (e.g. "Joe Smith","President","ACME Company","123 4th Street","South Ridgeport","Washington",96045,.....etc.). Note, option letters need to be capitalized (e.g. **[B]** not **[b]**).

Example:

Place all the data items from the CARDFILE database into a file for processing by a COBOL program:

WRITE CARDFILE NEWFILE [R]

Each record from the database becomes a fixed-length record in the resulting file. Data items occupy exactly as many positions in the record as their data entry fields on the database form, and appear consecutively in the record with no intervening spaces. The COBOL program would define an input "record" structure to read the data using Picture XXX... to read alphabetic/alphanumeric data and Picture 9999... to read numeric data. Similar record structures would be used by other languages.

EXCHANGING DATA WITH "HOST" COMPUTERS

Data from Condor 20 databases can be sent to a wide range of "host" computer systems using Link/125 and the **[R]** option of the **WRITE** command. In addition, data from host computer files can be transferred to the HP 125 and read into Condor 20 databases for storage and processing. Link/125 also provides easy access to the Image/3000 database management system inquiry facility on the HP 3000 system, allowing data from Image/3000 databases to be entered into Condor 20 for processing.

To transfer data to a host computer, use the **[R]** option of the **WRITE** command. The data from the database will be stored in fixed-length records in the resulting file, one database record per file record. Data items appear consecutively in the record, in fixed-length fields like those in the database form, with no intervening spaces. Most programming languages on host computer systems find this data format the easiest to process.

Example:

Transfer detail data from the ORDERS database to the corporate HP 3000 system for processing:

WRITE ORDERS DATAFILE [R]
(the data is written to a file named DATAFILE)

Exit Condor, and enter Link/125.

Press the **TRANSFER TO HOST** softkey.

Enter **DATAFILE** as the source filename, and the destination filename of your choice.

Press the **BEGIN TRANSFER** softkey.

The data will be transferred to the HP 3000 system.

File transfer from a host system is accomplished the same way, using the READ command:

Example:

Transfer ORDERS data from a host HP 3000 system into the Condor ORDERS database:

Using Link/125:

Press the **TRANSFER FROM HOST** softkey.

Enter the source file to be transferred, and enter **DATAFILE** as the name of the destination file.

Press the **BEGIN TRANSFER** softkey

When transfer is complete, exit Link/125 and enter Condor.

Type **READ ORDERS DATAFILE**

The data from the HP 3000 must be in the option [B] (quotes surrounding items) format described in a previous example, with all data items appearing in the proper sequence and with large enough lengths defined defined in the ORDERS database.

To read data from an Image/3000 database into a Series 20 database, you must first write a Query/3000 procedure that generates a row/column report of information from the HP 3000. The format of the report must generate fixed-length records, as described in the two preceding examples. When the procedure has been written and stored on the HP 3000, you are ready to transfer the data. Follow the same steps as in the previous example, but use the **QUERY/3000** softkey in Link/125 instead of the **TRANSFER FROM HOST** softkey.

The information on **READ/WRITE** file formats in Appendix C is especially useful when exchanging information with an applications system on a host computer. Depending upon the host system being used, one or another of the possible formats may make the host's processing burden easier, or eliminate the need to rewrite an application program.

USING DATA FROM SERIES 100/WORD AND WORDSTAR/100 DOCUMENTS

Series 100/Word and Wordstar/100 are frequently used to prepare reports, and the row/column data in a report may become a source of data for a Condor 20 database. However, you will have to edit the report (using the word processor) to make its format agree exactly with the data item definitions in the database that is to receive the data. The example shows how to do this. To serve as a source of data for a database, a report must have the following two characteristics:

1. The rows of the report must represent the records of the database to be created; one row per record.

2. The columns of the report must represent the data items of each record, one data item per column, in the correct consecutive order, left to right.

The example shows how to transfer data into a database from a report that meets the criteria. **(Note: If creating the report in Wordstar/100, it must be created as a NON-DOCUMENT file.)**

Example:

A printed report shows the following information:

REGION	TARGET	ACTUAL	DIFFERENCE
WEST	30000	35000	5000
SOUTH	240000	235000	-5000
EAST	150000	152000	2000
	--- etc ---		

The data is to be entered into a database with the following data item definitions:

```
REGION:  AN, 5 characters
TGT-AMT: N,  6 character data entry field
ACT-AMT: N,  7 character data entry field
DIFF:    N,  6 character data entry field
```

The report meets the conditions necessary to use it as a source of data for this database, so you can proceed:

1: First, use Series 100/Word or Wordstar/100 to strip away all information except the lines of the report that actually contain data. Specifically, delete report titles, column headings, etc.

2: Eliminate any total lines, or any space lines in the report. After this step, there should be exactly as many lines in the report as there will be records in the database when you are finished.

3: Each column of the report must now be edited so that it occupies exactly as many characters as the corresponding data entry field on the database form. All intervening spaces between columns must be eliminated. For the example report, the REGIONS column must occupy exactly 6 spaces, followed immediately by the TARGET column at 7 spaces, etc. Note: data will probably be "smashed together" horizontally by this step. Although it appears strange, this format is required to read the data into Series 20. The example report will look like this:

```
WEST  30000 35000  5000
SOUTH240000235000 -5000
EAST 150000152000  2000
```

4, Save the (edited) report into an ASCII file, using the **/1** option:

Press the **SAVE FILE** softkey.

Enter **REPORT/1** for the filename.

5. Exit Word/125 and enter Series 20. Type the command:

READ ORDERS REPORT

to load the data.

READING VISICALC DATA INTO A DATABASE

Data from a Visicalc spread sheet can also be entered into a Series 20 database through the **READ** command. The data must be formatted in Visicalc so that each row in the Visicalc spread sheet corresponds to one record in the database. Each data item must occupy one column. When you have organized the data in this way, use Visicalc's "Print File" (/PF) command to print the spread sheet into a row/column ASCII file. Print **only** the rows with actual data that are to become records of the database; do not include title rows, blank rows, etc.

The width of each Visicalc column must correspond **exactly** to the size of the corresponding data item's data entry field size in order for the Visicalc print file to be entered directly into Condor 20. Since all columns in a given Visicalc report must be the same width, this means that each data item in the database must be the same size. Typically, this is too restrictive. Therefore, Series 100/Word or Wordstar/100 may be used to further edit the Visicalc spread sheet report to make it conform to the desired database. See the section on using data from Series 100/Word and Wordstar/100 documents above for detailed editing and formatting guidance.

CHAPTER SUMMARY

* Condor 20 can exchange information with other HP Series 100 applications, greatly increasing the power and flexibility of database applications.

* The **READ** command takes information from an ASCII file and enters it into a database. It is used to process data generated by other applications with Condor 20.

* The **WRITE** command takes information from a Condor 20 database, and writes it in an ASCII file. It is used to pass Condor 20 data to other HP Series 100 applications.

* Different file formats of the output ASCII file are available through options of the **WRITE** command.

CHAPTER 9

RESTRUCTURING AND REORGANIZING DATABASES

In the preceding chapters, a database has been treated as a single entity, which could be sorted, printed, copied, displayed, updated, etc. This is a very powerful feature of Series 20, since you can use databases, even in very complex ways, without having to understand the underlying files and data structures.

However, Series 20 becomes an even **more** powerful tool if you have an understanding of how a Series 20 database is "put together". This knowledge is also required if a database is to be redefined or restructured after it contains data. This chapter explains the underlying structure of a Series 20 database, and how it can be used to effectively restructure and manipulate the database in advanced ways.

DATABASE STRUCTURE

A Series 20 database is a complex structure, made up of data items, records, a database form, etc. Series 20 uses CP/M files to store the component parts of a database. Each Series 20 database has three associated files:

FILE	PURPOSE
Form File	Stores the database form. This file is edited by the **FORMAT** and **DEFINE** commands. The form file is named **xxxxxxxx.FRM**, where **xxxxxxxx** is the database name.
Definition File	Stores the data item definitions, including the data type, length, maximum, minimum, and default values for each data item. The definition file is named **xxxxxxxx.DEF**, where **xxxxxxxx** is the database name.

Data File	Stores the actual data of the database, along with the current record count. The data file is named **xxxxxxxx.DAT**, where **xxxxxxxx** is the database name.

Working together, these three component files store the data of the database, define the data so you may access it by name, and determine how the database data is displayed for your review.

THE RESULT DATABASE

The **RESULT** database was first discussed in Chapter 6, which described its role in relational database operations. Like all other databases, the **RESULT** database has three component files, but with the following unique names:

Form File:	**R$.FRM**
Definition File:	**R$.DEF**
Data File:	**R$.DAT**

The Series 20 commands that create a **RESULT** database use these filenames to generate the appropriate form, data item definitions, and data for the **RESULT** database.

OTHER DATABASE FILES

In addition to the component files associated with each database, Series 20 also uses CP/M files for other purposes. In each case, the filename extension shows at a glance the purpose of the file. The other files used by Series 20 are summarized in the table below:

FILENAME	PURPOSE
xxxxxxxx.HLP	Stores a HELP screen. (See Chapter 10)
xxxxxxxx.CMD	Stores a Command Procedure. (See Chapter 10)
xxxxxxxx.DBM	Series 20 Command File -- stores instructions needed to execute Series 20 commands.
DATA.DIC	Stores the Data Dictionary.
R$.DBM	Stores the work file used by the **RUN** command in processing command procedures. (See Chapter 10)

R$.ASC	Stores the ASCII file used by the READ and WRITE commands when no filename is given. (See Chapter 8)

MANIPULATING DATABASE FILES

Most of the "utility" commands available through Series 20 can be used with individual database files, as well as entire databases. To manipulate individual database files, simply specify the full filename (including the filename extension) in the command.

Examples:

COPY NEWDBASE.DAT = OLDDBASE.DAT
(copies data files only)

DESTROY NEWDBASE.DAT
(erases the data file, leaving the definition file and form files intact)

These commands may be useful for database backup, and they are required for some of the advanced database reorganization and restructuring described later in the chapter. Appendix A lists, for each Series 20 command, whether it may be used with files as well as entire databases.

REVISING A SCREEN FORM

When you create a new database, the **DEFINE** command steps you through creating a new database form and entering a new set of data item definitions. If you need to revise the form later, the **FORMAT** command allows you to edit the form file directly, without effecting the data item definitions or data in the database. You must be careful to edit the form only cosmetically. The lengths of the data entry fields, and the order in which the data items appear on the screen must not be changed after data has been entered into the database! If you change any of these characteristics of the form, the data stored in the database cannot be displayed correctly.

The **FORMAT** command may also be used to create a new form file, as an alternative to the **DEFINE** command. Additionally, it is used to create and edit **HELP** screens. The **FORMAT** command names the file to be created or edited:

FORMAT filename.FRM
or
FORMAT filename.HLP

The **FORMAT** command displays the form to be edited on the screen (or displays a blank screen, if the form is being created for the first time). The HP Series 100 editing keys may be used to move the cursor about the screen, and edit the form at will:

Arrow keys -- move the cursor about the screen

DEL CHAR -- deletes the character at the current cursor position

DEL LINE -- deletes 80 characters from the form, beginning at the cursor position, and wrapping around to the next line.

INS LINE -- inserts 80 blanks into the form, beginning at the cursor position, and wrapping around to the next line.

The **INS CHAR** key toggles the **FORMAT** command back and forth between its two modes of operation, **replace** mode (the normal mode) and **insert** mode.

In **replace** mode, the characters you type **replace** (over type) the characters already on the screen. Data item names, underscores and comments may all be entered on the screen in this manner. If you need to add a character to the screen and not replace other characters (e.g. insert a word between two words), the **insert** mode should be used.

Pressing the **INS CHAR** key enters **insert** mode. In this mode, each character you type is **inserted** into position on the screen, at the current cursor position. All subsequent characters in the line are shifted one character position to the right, to make room for the inserted character. The character in column 80 of the row is shifted off the screen and lost. Pressing **INS CHAR** again returns to replace mode.

Pressing the **Grid** softkey will paint a grid on your screen, making it simpler to line up items. The grid will not appear in your saved form. The **Refresh Screen** softkey will erase the grid from the screen.

When you have finished editing a form, press the **End** softkey. The form file will be replaced on the disc. To end editing without storing the revisions, press the **Abort** softkey. The **Refresh Screen** softkey will "repaint" the screen.

PRINTING DATA ITEM DEFINITIONS

You can print out the data item definitions of a database (from

the definition file) with the **DEFINE** command. Simply enter:

DEFINE database-name

Since the database already exists, you are asked whether you want to print its description, redefine it, or end. Typing **D** (for describe) lists the data item definitions on the screen or the printer, at your option.

Example:

To list the CARDFILE data item definitions:

Type **DEFINE CARDFILE <RETURN>**

Condor 20 responds:
This Data Base exists already.
Choose Option: Describe (D), Re-define (R), End <C/R>

Type **D <RETURN>**
(select the "Describe" option)

Condor 20 responds:
Send output to printer (Y/N)?

Type **Y <RETURN>**

The data item definitions, record size, and record count are printed on the printer.

REVISING DATA ITEM DEFINITIONS

If a database is empty (i.e. it contains no records), the **DEFINE** command can be used to revise its data item definitions. Follow the sequence:

Type **DEFINE database <RETURN>**

Condor 20 will respond:
This Data Base exists already
Choose Option: Describe (D), Re-define (R), End <C/R>?

Type **R <RETURN>**
(to redefine the data items)

Each data item in the database will be displayed, with its data item type, field size, minimum value, maximum value, and default value. To leave the data item definition unchanged, press **<RETURN>**, and the next data item will be displayed.

To change the definition of the data item, use the left and right arrow keys to move the cursor and edit the definition. Simply type over the current definition, and press **<RETURN>** when you have edited the definition to your satisfaction. If you change the type of a data item, it is a good idea to let the system recalculate the field size, minimum and maximum by "wiping out" these definitions with the space bar before pressing **<RETURN>**.

When redefining data items, you can jump to a specific data item by typing a "#" symbol. The system will prompt with a ">", to which you should respond with the data item name to be changed. To end the revision without **<RETURN>**ing all the way through the remaining definitions, type a "#", and when prompted, type another # followed by **<RETURN>**.

```
**********************************************************
*                  ***** CAUTION *****                   *
*                                                        *
*  If you lengthen an item's definition field length,    *
*  be sure to lengthen its field length in the .FRM      *
*  by using the FORMAT command.  Otherwise, when dis-    *
*  playing or updating your database, you will receive   *
*  the error message:  Insufficient space for data item  *
*  value.                                                *
**********************************************************
```

CHANGING MAXIMUM, MINIMUM AND DEFAULT VALUES

You cannot directly change the data item definitions for a database that contains data. However, a special version of the **UPDATE** command allows you to change the maximum, minimum and default values for data items -- even for a database containing data. This special **UPDATE** command is used to edit the definition file of the database directly:

UPDATE filename.DEF ST FIELD IS nn, ...

The filename specified in the command should be the definition file whose maximum, minimum and default values are to be changed. Specify the field numbers for the data items to be changed in the command. **Example:**

To change the default value for FIRM (the third data item) in the CARDFILE database, enter the command:

UPDATE CARDFILE.DEF ST FIELD IS 3

The form shown in Figure 9-1 is displayed, together with the data item type, length, maximum, minimum, and default values for the selected data item. The form operates just like any **UPDATE** command -- you may revise the data, or leave it unchanged. Other fields may be selected by entering further **"FIELD IS nn"** selection criteria.

```
DATA DEFINITION SET

FIELD No.  3          TYPE of Field AN          SIZE of field  30 (Bytes)
                      (Table I)                 (Table II)

                          Edit Criteria

MIN          0  MAX          30  DEFAULT                  rsvd     0
```

I. FIELD TYPES		Modifiers	II NUMERIC FIELD SIZES (Bytes)	Min	Max
Alphabetic	A	Required Entry R	1	-128	+127
Numeric	N		2	-32768	+32767
Alphanumeric	AN		3	-8,388,608	+8,388,607
Dollar	$		4	-2,147,483,648	+2,147,483,647
Date (Julian)	J				

Figure 9-1. Updating Max, Min and Default Values

This version of the **UPDATE** command allows you to edit the definition file as if it were itself a database. For a complete description of the softkeys available in the **UPDATE** command, see Chapter 4.

THE DATA DICTIONARY

How does Series 20 keep track of all the component files of a database, given the dozens of databases you will define and use? It uses a special file called the **Data Dictionary.** Series 20 automatically creates a Data Dictionary on each disc the first time you define a database on the disc. From that point on, each time you add a new database or destroy a database on the disc, the Data Dictionary is updated to reflect the change.

The Data Dictionary is organized so that it appears as a database to you. Each "record" of the Data Dictionary shows the structure of one database -- its database name (title), and the names of its component form file, definition file, and data file. Figure 9-2 shows a Data Dictionary listing for some of the example databases in this manual.

Title	Form	Definition	Data
CUSTOMER	CUSTOMER	CUSTOMER	CUSTOMER
PRODUCTS	PRODUCTS	PRODUCTS	PRODUCTS
REGIONS	REGIONS	REGIONS	REGIONS
CARDFILE	CARDFILE	CARDFILE	CARDFILE
ORDERS	ORDERS	ORDERS	ORDERS
RESULT	R$	R$	R$

Figure 9-2. Sample Data Dictionary

When you use a database name in a command, Series 20 uses the Data Dictionary to locate the files needed to process your request. Each record in the Data Dictionary "defines" one database, by specifying its form file, its definition file, and its data file.

Because the Data Dictionary appears as a database, you can use Series 20 commands to process it. For example, the commands:

ENTER DATA.DIC and
UPDATE DATA.DIC

can be used to add new database definitions to the Data Dictionary, and to modify database definitions. Similarly, the commands:

LIST DATA.DIC BY TITLE, FORM, DEFINITION, DATA
and
PRINT DATA.DIC

will print the Data Dictionary contents. Figure 9-3 shows the form that is used to enter or update information in the Data Dictionary directly. More examples of manipulating the Data Dictionary are given later in this chapter.

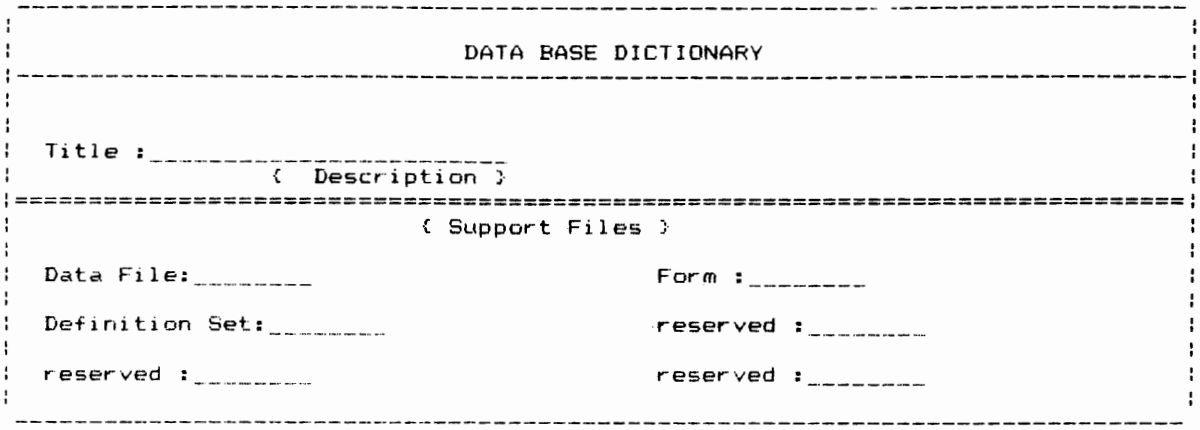


Figure 9-3. Data Dictionary Form

ALTERNATE DATABASE DESCRIPTIONS

Thus far, each database you have seen has had its own unique form file, definition file and data file, separate from all other databases. With the Data Dictionary, however, a single component file can be shared among several different databases. For example, by sharing a single form file, definition file, or data file among multiple databases, you can:

-- look at several different sets of data through the same form
-- look at one set of data through several different forms
-- use two different sets of data item names to describe the same set of data

The Data Dictionary thus allows a database to have several different "alternate" definitions, to meet different needs. Many database applications can benefit from this capability, using it to describe a single database in two or more different ways. For example, you may want to use a form with elaborate data entry instructions to enter orders into a database, but use a much simpler form to display the orders for inquiry. Or, you may need to use two different sets of data item names to relate a database to other databases, if data item naming had not been planned carefully in advance. The two examples that follow illustrate how you may create alternate database descriptions through the data dictionary.

Example:

A separate ORDERS database is required for each of two subsidiaries. Both will contain the same data items, and both will use the same form for data entry. We can use the **DEFINE** command to create two databases that share common forms and data item definitions, but each contain their own data:

Define the first database normally:

DEFINE ORDERS <RETURN>

Type in the form, and the data item definitions.

For the second database, answer the questions in the **DEFINE** command dialogue as follows:

Type **DEFINE ORDERS2 <RETURN>**

Series 20 responds:
Do you wish to create a new form (Y/N)?

Type **N <RETURN>**
(you want to use the form from the ORDERS database)

Series 20 responds:
ORDERS2 FRM does not exist -
Enter different name or End Program <C/R>:

Type **ORDERS <RETURN>**
(you tell Series 20 that the ORDERS2 will share the ORDERS.FRM form file)

Series 20 responds:
Do you wish to create a new definition file (Y/N)?

Type **N <RETURN>**
(you want to use the data item definitions from the ORDERS database)

Series 20 responds:
Enter Filename or End Program <C/R>:

Type **ORDERS**
(you tell Series 20 that the ORDERS2 database will share the ORDERS.DEF definition file)

Series 20 responds:
Do you want a printed copy of the data definitions (Y/N)?

Type **N <RETURN>**

Condor 20 will display a summary of the database attributes, and a new data dictionary record is created for the database ORDERS2. You can see the new entry:

Type **LIST DATA.DIC BY TITLE, FORM, DEFINITION, DATA**
(ORDERS2 is listed, composed of the ORDERS.FRM, ORDERS.DEF, and ORDERS2.DAT files)

As this example shows, a new database that shares a form file or data definition file with other databases can be created by naming those files in the **DEFINE** command dialogue. The form and definition files must already exist when the **DEFINE** command is entered. A new database created in this way will always have its own, unique data file.

If two databases are to share the same data file, the Data Dictionary records must be edited directly to create the alternative database definition. The following example illustrates how this can be done.

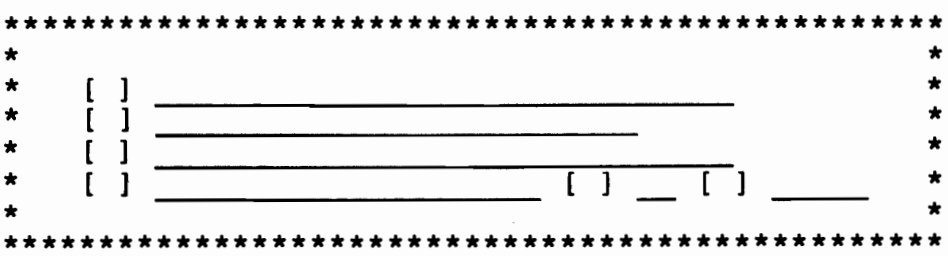
Example:

A CUSTOMER database has been created with the following form:

```
**************************************************
*  [NAME] _____________________________           *
*  [COMPANY] _____________________________        *
*  [ADDRESS] _____________________________        *
*  [CITY] _____________________                   *
*  [STATE] __                                     *
*  [ZIP] _____                                    *
**************************************************
```

We wish to print mailing labels for the customers. One way to do this is to define a new database (named MAILLABL) that has a different form, but shares the same data and data item definitions as the CUSTOMER database. The MAILLABL form will position the data items correctly for printing on the mailing label. To define the alternative MAILLABL database description, do the following:

Use the **FORMAT** command to create a MAILLABL.FRM file with the proper form for the mailing label. The form will have underscores (for data fields), but no data item names, as shown below:



When the system asks if you wish to define a database using this form:

Type **N**

With the form defined, you can create the Data Dictionary entry for the new database:

Type **ENTER DATA.DIC**

The Data Dictionary form from Figure 9-2 will appear. Enter **MAILLABL** for the database title, and **MAILLABL** for the database form file name. Enter **CUSTOMER** for the definition file name, and **CUSTOMER** for the data file name.

Now, you can print the mailing labels:

```
SORT CUSTOMER BY ZIP
PRINT MAILLABL
```

The preceding example also illustrates how you could define an alternative database description with different data item names than the original description. This is a common requirement when two databases are first related using Series 20's relational capabilities. Unless great care was taken when the databases were first defined, you may find that the data items you wish to use as matching data items (for the JOIN, COMPARE or POST commands) have different names in the two databases. An alternative database definition for one of the databases solves the problem easily, without reorganizing one of the databases unnecessarily.

You must use extreme caution when creating alternative database descriptions using the **ENTER** and **UPDATE** commands directly on the Data Dictionary. It is possible to create alternative descriptions where the database form, descriptions, and data make no sense together, resulting in nonsense output when you try to print or use the database. Be certain that the form, definition and data files that you combine describe a consistent, logical structure for the new database.

REORGANIZING A DATABASE

Great care should be taken when initially defining a database, to insure that it contains the needed data items, that they are named correctly for relating them to other databases, and that they have the right data type for the processing required. Nonetheless, you may occasionally need to restructure a database that contains data. Using the **REORG** command, you can delete data items from a database, add new data items, or change the order in which the data items appear.

The idea behind a **REORG** is as follows:

1. The form file is to be altered so as to add, delete, or reorder data items.

2. The definition file will be changed to reflect any additional or deleted data items.

3. The data that currently exists is to be transferred into the newly reorganized database.

To reorganize a database, the **REORG** command is followed by the database to be altered:

REORG database

REORG will automatically bring up the form file for modification. At this time, new items or titles may be added, reordered, and/or deleted. All of the current data items are remembered by Condor, so if an item name is changed, it is assumed that the item has been deleted and a new one added.

Care should be taken when altering underscores. If there are less underscores than the definition of a item calls for, an error will be issued when later trying to display data: **Insufficient space for data item value.**

After the form file has been altered, press the **End** softkey. **REORG** begins modification of the database definition file. If an item is added, Condor prompts you for the specifications of that item. Enter definition values in the same manner as when a database is defined. If an item is deleted, Condor will denote that the item is to be deleted, then ask for a confirmation. **If you answer Y, all of the data for that item will be deleted from the database.** If you decide that you don't want to delete that item, answer **N**, and REORG will be aborted. You may then use the REORG command again and add the item name back onto the form.

Example:

We wish to delete the HOME-PHONE data item from the CARDFILE database, and add a new data item for EXTENSION-NR. This is accomplished as follows:

REORG CARDFILE

The form file is displayed on the screen, just as is seen using the FORMAT command. Using the cirsor and screen control keys, delete the HOME-PHONE item and replace it with the EXTENSION-NBR item. When finished, press the **End** softkey. The system will respond by asking you to confirm that the HOME-PHONE item is to be deleted.

Type **Y**

Now, Condor prompts you for the specification of only the new item that has been added, EXTENSION-NBR. Enter specifications the same way as is done using the DEFINE command. The system will ask **Definitions OK (Y/N)?**

Type **Y**

After listing the attribute summary of the new definition to the screen, Condor will ask if it is okay to replace the current database with the **REORG** result.

Type **Y**

REORGANIZATION USING READ/WRITE

When changing an item's field definition length or a field name, the reorganization **must** be accomplished by copying the database contents out of the database (into an ASCII file), and then copying the contents back into another (restructured) database. The **READ/WRITE** command pair is used to copy the database contents out of, and back into a database. Extreme caution must be exercised when using this procedure. Series 20 cannot check the **WRITE/READ** operation to be sure the "new" database definition is compatible with the "old" definition; and if the two definitions are not compatible, data will appear "garbled" when displayed. Be careful.

The procedure to follow to perform this type of database restructuring is:

1. Be sure you have a backup of the database to be restructured. If not, make a duplicate, using the Condor COPY command.

2. Write out the database contents to a CP/M file using the **WRITE** command.

3. Use the **EMPTY** command to delete all the data from the database; now use the **DEFINE** command to redefine the data item definitions in the empty database (the **DEFINE** command will not work if the database is not empty).

4. Read the database contents back into the redefined database using the **READ** command

5. Destroy the temporary files used by the **WRITE** and **READ** commands

Example:

We wish to change the name of the FIRM data item in the CARDFILE database to COMPANY, and make the DATE data item an 8-character alphanumeric field. The following sequence of commands will restructure the database:

```
COPY TEMP = CARDFILE
     (we will work with a duplicate copy, just in case!)

WRITE TEMP TEMPFILE [B]
     (write contents out to a file)

EMPTY TEMP
DEFINE TEMP
     (revise the item data definitions, changing
      NAME to COMPANY)

FORMAT TEMP.FRM
     (revise the form file, changing NAME
      to COMPANY)

READ TEMP TEMPFILE
LIST TEMP
     (read data back in, and check it!)

COPY CARDFILE = TEMP
DESTROY TEMP
DESTROY TEMPFILE
     (copy the RESULT database, if good, and destroy
      the temporary database and file)
```

CHAPTER SUMMARY

* A database is made up of three files:
  - a **FORM** file
  - a **DEFINITION** file
  - a **DATA** file

* Many Series 20 commands can be used to process individual database files as well as entire databases.

* The **FORMAT** command is used to edit or revise database forms or **HELP** screens.

* The **DEFINE** command can be used to list and revise data item definitions.

* The **Data Dictionary** defines the databases on a disc; it keeps track of their form, data, and definition files.

* The **ENTER, UPDATE, DELETE, LIST, PRINT** and other commands can be used with the Data Dictionary (DATA.DIC) to allow databases to share common files.

* The **DEFINE** command can be used to reorganize an empty database.

* Reorganization of a database containing data can be accomplished with the **REORG** command.

* The **READ/WRITE** commands can be used for other kinds of reorganization when a database contains data.

CHAPTER 10

DATABASE COMMAND PROCEDURES

One of the most powerful features of Series 20 is the ability to manipulate information on demand. Using the commands described in the preceding chapters, databases can be queried, combined, interrelated and manipulated to produce information as it is needed.

Often, however, the same procedure for processing a database is repeated over and over again. For example, the month-end reporting procedure that generates reports from a ORDERS database may consist of twenty or thirty commands, which must be entered every month. Or the daily posting procedure for the database may require a long sequence of commands that is repeated every day.

Series 20 allows very efficient processing of repetitive database procedures through two powerful capabilities -- **Command Procedures** and **Help Screens.** Using these facilities, repetitive database operations can be defined once, and then performed over and over again with the press of a few keys.

Command Procedures and Help Screens can also be used to make Series 20 the foundation of an entire transaction processing application, complete with data entry and inquiry functions, reports, and daily, weekly and monthly procedures. Application users (who are typically clerical personnel unfamiliar with the structure of the database) can select the procedures they need through simple menus, and access database data through familiar business forms.

COMMAND PROCEDURES

A Command Procedure is a sequence of one or more Series 20 commands stored in a "command procedure" file on a disc. Once the command procedure has been defined, it may be retrieved and processed at a later time, using the **RUN** command. In this way, long and complex command sequences can be defined once, and then executed whenever they are needed, with a single command.

Command Procedures are processed by the Series 20 **RUN** command. The **RUN** command is similar to the CP/M SUBMIT command. It allows you to tailor the execution of a command procedure through symbolic parameters, and a powerful set of command procedure directives. Among its capabilities are:

-- Conditional processing of a group of commands, through an ***IF...*ENDIF** structure

-- Assignment of parameter values through the **RUN** command, through user prompts, or through assignment directives within the command procedure

-- Display of informational messages for the user

-- Dynamic suppression/enabling of screen echo of command processing

-- Orderly end, abort and restart of command procedures

Command procedures are started by typing the **RUN** command:

RUN procedure param1 param2 ... param9

When the **RUN** command is entered, the procedure must have been previously defined. The **RUN** command can process up to 9 parameters (variables). These parameters may optionally be assigned values by listing the values in the command line, following the procedure name.

Example:

```
RUN  ENDMONTH  FEBRUARY  3/3/82
   (run end-of-month procedure; parameter#1 = "FEBRUARY",
    parameter #2 = "3/3/82")
```

Command procedure files are stored with a file extension of ".CMD". For example, the procedure "MONTHEND" is stored in the command procedure file "MONTHEND.CMD". The extension is not specified in the RUN command.

COMMAND PROCEDURE PROCESSING

It is very important that the designer of a Series 20 command procedure understand **exactly** how the **RUN** command will go about processing that procedure. The **RUN** command, and its associated directives, actually form a small macro processor capability within Series 20. Command procedure processing by the RUN command takes place in two steps.

First, the command procedure is scanned, and a work file of commands to be executed is built. (The work file is named R$.DBM, and is created on the current disc drive.) Several functions are performed during the scan:

1) Parameters ($1, $2, ... $9) are replaced with the value given in the **RUN** command line, assigned with the ***LET** or entered through the ***GET** directive. **$TODAY** is replaced with Series 20's current date.

2) When a ***IF** directive is encountered, the specified condition is tested to determine if the command lines between the ***IF** and the ***ENDIF** will be written to the work file. Command directives appearing between the ***IF** and ***ENDIF** directives are processed only if the specified condition is satisfied.

3) When a ***MESSAGE** directive is encountered, the message is displayed on the screen.

4) When a ***GET** directive is encountered, a colon (:) prompt is displayed on the screen. Processing is suspended until the requested parameter values are entered. The value entered is used for all subsequent appearances of a parameter until a subsequent ***GET** directive is processed for the same parameter.

5) When a ***LET** directive is encountered, the parameter on the left side of the equals sign (=) is assigned the value appearing on the right side of the equals sign. If a parameter appears on the right side, its value is assigned. The ***LET** directive is particularly useful for setting parameter default values.

6) When a ***END** directive is encountered, scanning terminates. No further lines in the command procedure file are scanned. Commands that have been written to the work file prior to encountering the ***END** are now executed in sequence.

7) When a ***ABORT** directive is encountered, scanning terminates. No commands are written to the work file. No commands will be executed.

8) The ***ECHO** directive has the effect of writing a **SET ECHO** command as the first line of the work file. If several ***ECHO** directives are processed, the **ECHO** status during execution of the commands in the work file will be that of the last ***ECHO** processed during scanning.

9) Comments (lines beginning with a semicolon (;)) are not written to the work file.

10) Directives are not written to the work file. Therefore, no conditional testing occurs during execution. Condition testing applies only during the scanning operation.

After the scan is complete, the second step is to execute the command procedure. The commands in the work file are read one by one, and executed. When the last command in the work file has been executed, the command procedure is complete.

A command procedure may leave Series 20 control with a SYS command to CP/M level on disk A:. The next commands that follow need to be CP/M level commands. With this feature you may call other applications.

COMMENTS IN COMMAND PROCEDURES

Command procedures may include comment lines, which do not affect procedure processing. Comments improve command procedure readability, and may be used to make procedures self-documenting to aid later maintenance. Any command procedure line which begins with a semicolon (";") is treated as a comment by the **RUN** command.

Example:

```
; Command Procedure to print Month-End Reports
; These are comment lines
;
; First, print orders by region report
SORT ORDERS BY REGION
TITLE "REGIONAL ORDER REPORT"
TABULATE ORDERS BY REGION, AMOUNT, TYPE
  SUBTOTAL USING REGION
;
; Follow with orders by order type report
SORT ORDERS BY TYPE
TITLE "MONTHLY ORDERS BY TYPE"
PRINT ORDERS BY TYPE, ORDER-NO, AMOUNT
  AND COMPUTE TOTAL AMOUNT SUBTOTAL USING TYPE
*END
```

ECHOING COMMAND PROCEDURE EXECUTION

Normally, when Series 20 executes a command procedure, the individual commands and their results are echoed to the HP 125 screen during execution. This allows tracing of the command procedure execution as it occurs. If no echo is desired, include the directive:

***ECHO OFF**

in the command procedure. This has the effect of issuing a SET ECHO OFF command before execution of the command procedure begins. Echo is always turned back on again at the end of the command procedure execution, or if an error or abort condition occurs, or a **HELP** command is executed.

The opposite effect may be obtained by including the directive:

***ECHO ON**

in the command procedure. If multiple **ECHO** directives are processed during scanning of a command procedure, the last one processed before actual procedure execution commences is the directive that will be used.

MESSAGES IN COMMAND PROCEDURES

Messages may be displayed for the user during processing of a command procedure through the ***MESSAGE** directive:

***MESSAGE message-text**

The message is displayed on the HP 125 screen when the directive is encountered during initial scan of the command procedure. No resulting command is written into the work file for later execution. The ***MESSAGE** directive is especially useful in combination with the ***GET** directive, prompting the user with an easy-to-understand message for the parameters which must be supplied.

COMMAND PROCEDURE PARAMETERS

A command procedure may have up to nine parameters (variables), which can be used to make its use more flexible. Parameters may be used, for example, to pass a report date to a procedure, or to pass a yes/no response to a question, or to select which of several report options is to be printed.

The parameters are indicated within the procedure by the symbols **$1, $2, ... $9.** They are treated as variable-length character strings. During command procedure processing, as each line is scanned, the current value of any parameters used in that line is substituted for the parameters, in the exact position where they appear. The parameter-substituted command line is then copied to the work file for later execution. The parameters thus serve exactly the same role as macro parameters in the classic definition of a macro processor.

In addition to the nine user-defined parameters, Series 20 provides a special parameter named **$TODAY.** **$TODAY** always has the value of current date (as entered by the user through the **DATE** command). **$TODAY** may be freely used in command procedures, for comparison to other dates, etc.

******* Caution *******

In a command procedure, Series 20 treats any word beginning with a dollar sign ($) as a parameter, unless the dollar sign is immediately followed by another dollar sign, in which case it is treated as a literal character. A word such as "$AMOUNT" must therefore be specified as "$$AMOUNT" when referenced in a command procedure.

CONDITIONAL PROCESSING

Commands in a command procedure can be selectively included or excluded from execution through the ***IF** and ***ENDIF** directives. The syntax of these directives is:

```
*IF condition

...other commands & directives go here

*ENDIF
```

The condition may be any logical condition, with the same rules applying as those used for the condition in the **SELECT** command. ***IF...*ENDIF** directives can be nested within a command procedure, up to seven levels deep.

When an ***IF** directive is encountered during procedure scanning, the condition is tested. If true, scanning continues with the command immediately following the ***IF** directive, and the ***ENDIF** is ignored when encountered. If the condition is false, then the commands and directives that follow the ***IF** are skipped over, until the matching ***ENDIF** is found. Scanning then continues normally with the command following ***ENDIF.**

It is important to remember that conditional testing takes place at scanning time, and not at actual procedure execution.

ENDING OR ABORTING A PROCEDURE

The ***END** directive is used to halt scanning of a command procedure, and cause actual execution of the procedure (from the work file) to begin. Multiple ***END** directives may, of course, be used within a single command procedure -- the first ***END** encountered during scanning will halt the scan. An ***END** directive is not required at the physical end of the command procedure file; one is assumed.

If an abnormal condition is encountered during procedure scanning, the ***ABORT** directive may be used to halt scanning immediately and prevent execution from taking place. When the ***ABORT** directive is encountered during scanning, no commands are written to the work file, and none are executed. Processing of the **RUN** command immediately halts.

CHANGING PARAMETER VALUES

Command procedure parameters get their initial values from the **RUN** command. If parameter values are specified in the **RUN** command, they are assigned to the parameters before procedure scanning begins.

The values of parameters can also be altered during the scanning of the command procedure. The ***LET** directive serves as an "assignment statement" to assign a parameter a value within the procedure:

```
*LET parameter = value
```

Examples:

```
*LET $4 = $5
*LET $4 = "Boston"
```

Another way of altering parameter values is to request a value for the parameter from the user. This is done with the ***GET** directive:

```
*GET parameter
```

Example:

```
*GET $4
*GET $3, $5
```

When the ***GET** directive is encountered during procedure scanning, the user is prompted with a colon (":") on the HP 125 screen, requesting that he supply a value for the parameter. Multiple parameter values may be assigned with a single ***GET** directive by listing the parameters in sequence.

COMMAND PROCEDURE EXAMPLE

The sample command procedure that follows prints a report of sales within a specified range of dates, with optional totals and subtotals on a user-selected data item. It illustrates use of all the command procedure directives discussed in this chapter.

```
 1)   ;THIS PROCEDURE PRINTS A SALES REPORT
 2)   ;Syntax:  RUN SALEREPT start-date end-date
 3)   ;     $1 IS THE BEGINNING DATE OF THE REPORT PERIOD
 4)   ;     $2 IS THE ENDING DATE OF THE REPORT
 5)   *LET $4 = N
 6)   *MESSAGE The report will show sales from $1 to $2
 7)   *MESSAGE Are these the dates you want to use? (Y/N)
 8)   *GET $3
 9)   *IF $3 NE Y
10)       *ABORT
11)   *ENDIF
12)   SELECT SALEDATA ST DATE GE $1 AND DATE LE $2
13)   *MESSAGE Do you want Totals? (Y/N)
14)   *GET $3
15)   *IF $3 = Y
16)       *MESSAGE Do you want Subtotals? (Y/N)
17)       *GET $4
18)       *IF $4 = Y
19)            *MESSAGE Enter Subtotal Data Item
20)            *GET $5
21)            SORT RESULT BY $5
22)       *ENDIF
23)   *ENDIF
24)   *IF $3 = Y AND $4 = Y
25)       PRINT RESULT BY REGION @ TOT SALES SUBTOT BY $5
26)       *END
27)   *ENDIF
28)   *IF $3 = Y
29)       PRINT RESULT BY REGION @ TOTAL SALES
30)       *END
31)   *ENDIF
32)   PRINT RESULT BY REGION SALES
```

Explanation of Example:

Lines 1-4 are comments; they are not written to the work file

Line 5 sets the value of parameter $4 to N. If $4 is not given a value, the ***IF** on line 24 will cause an error. The user may change $4 on line 17.

Lines 6 and 7 display a message with the two dates entered in the **RUN** command.

Line 8 requests confirmation of these dates.

Lines 9-11 test the user response, and abort the procedure if the response is not "Y".

Line 12 writes the **SELECT** command to the work file.

Lines 13 and 14 determine if totals are desired in the report. Note that parameter $3 is reused.

If totals are not desired, the scan skips to line 32, because of the ***IF** directives on lines 15, 24, and 28. Two commands will be written to the work file and executed:

```
SELECT SALEDATA ST DATE GE 1/1/81 AND DATE LE 6/1/81
PRINT RESULT BY REGION SALES
```

If totals are desired, lines 16-17 ask if subtotals are desired.

If subtotals are requested, the ***IF** directive on line 18 is satisfied, and the **SUBTOTAL** data items are requested (lines 19 and 20).

Once the subtotal items are entered, the data item names ($5) are substituted in the **SORT** command on line 21, and it is written to the work file.

If both totals and subtotals have been requested, (***IF** directive on line 24), the **PRINT** statement on line 25 is written to the work file, with the **SUBTOTAL** data items ($5) substituted. No further scanning will occur, because of the ***END** directive on line 26.

If only totals are requested, the ***IF** directive on line 24 causes scanning to skip to line 28. The ***IF** directive on line 28 is satisfied, and the **PRINT** statement on line 29 is written to the work file. The ***END** directive on line 30 halts scanning.

Note that the line numbers are used here only for reference; they must not be entered into the command procedure file.

RESTARTING A COMMAND PROCEDURE

If no error conditions are encountered during execution of a command procedure, all commands in the work file will be executed, and control is returned to Series 20. If one of the commands in the work file completes abnormally (aborts), however, execution will stop immediately, and the work file will contain all commands following the one that was aborted. Possible reasons for abnormal termination include:

1. The operator enters a control-C

2. A command terminates abnormally (e.g. the user enters **N** to a confirmation to erase a database

3. A power failure or hardware reset occurs

4. An invalid command line is encountered

An aborted command procedure can be restarted. However, before restarting, the command procedure should be reviewed to identify the command that was aborted. The user must then determine whether to repeat the command that was aborted. For example, repeating an **APPEND**, **POST** or **READ** command may cause undesired effects.

Once the status is determined, execution of commands from the work file may be continued using the **RESTART** command:

RESTART

If the work file is on a drive other than the current drive, the command must be entered as **RESTART drv:** (e.g. **RESTART B:**). The current drive will not be the same as when the command procedure aborted if the current drive was reset by the command procedure.

EDITING COMMAND PROCEDURES

Condor 20 command procedures are created and edited using the **EDIT** command. This command provides a powerful, pushbutton-oriented "on-screen editor" for the HP Series 100.

The syntax of the EDIT command is:

EDIT procedurename

After you enter the **EDIT** command, the screen is blanked, and a ">" prompt is displayed, indicating that the **EDIT** command is ready to accept the first line of your command procedure file. If you are editing a command procedure file that already exists, the first several lines of the file will appear on the screen, again ready for editing. The lines of the command procedure are displayed on every other line of the display to make inserting lines easier.

The HP Series 100 editing keys can be freely used to edit the command procedure. New command procedure lines are entered line by line, pressing the **RETURN** key to move to the next line. You may use the arrow keys to return to a previously-entered line and correct mistakes simply by "typing over" the text in error on the screen. In addition, the editing keys provide useful editing functions:

DEL CHAR -- deletes one character, at the cursor position.

DEL LINE -- deletes 80 characters, starting at the cursor position, and wrapping around to the next line.

INS LINE -- inserts 80 blank characters, starting at the cursor position and wrapping around to the next line.

The **INS CHAR** key toggles the **EDIT** command back and forth between **replace** mode (the normal mode) and **insert** mode. These modes operate in the same way for the EDIT command as described for the **FORMAT** command. See Chapter 9 for details.

In addition to the editing keys, the **EDIT** command displays softkeys to aid in command procedure editing:

Press **NEXT PAGE** to bring the next "screenful" of the command procedure file onto the screen for editing.

Press **PREVIOUS PAGE** to bring the previous "screenful" of the command procedure file onto the screen for editing.

Press **TOP PAGE** to bring the first "screenful" of the command procedure file onto the screen.

Press **BOTTOM PAGE** to bring the last "screenful" of the command procedure file onto the screen.

Press **REFRESH SCREEN** to cause Series 20 to re-paint the screen, if it becomes garbled.

Press **PRINT** to print the edited command procedure file on the printer.

Press **ABORT** to stop editing the command procedure and leave the command procedure file on disc unchanged.

Press **END** to stop editing the command procedure and replace the command procedure file on disc with the edited version.

In addition to Condor 20's built-in editor, the CP/M editor (ED), Series 100/Word or Wordstar/100 (in NON-DOCUMENT mode) may be used to edit command procedures.

HELP SCREENS

Condor 20 **HELP** screens simplify computer operations by giving the user a menu selection among processing options. The **HELP** screen lists, by number, the choices available to the user; each choice corresponds to a Condor 20 command. Pressing the appropriate number on the keyboard automatically executes the associated command (often a **RUN** command), without any entry of commands by the user. Condor 20 application designers will find **HELP** screens to be useful "glue" that holds their individual application modules together.

Figure 10-1 shows how **HELP** screens can simplify processing in a routine daily application such as Accounting.

```
**************************************************************
*        GENERAL LEDGER ACCOUNTING SYSTEM                    *
**************************************************************
*                                                            *
*             SELECT ONE OF THE FOLLOWING OPTIONS            *
*                                                            *
*     1.  INPUT CHART OF ACCOUNTS                            *
*                                                            *
*     2.  UPDATE GENERAL LEDGER                              *
*                                                            *
*     3.  ENTER JOURNAL ENTRIES                              *
*                                                            *
*     4.  JOURNAL TRIAL BALANCE                              *
*                                                            *
*     5.  POST GENERAL LEDGER                                *
**************************************************************
```

Figure 10-1. A HELP Screen

The **HELP** screen is requested by the user with the **HELP** command:

HELP helpfile

When the **HELP** menu has been displayed, an option is selected by typing a number followed by a carriage return. Condor 20 executes a command (or command procedure) that is associated with the number typed.

The command associated with each choice on a **HELP** screen is defined when the **HELP** screen is created by the application designer. The associated command is placed in square brackets ([ ]) following the number on the screen. Figure 10-2 shows the **HELP** screen from figure 10-1, as defined by the application designer. The bracketed commands are automatically removed when the **HELP** screen is displayed.

The commands enclosed in brackets on a **HELP** screen may be any Condor 20 command except **DIC, DIR, LOG, RESTART, SET** and **SYSTEM.** These "built-in" commands are an integral part of Series 20, and may be selected through a **HELP** screen by causing the appropriate selection to **RUN** a command procedure that includes them.

```
**********************************************************************
*         GENERAL LEDGER ACCOUNTING SYSTEM                           *
**********************************************************************
*                                                                    *
*              SELECT ONE OF THE FOLLOWING OPTIONS                   *
*                                                                    *
*     1.  INPUT CHART OF ACCOUNTS  [RUN B:ACCNTS]                    *
*                                                                    *
*     2.  UPDATE GENERAL LEDGER  [RUN B:UPGLEDG]                     *
*                                                                    *
*     3.  ENTER JOURNAL ENTRIES  [RUN B:JOURNIN]                     *
*                                                                    *
*     4.  JOURNAL TRIAL BALANCE  [RUN B:TRIALB]                      *
*                                                                    *
*     5.  POST GENERAL LEDGER  [RUN B:POSTGL]                        *
**********************************************************************
```

Figure 10-2. A HELP Screen

HELP screens are created using the **FORMAT** command. Enter the command:

FORMAT filename.HLP

and use the HP Series 100 editing keys to "paint" the **HELP** form on the screen. You may also use the CP/M editor, Series 100/Word, or Wordstar/100 (in NON-DOCUMENT mode) to edit HELP screens. See Chapter 9 for a description of the **FORMAT** command and its editing capabilities.

ABORTING A HELP SCREEN

A **HELP** screen may include an option to abort the **HELP** screen and return to the Series 20 command prompt by including the **ABORT** command as one of the choices in the screen. The **ABORT** command serves only in this special function, and consists of the single word:

ABORT

CHAPTER SUMMARY

* Series 20 command procedures and **HELP** screens are used to create applications systems, or to ease repetitive database chores.

* The **RUN** command processes a command procedure, a pre-stored sequence of Series 20 commands and special directives.

* **RUN** command directives allow for parameter-driven processing, conditional processing, and user interaction during command procedures.

* **HELP** screens provide a pushbutton-oriented menu choice for the user; each choice executes a corresponding command on the user's behalf.

CHAPTER 11

TRANSACTION PROCESSING WITH CONDOR SERIES 20

Chapter 10 showed how HELP screens and Series 20 command procedures can be used to streamline repetitive Series 20 procedures. In the hands of a data processing professional, these features also make Series 20 an attractive foundation for a complete transaction processing application. This chapter describes the use of Series 20 for transaction processing, using a basic General Ledger Accounting System as a comprehensive example. Design considerations for transaction processing with Series 20 are presented through the example. In addition, the application provides an excellent, integrated summary of the Series 20 capabilities discussed in earlier chapters.

GENERAL LEDGER APPLICATION EXAMPLE

The example used throughout this chapter is a simple General Ledger Accounting System. The application uses three databases, with the following names and functions:

GLEDGER -- the General Ledger Database; stores data on each account in the General Ledger (account#, name, balance, etc.).

JOURNAL -- the General Journal Database; stores journal entries as they are entered, until they are eventually posted to the General Ledger accounts

JOURAUDT -- The Journal Audit Trail Database; stores all journal entries after they are posted, providing an audit trail of all general ledger transactions.

The application accumulates journal entries as they are made, verifies their correctness, and ultimately posts them to the appropriate general ledger accounts. Various reports are also generated.

DEFINING THE EXAMPLE DATABASES

The first steps in designing the example application is to define the databases. Figure 11-1 shows the **GLEDGER** database form. The **DEFINE** command is used to "paint" the form on the HP 125 screen, and define the database to Series 20.

```
**********************************************************************
*              GENERAL LEDGER ACCOUNTING SYSTEM                      *
**********************************************************************
*                                                                    *
*           GENERAL LEDGER RECORD                                    *
*                                                                    *
*      [ACCOUNT]:_________      [ASSET]:__   [LIABILITY]:__          *
*                                                                    *
*      [ACCNAME]:_____________________________                       *
*                                                                    *
*      [DEBIT]:_____________      [CREDIT]:_____________            *
*                                                                    *
*      [YTDAMT]:_____________ (Year - To - Date Amount)              *
*                                                                    *
*      [MONTH]:_____________ (Last Accounting Month                  *
*                             Posted to General Ledger)              *
**********************************************************************
```

Figure 11-1. GLEDGER Database Form

The **JOURNAL** database is also defined to Series 20 using the **DEFINE** command. Its database form is shown in Figure 11-2.

```
**********************************************************************
*              GENERAL LEDGER ACCOUNTING SYSTEM                      *
**********************************************************************
*                                                                    *
*           JOURNAL RECORD                                           *
*                                                                    *
*      [ACCOUNT]:_________                                           *
*                                                                    *
*      [DESC]:_____________________________                          *
*                                                                    *
*      [DEBIT]:_____________      [CREDIT]:_____________            *
*                                                                    *
*      [MONTH]:_____________ (Accounting Month)                      *
*                                                                    *
*      [DATE]:_____________ (Date of Journal Entry)                  *
**********************************************************************
```

Figure 11-2. JOURNAL Database Form

The **JOURAUDT** database, consisting of journal records which have already been posted, has the same data items as the **JOURNAL** database. The **JOURAUDT** database can therefore use exactly the same form and the same data item definitions as the **JOURNAL** database. This is also specified through the **DEFINE** command:

Type **DEFINE JOURAUDT <RETURN>**
(Series 20 will indicate that no form file is found, and ask if you want to create a new one. Responding **N** signals that an existing form file will be used; Series 20 asks for its name:

Type **JOURNAL <RETURN>**
(Series 20 next asks if a new data item definitions file is to be created. Responding **N** signals that an existing definition file will be used; Series 20 asks for its name:)

Type **JOURNAL <RETURN>**
(Series 20 creates the JOURAUDT database, with its own unique data file (JOURAUDT.DAT), but with a shared form file (JOURNAL.FRM) and data item definition file (JOURNAL.DEF). The Data Dictionary entry reflects these file names.

The above steps may also be accomplished with the one line command:

DEFINE JOURAUDT JOURNAL.FRM, JOURNAL.DEF <RETURN>

This completes the definition of the example databases.

PREPARATION FOR ENTERING DATA INTO THE EXAMPLE DATABASES

In any transaction processing application, there are typically two types of data entry operations; entry of new records in a "master" database, and entry of new "transaction" records. In addition, data in a "master" database is generally updated from one or more "transaction" databases through a "transaction posting" operation. These three operations are described below, and illustrated with the General Ledger example in the sections that follow.

Entering Unique Records (Generally **master** records)

When entering new records to an existing database, duplicate records may not be allowed. For example, in general ledger account records, two records having identical account numbers are not allowed.

Entering Matching Records (Generally **transaction** records)

When entering records to a "transaction" database, there must be a "matching record" in the "master" database. For example, a payment from a client would be entered in an accounts receivable database with an account number, which must match the account number of a record in the general ledger database.

Posting (Generally transactions are posted to master records)

New records that have been entered into a "transaction" database are posted to records in a master database. The database posted to (master database) has unique records, and the database with posting data (transaction database) may have multiple records, though it is restricted to matching records in the master database. One example is a purchase order receipt transaction that updates the purchase order database.

ENTERING UNIQUE RECORDS

Inputting the General Ledger Chart of Accounts illustrates the entry of unique records.

In the general ledger "master" database, **GLEDGER**, there is to be one record for each general ledger account number. If the same account number is entered into the database twice, it is an error. To eliminate this error, a data entry procedure is followed which will ensure that only unique records are added to the database.

To ensure unique records, new records are never entered directly into a master database. A holding database is defined into which new records are entered. Upon completion of entry into the holding database, the records in the holding database are compared to the records in the master database to find duplicates. Only the unique records are appended to the master database. The holding database need only be defined once. After each entry of data, the holding database is emptied.

Example:

The procedure outlined below enters unique records into the GLEDGER database:

1. DEFINE GLTEMP

GLTEMP will be the holding database. The system will ask if a new form is to be created. Answer **N**, and tell Series 20 that GLTEMP uses the same form as the GLEDGER database, GLEDGER.FRM.

The system will ask if a new definition file is to be created. Answer **N**, and tell Series 20 that GLTEMP will use the same data item definitions as the GLEDGER database, from GLEDGER.DEF.

2. ENTER GLTEMP

New account numbers and other information are entered through the GLEDGER form, and accumulated in the GLTEMP holding database. All new accounts are added in this way; existing accounts may be updated directly with the **UPDATE GLEDGER WHERE ACCOUNT IS xxxxxx** command. **(Note: For proper accounting practice, always enter Credits, Debits, and Totals in the Journal database, never in the General Ledger database. An accountant can give you additional information on accounting practices.)**

3. COMPARE GLTEMP GLEDGER NOT MATCHING ACCOUNT

The holding database is compared to the master database to find account numbers that do not match. These records are written to the **RESULT** database, and are valid accounts to be added to the master database. The **RESULT** database has the same form and data item definitions as the GLEDGER database.

4. TITLE T,'GENERAL LEDGER REPORTS',S,S,
'NEW ACCOUNT NUMBER ADDITIONS FOR ',DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT

A report of the new accounts to be added is printed. The headings and date will be printed at the top of each page of the report. The heading is printed on two lines, and the **S**'s will cause lines to be skipped.

5. APPEND GLEDGER RESULT

New and unique general ledger account numbers are appended to the master (GLEDGER) database. The **APPEND** command appends the **R$.DAT** file to the **GLEDGER.DAT** file.

6. SORT GLEDGER BY ACCOUNT

The GLEDGER database is sorted in account number sequence.

7. TITLE T,'GENERAL LEDGER REPORTS',S,S,
'GENERAL LEDGER CHART OF ACCOUNTS DATE ',DATE,S,S
PRINT GLEDGER BY ACCOUNT ACCNAME DEBIT CREDIT

A revised chart of accounts report is printed in account number sequence. The headings and date are printed at the top of each page.

8. EMPTY GLTEMP

The new accounts have been added, and the GLTEMP database can be emptied.

ENTERING MATCHING RECORDS

Inputting journal entries illustrates the entry of matching records.

Journal entries in the general journal database must match, by account number, records in the general ledger database. If a record entered in the JOURNAL database does not match a record in the GLEDGER database, it is an error.

Before being posted to GLEDGER, JOURNAL records are compared to the GLEDGER records to verify that they match an account number in the chart of accounts. Those JOURNAL records that do not match are corrected before posting.

Example:

The procedure outlined below enters matching records into the JOURNAL database.

1. ENTER JOURNAL

Journal transactions are entered, using the JOURNAL.FRM form.

2. SORT JOURNAL BY ACCOUNT
SORT GLEDGER BY ACCOUNT

Sorting the JOURNAL and GLEDGER databases before comparing and posting can speed the time for these commands by a large margin over unsorted operation.

3. COMPARE JOURNAL GLEDGER NOT MATCHING ACCOUNT

JOURNAL records are compared to GLEDGER records by matching ACCOUNT. JOURNAL records that do not match are written to the **RESULT** database. The form and data items for the **RESULT** database are the same as for the JOURNAL database.

4. TITLE T,'GENERAL JOURNAL REPORTS',S,
JOURNAL EXCEPTIONS FOR ',DATE
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT

Journal records that do not match a record in the GLEDGER database are printed in an exception report. The problem in either the GLEDGER or JOURNAL database is determined, and the respective databases are updated, if needed. If either database is updated, the procedure restarts with step #2 above.

Entry of correct JOURNAL transactions is complete when the printed report shows zero records; that is, when all JOURNAL records match records in the GLEDGER database. The next step is posting the JOURNAL to the GLEDGER database.

POSTING TRANSACTIONS TO A MASTER DATABASE

The posting of journal entries to the general ledger serves as an example of transaction posting.

In the general ledger accounting system, journal entries are collected for a month accounting period. At the end of a month, the journal entries are posted to the general ledger. After posting, journal entries are accumulated in an audit trail database, and the JOURNAL database is emptied in preparation for the next month's entries.

Example:

The procedure outlined below posts journal entries to the general ledger.

1. STAX JOURNAL BY DEBIT, CREDIT

Statistics are calculated for DEBIT and CREDIT, showing the total debits and credits in the JOURNAL database. For a "trial balance", the total credit must equal the total debit. If these totals are not equal, the error must be found and the journal file updated with the **UPDATE** command.

2. POST GLEDGER JOURNAL BY ACCOUNT AND ADD DEBIT, CREDIT REPLACE MONTH

For each matching ACCOUNT, DEBIT and CREDIT amounts from the JOURNAL database record are added to the DEBIT and CREDIT amounts in the correct GLEDGER record. The MONTH in the GLEDGER record is replaced by the MONTH from the JOURNAL record.

3. COMPUTE GLEDGER ST YTDAMT = DEBIT - CREDIT

An account balance (YTDAMT) is calculated by subtracting CREDITs from DEBITs.

4.
```
SELECT GLEDGER ST ASSET IS Y
TITLE T,'GENERAL LEDGER MONTHLY REPORTS',S,
   'ASSET ACCOUNTS REPORT FOR ',DATE,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT
   AND COMPUTE TOTAL YTDAMT
```

General ledger records for asset accounts are selected and copied to the **RESULT** database. A report showing asset account debits and credits, and the total of the asset accounts is printed. The total will be compared to the total liabilities in the next report.

5.
```
SELECT GLEDGER ST LIABILITIES IS Y
TITLE T,'GENERAL LEDGER MONTHLY REPORTS',S,
   'LIABILITIES ACCOUNTS REPORT FOR ',DATE,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT
   AND COMPUTE TOTAL YTDAMT
```

Liabilities accounts are selected, and the corresponding report is printed for liabilities. If the total year-to-date totals from the assets and liabilities reports are not equal, the general ledger database is in error. Correcting entries must be entered into the JOURNAL database and the JOURNAL re-posted to the GLEDGER as described above. Prior to any adjusting entries, the JOURNAL database is emptied as described below.

6.
```
APPEND JOURAUDT JOURNAL
EMPTY JOURNAL
```

Finally, the JOURNAL database's journal entries are added to the audit trail database (JOURAUDT), and the JOURNAL database is emptied for the next cycle.

TRANSACTION PROCESSING HELP SCREENS

The General Ledger Accounting System could be used repetitively, day by day, by clerical personnel following the procedures in the preceding sections. However, daily use of the application would be greatly simplified through the use of a **HELP** screen. The **HELP** screen in Figure 11-3 illustrates how simple data entry and posting can become with this approach.

```
**************************************************************************
*          GENERAL LEDGER ACCOUNTING SYSTEM                              *
**************************************************************************
*                                                                        *
*      SELECT ONE OF FOLLOWING OPTIONS                                   *
*                                                                        *
*                                                                        *
*      1.  INPUT CHART OF ACCOUNTS  [RUN B:ACCNTS]                       *
*                                                                        *
*      2.  UPDATE GENERAL LEDGER  [RUN B:UPGLEDG]                        *
*                                                                        *
*      3.  ENTER JOURNAL ENTRIES  [RUN B:JOURNIN]                        *
*                                                                        *
*      4.  JOURNAL TRIAL BALANCE  [RUN B:TRIALB]                         *
*                                                                        *
*      5.  POST GENERAL LEDGER  [RUN B:POSTGL]                           *
**************************************************************************
```

Figure 11-3. General Ledger HELP Screen

The **HELP** Screen is defined with the **FORMAT** command:

```
FORMAT ACCNTG.HLP
```

Note that when the **HELP** screen is displayed in response to the **HELP** command, for function selection, the information in square brackets on the screen is not displayed.

EXAMPLE COMMAND PROCEDURE FILES

The **HELP** screen of Figure 11-3 responds to each selection by **RUN**ning a corresponding Series 20 command procedure file. The example requires five command procedures:

ACCNTS.CMD -- Entering Chart of Accounts

UPGLEDG.CMD -- Updating General Ledger

JOURNIN.CMD -- Entering General Journal

TRIALB.CMD -- General Journal Trial Balance

POSTGL.CMD -- Posting Journal to General Ledger

Each command procedure would be created in turn using the **EDIT** command. The command procedures to implement the required functions are given below.

1. ACCNTS.CMD COMMAND PROCEDURE

```
COMPARE GLTEMP GLEDGER NOT MATCHING ACCOUNT
TITLE T,'GENERAL LEDGER REPORTS',S,S,'NEW ACCOUNT
   NUMBER ADDITIONS DATE', DATE,S,S
PRINT RESULT BY ACCOUNT NAME DEBIT CREDIT
APPEND GLEDGER RESULT
COMPARE GLTEMP RESULT NOT MATCHING ACCOUNT
TITLE T,'GENERAL LEDGER REPORTS',S,S'DUPLICATE
   ACCOUNT NUMBERS',DATE,S,S
PRINT RESULT BY ACCOUNT NAME DEBIT CREDIT
SORT GLEDGER BY ACCOUNT
TITLE T,'GENERAL LEDGER REPORTS',S,S,'GENERAL
   LEDGER',DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT
EMPTY GLTEMP
HELP ACCNTG
```

2. UPGLEDG.CMD COMMAND PROCEDURE

```
UPDATE GLEDGER
SORT GLEDGER BY ACCOUNT
TITLE T,'GENERAL LEDGER REPORTS',S,S,
   'GENERAL LEDGER CHART OF ACCOUNTS DATE',DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT
HELP ACCNTG
```

3. JOURNIN.CMD COMMAND PROCEDURE

```
ENTER JOURNAL
SORT JOURNAL BY ACCOUNT
SORT GLEDGER BY ACCOUNT
COMPARE JOURNAL GLEDGER BY ACCOUNT
*MESSAGE 'ENTER ACCOUNTING MONTH'
*GET $1
TITLE T,'GENERAL LEDGER REPORTS FOR $1',PAGE,S,S,
   'JOURNAL EXCEPTIONS FOR DATE', DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT
```

4. TRIALB.CMD COMMAND PROCEDURE

```
*MESSAGE 'ENTER ACCOUNTING MONTH'
*GET $1
TITLE T,'GENERAL LEDGER REPORTS FOR $1',PAGE,S,S,
   'JOURNAL TRIAL BALANCE FOR DATE ',DATE,S,S
STAX JOURNAL BY DEBIT CREDIT [T]
TITLE T,'GENERAL LEDGER REPORTS FOR $1',PAGE,S,S,
   'GENERAL LEDGER TRIAL BALANCE FOR DATE',DATE,S,S
STAX GLEDGER BY DEBIT CREDIT [T]
HELP ACCNTG
```

5. POSTGL.CMD COMMAND PROCEDURE

```
*MESSAGE 'ENTER ACCOUNTING MONTH'
*GET $1
POST GLEDGER JOURNAL BY ACCOUNT AND ADD DEBIT, CREDIT
   AND REPLACE MONTH
COMPUTE GLEDGER ST YTDAMT = DEBIT - CREDIT
SELECT GLEDGER ST ASSET IS Y
TITLE T,'GENERAL LEDGER REPORTS FOR $1',
   PAGE,S,S'ASSETS ACCOUNT REPORT FOR DATE',DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT AND
   COMPUTE TOTAL YTDAMT
SELECT GLEDGER ST LIABILITY IS Y
TITLE T,'GENERAL LEDGER REPORTS',DATE,S,S,
   LIABILITIES ACCOUNT REPORT FOR DATE',DATE,S,S
PRINT RESULT BY ACCOUNT ACCNAME DEBIT CREDIT AND
   COMPUTE TOTAL YTDAMT
HELP ACCNTG
```

DAILY TRANSACTION PROCESSING

This concludes the steps required to set up the General Ledger example for use. Clerical personnel can now enter the single command:

HELP ACCNTG

to gain access to the **HELP** screen each morning. Further database processing takes place by selecting an option (from 1-5) from the **HELP** screen, and using the database form for data entry and update. Of course, if an unanticipated inquiry arises, Series 20's powerful ad hoc capabilities can be used to answer it quickly, drawing on the data stored in the three databases. It is this combination of easy daily operation with powerful ad hoc facilities that makes Series 20 an attractive "programming language" for implementing transaction processing applications.

CHAPTER 12

WRITING REPORTS

In the preceding chapters, you learned about commands that produce simple reports: LIST, PRINT, STAX, TABULATE, and TITLE. Outside of building a form file, you cannot, however, specify the format of a report using these commands. To specify the format of a report, use the **Report Writer** included in Condor 20-3 (accessed by entering the **REPORT** command).

With the Report Writer, you control of all aspects of formatting a report:

- Special formats and columnar formats may be combined in a single report.
- Column headings of one or more lines may be specified independent of the data item names.
- The printing width and format of the data items may be selected or changed at will.
- Labels, text, controlled section breaks, and other special features, such as computed items and totals, may be included in a report specification.

In addition to formatting, Report Writer in Condor 20-3 enables you to print and list reports. Several report specifications may be created for a single database, or one report specification may be used with several different databases. If a report format is used with more than one database file, each of the database definitions must have the same data items which are used in the report specification.

This chapter will teach you how to use the Report Writer to create, modify, and describe a report specification as well as print reports. The chapter will also display how to invoke the Report Writer using or not using its command menu, and how to access the Report Writer through command procedures.

All of the details for creating a report specification are contained in this chapter. Appendix F shows a step-by-step example of how to create the Warehouse report that is displayed later in this chapter.

SETTING UP YOUR CONDOR 20-3 DISCS

Condor 20-3 consists of Condor 20-2 plus database indexing capabilities and an advanced yet simple-to-use report writer. Before you begin to use the Condor Report Writer, you must "install" the 20-2 disc and the 20-3 section disc onto Work Discs in everyday operation. If you have a Winchester or 8" disc drive (HP 9895), you will be able to place all of Condor 20-3 onto one disc drive (if there is space left on the disc). Chapter 2 of this manual specifies how to install your 20-2 and 20-3 discs so that you may begin using them.

ABOUT USING THE REPORT WRITER

The Report Writer produces a range of reports, including:

- Columnar reports
- One record per page, full page formats
- Reports requiring subtotals, totals, and/or averages
- Mailing labels
- Reports which require section breaks

The concept of the Report Writer is straightforward: you create a report specification for a database and that specification is used to print a report. You may save the report specification so that the report can be printed at any time.

The Report Writer will reference a database's .DEF (definition) file during the creation of a report specification. Its .DAT (data) file is utilized during report printing. Report Writer never refers to the database .FRM (form) file.

After creating a report specification, two files will be produced, **database.RPT** and **filename.FRM**. You will be asked to name both of them.

database.RPT - This will be your saved report specification. Whenever revising, printing, or describing a report, this file will be referenced.

filename.FRM - This will be a report form which you build as a model to create a report specification. It should be saved in case the report specification must be re-created. **Caution: This form is different from the database form and should not be given the same name. Assigning the same name will cause the database form to be lost, forcing you to create it again.**

The Report Writer is set up to allow you to quickly define a report specification and then edit it to meet your exact requirements. Designing a report with the Report Writer is similar to using the DEFINE command to define a database. You begin by creating a model of the report form on your screen. Then, the Report Writer uses the report form and responses from you to specify the format and contents of the report.

A report specification may be revised after it has been created by using the revise option of the Report Writer. Page length and margins can be altered, lines can be repositioned within the report, and text and data items can be repositioned within a line. However, you will not be able to insert or delete new lines of text or data or correct spelling errors. If a spelling error is detected on the report form (.FRM) file when creating a report, that form will have to be changed and the report specification created over again.

When you are ready to view the report with data in it, you may send the report to the printer (print the report), or to the screen (list the report). In addition, you may choose a particular set of data records from your database to be included in the report. This is done by specifying a WHERE condition in the Report Writer, similar to the WHERE condition in the DELETE, DISPLAY, SELECT, and UPDATE commands.

The best way to understand the Condor Series 20 Report Writer is by using it to create an actual report specification. The following page (Fig. 12.1) shows a sample warehouse inventory report organized by division and warehouse. It has titles, category headings, subtotals, and a grand total, along with basic information. Fig. 12.2 shows the screen format of the Warehouse database used to create this report and its attribute summary. You will probably want to create your own warehouse database so that you can follow along with the examples given for using the Report Writer.

INVOKING THE REPORT WRITER BY WAY OF THE MENU

(Note: All examples assume that drive B: contains the databases to be used with the Report Writer. If you have placed your databases on another drive, substitute that drive letter wherever B: or B>> is shown in these examples.)

Once you have set up your discs and have created your database, you are ready to begin using the Report Writer. First, set the current drive to the one with your data disc by typing the following:

B: <RETURN>

With your Report Writer disc in drive A:, Condor 20-3 will respond with the prompt: **B>>**. Type the following:

REPORT database-name <RETURN>

If the database does not already exist, or if it is not listed

WAREHOUSE UPDATE REPORT

Item Number	Item Description	Prev. On Hand	Shipped	Received	Current On Hand	Unit Price	Total Value
DIVISION 1		WAREHOUSE 1					
1	SHOVELS	25	7	4	22	5.80	127.60
2	TROWELS	20	100	60	-20	3.33	-66.60
5	RED STONES	87	91	23	19	4.71	89.49
7	YELLOW STONES	97	7	14	104	4.58	476.32
8	BLACK STONES	50	19	23	54	4.88	263.52
			WAREHOUSE 1		TOTAL		890.33
DIVISION 1		WAREHOUSE 2					
1	SHOVELS	27	9	14	32	5.80	185.60
2	TROWELS	15	6	0	9	3.33	29.97
5	RED STONES	93	75	12	30	4.71	141.30
7	YELLOW STONES	47	14	11	44	4.58	201.52
8	BLACK STONES	3	8	31	26	4.88	126.88
			WAREHOUSE 2		TOTAL		685.27
			DIVISION 1	TOTAL			1575.60
DIVISION 2		WAREHOUSE 1					
11	BLUE CEMENT	12	10	2	4	3.40	13.60
12	YELLOW CEMENT	30	10	5	25	4.50	112.50
8	BLACK STONES	50	1	34	83	4.50	373.50
			WAREHOUSE 1		TOTAL		499.60
DIVISION 2		WAREHOUSE 2					
11	BLUE CEMENT	40	12	3	31	3.40	105.40
			WAREHOUSE 2		TOTAL		105.40
			DIVISION 2	TOTAL			605.00
			GRAND TOTAL				2180.60

(Fig. 12.1)

SCREEN FORMAT OF WAREHOUSE DATABASE

```
[DIVNBR] _                [WHSENBR] _
 (division number)         (warehouse number)

[NO] __                   [ITEM] _____________
 (item number)             (item description)

[PREQTY] __               [SHIPPED] ___          [RCVED] __
 (previous quantity)       (number shipped)       (number received)

[PRICE] ______
```

****

Number of underscores in screen format

[DIVNBR]	1
[WHSENBR]	1
[NO]	2
[ITEM]	13
[PREQTY]	2
[SHIPPED]	3
[RCVED]	2
[PRICE]	6

****

Attribute summary of Data Base WAREHSE

```
1.DIVNBR: N,1,1,5,"       "
2.WHSENBR: N,1,0,5,"       "
3.NO: N,1,0,127,"       "
4.ITEM: AN,13,0,13,"       "
5.PREQTY: N,1,0,127,"       "
6.SHIPPED: N,1,0,127,"       "
7.RCVED: N,1,0,127,"       "
8.PRICE: $,3,.00,327.67,"       "

Record Size (Bytes) = 23
Total Records = 0
```

(Fig. 12.2)

in the data dictionary, the following error message is displayed:

Title not found in dictionary
****REPORT ABORTED****

If you receive this message, make sure that you have your database on the disc in your current drive. Once you have ensured this and performed LOGDISK (described in Chapter 7), type the REPORT command again.

If the database does exist and is listed in the data dictionary, the system displays the menu shown below. In our example, we assume that a database named **WAREHSE** already exists.

```
**************************************************************
*                                                            *
*           CONDOR SERIES 20 RDBMS REPORT WRITER             *
*                  Version 1.xx**xx                          *
*                                                            *
*                  Choose option                             *
*                                                            *
*   Create New Report Specification          (C)             *
*   Describe existing Report Specification   (D)             *
*   Revise existing Report Specification     (R)             *
*   Print or Display Database Report         (P)             *
*                                                            *
*   Enter option or End <C/R>:                               *
*                                                            *
**************************************************************
```

At this point, you must decide whether you want to create a new report specification, describe (look at) an existing report specification, revise an existing report specification, or print a database report. Once you have decided, go on to the appropriate section.

CREATING A REPORT SPECIFICATION

To create a report specification, press the letter **C** in the Report Writer menu.

The process of creating a report specification is divided into two parts: 1) building a report model form, and 2) generating a report specification around the model form. Building a report form is much like creating a database form. Generating the

report specification is a matter of responding to questions asked by Report Writer. You may select a long or short method of creating a report specification.

You are first asked if you want to use the **short method** of creating the report specification:

Do you want to use the short method (Y/N)?

Answer **Y** if you wish to use the short method of creating you report specification. Answer **N** if you would rather use the long method.

Short Method: This method is used to quickly define a report specification. You are only asked to specify: 1) The line type of each model line on your report form, 2) Details about calculated items, and 3) Control items for section breaks. After quickly defining the report specification, you are given the option to immediately revise it, so you may define special line spacing and other forms of fine tuning. Alternatively, you may choose to not revise at this point, but rather print portions of the report to check the layout. If you desire revisions after printing, you may do that by entering the Revise option at the Report Writer menu.

Long Method: This method asks many more questions and allows you to completely define your report specification as well as refine it during the creation process. After defining the report layout using the long method, you will probably want to print portions of the report and revise it if the results are not satisfactory. **This method must be used if you wish to print reports with greater than 80 columns** (see "REPORT WRITER SPECIFICATIONS AND HELPFUL HINTS" near the end of this chapter when you wish to create such a report).

A. Creating a Report Form

The next question Report Writer asks you is:

Do you wish to create a new report format (Y/N)?

A report format is the model form used to create a report

specification. If you specify **N**, Report Writer asks you for the name of a report model form that already exists. Upon supplying the name of the report form, you are given the option to modify it or create a report specification around the unchanged form.

Since you have not previously created a report form, type **Y** for "yes". The screen changes, giving you the same blank screen with softkeys that you used in creating a database form.

A report form is similar to a database screen form, as follows:

- Data items are identified by placing them in brackets.
- Text, page numbers, and other types of information are typed in without brackets.

However, unlike a database screen form, underscores are not used.

Lines created within the report form are called model lines. Each model line tells Report Writer which combination of data items, text, and other information you want printed on that line.

To produce the sample report seen in Fig. 12.1, enter the model lines exactly as they appear in Fig. 12.3. If you feel that a grid on the screen will help you lay out the report form, press

---

```
                        WAREHOUSE UPDATE REPORT

                                                                PAGE    $PAGE

Item     Item          Prev.                                    Current  Unit     Total
Number   Description   On Hand   Shipped     Received           On Hand  Price    Value
DIVISION [DIVNBR] WAREHOUSE [WHSENBR]
  [NO] [ITEM]          [PREQTY] [SHIPPED]    [RCVED]           [CURQTY] [PRICE] [VALUE]

                                  WAREHOUSE [WHSENBR] TOTAL                    [VALUE]
                          DIVISION [DIVNBR] TOTAL                              [VALUE]
                  GRAND TOTAL                                                  [VALUE]
```

(Fig. 12.3)

the **Grid** softkey. The **Refresh Screen** softkey erases the grid while leaving the items you have typed.

Only one model line in the report form is used for each different type of line in the final report, even though the same model lines may be printed more than once per page. You can create 24 different model lines, one for each line available on the Series 100 screen, more than enough for most reports. Each individual word or item in a model line becomes a specification, which can be moved within the line and printed in a variety of formats.

After you finish creating the report form, check the spelling and make sure no text or data items have been left out. A quick, efficient check just after you create the form will save you time; to correct a misspelled data item or title, you must return to the form and correct the model line, and then respecify the report. After the final check, press the **End** softkey to save the report form.

B. Indicating Line Printing Specifications

The next few sections provide you with information you need to build an entire report specification. Appendix F shows you, step-by-step, how to create the report specification for the Warehouse database.

After creating the report form, you begin structuring the report specification. For every model line in your form, you are asked to define its line type after seeing the following menu:

```
**************************************************
*                                                *
*  Describe when this line is to be printed:     *
*                                                *
*     Prints on first page only          [FP]    *
*     Prints on every page heading       [PH]    *
*     Prints on every line               [EL]    *
*     Prints before a section            [BS]    *
*     Prints after a section             [AS]    *
*     Prints on last page only           [LP]    *
*                                                *
*     Enter code                         [PH]:   *
*                                                *
**************************************************
```

[FP] - Lines printed on the first page only are normally used to describe a report.

[PH] - Page headings normally appear at the top of each page. They may include the system default values. **$PAGE** prints the current page number. **$TODAY** is the current data as set by the DATE command before entering the Report Writer. Page headings normally contain titles or other information, reporting periods, and column headings. A report may have more than one page heading model line, so multiple-line titles are possible.

[EL] - This indicates that the model line is to be printed once for each valid database record, and is used to print the detailed contents of the report that are repeating. It is possible to have more than one [EL] model line in the report form, creating a pattern of lines that will be printed once for each database record. You may limit the records that Report Writer uses. (This is explained in the section "PRINTING A REPORT", found later in this chapter.)

[BS] - A "Before Section" line is often used as a subtitle describing the group of lines that follow it. For instance, in our WAREHSE database report, sections pertain to particular warehouses and divisions. A "Before Section" must have one or more control items to indicate when the section is changed. A control item is a data item from the database. Each time the control item changes, a section break occurs, and the "Before Section" line is printed. More than one "Before Section" line can exist, each with its own set of control items. Up to 8 sets of control items are allowed.

[AS] - "After Section" lines are identical to "Before Section" lines, except thay are printed after the group of lines that make up a section. They are printed prior to any "Before Section" lines beginning a new section, and are most often used to accumulate and print subtotals, averages, minimums, maximums, and other types of information.

[LP] - These lines are printed only on the last page of the report. They are commonly used to print summary material and grand totals.

C. Item-type Specifications

Reports can have four different types of items:

Item Type	Use
TEXT	Titles, headings, descriptive text
SYSTEM	$PAGE, $TODAY
DATA	Read from database
CALCULATED or **COMPUTED**	Calculated by Report Writer

Report Writer can tell which type an item is, so you will not be asked about it.

Text items are placed in a report exactly as shown in the report form, and are **not** placed in square brackets ([ ]) on the model line. They are used for titles, headings, and as labels for system values, data items, and calculated items within the report.

System items consist of $PAGE and $TODAY keywords. **$PAGE** prints the current page number in a report. **$TODAY** prints the date which was set by the DATE command before entering Report Writer. These values are **not** placed in square brackets and normally appear in "page heading" lines.

Data items are read from the database file. They have the same name as the item in the database from which they are to be read. In the report form, they are placed within square brackets. Data items normally make up the bulk of a report, and appear in "before section" or "every line" model lines.

Calculated items are calculated by Report Writer from data items or constants, according to a formula which you will supply. They are placed within square brackets, and **cannot** have the same name as a database data item. In the WAREHSE example, CURQTY and VALUE are calculated items.

Other types of information are used to specify how an item is printed in the report:

1. **Starting Column**

   Short Method. Report Writer does not ask you for this data. All starting column positions will be determined directly from the model form. If items are expected to overlap, Report Writer will determine a recommended starting column and use it.

   Long Method. Report Writer asks you to supply the starting column for each item. A default value is displayed in square brackets ([ ]) and is based on the position of the item in the model line. To select the default, press <RETURN>. If items will overlap because of their lengths, Report Writer will issue a warning which recommends a starting position.

2. **Read from database or computed**

   Short Method. If an item name is not found in the database, Report Writer will ask if it is to be a computed item, with a **Y** or **N** response required. If **Y**, you will be asked to specify the item length, item type (date, numeric, or dollar), the equation for the item (discussed on page 12-14), and its printing format.

   Long Method. If an item name is found in the database, Report Writer will ask if the item is to be obtained from the database. A **Y** or **N** response is required. If you answer **Y**, item length, type, and format requested. If the item name is **not** found in the database, the procedure follows that of the short method above.

3. **Printing width**

   This specification is only asked in the long method or when specifying a computed item. The default for a database item is the number of underscores in the database screen format. A text item's default is the number of characters that appear in the item. For computed items, the default is the number of characters in the name of the item. For example, since the name of the item **[TOTAL]** is five characters, it will have a default printing width of 5 if **TOTAL** is a computed item.

4. Printing format

This is also only asked for in the long method or when working with a computed item. Data items and computed items can be printed in several formats. These formats are listed below, with descriptions and examples of printed output.

[SD] **SYSTEM DEFAULT** - Prints as in the LIST command.

[LJ] **LEFT JUSTIFY** - Aligns printing along the left margin of the printing width.
"EXAMPLE "

[RJ] **RIGHT JUSTIFY** - Aligns printing along the right margin of the printing width.
" EXAMPLE"

[NP] **NOT PRINTED** - Useful for accumulating subtotals without printing them, or for temporarily removing items from a report.

[LZ] **LEADING ZEROS** - Aligns numbers along the right margin of the printing width, and fills the preceding positions with zeros. Useful for numbers used as codes.
"00000001234"

[BZ] **BLANK IF ZERO** - Prints blanks instead of "0" or "0.00".

[FD] **FLOATING DOLLAR SIGN** - Places a dollar sign to the left of the number. If the number if negative, the minus sign is printed to the left of the dollar sign.
" $23.50"
" -$23.50"

5. Totals, Subtotals, Averages, Min, and Max

In a report, totals and averages will be computed items. If your total or average item appears in "after section" or "last page" type lines, you may only use the same computed item that appeared on "every line" type lines. If you do not want to print the calculation on every line, you can specify the item to be printed with the [NP] format (see "Printing formats" above). In the WAREHSE example, **VALUE** is used to accumulate subtotals and totals.

Choices for computed item types are:

VAL	Print the item's most recent value.
SUB	Print the subtotal. The subtotal is reset to zero whenver the section changes.
TOT	Print the accumulated total. This total is **not** reset to zero at every section change.
MIN	Print the minimum value found in the section.
MAX	Print the maximum value found in the section.
AVG	Print the average value for the section.

6. Computed item equations

Report Writer must have an equation or formula for each computed item. This equation must have the form:

itemname = term1 opr. term2 opr. ...

itemname is the name of the computed item. Only one item name can appear to the left of the equals sign.

The right side must have at least 2 terms and 1 operator, but no more than 32 terms and operators combined. Terms may be either numeric constants or data items read from the database.

Operators are:

+	addition
-	subtraction
*	multiplication
/	division

A data item used in a formula need not be part of the report specification, but it must exist in the database file. Equations are evaluated from left to right without regard for parentheses. **Note that this differs from the COMPUTE commmand.**

D. Section Breaks

You learned how to designate section breaks earlier in this chapter in the section entitled "Indicating Line Printing Specifications". The [BS] and [AS] line types allow you to group a section of common database values (for example, all orders for a product number, all transactions for particular dates, etc.). Each line that is printed within a section is most likely going to be an [EL] line type.

E. Use of Form Feeds

After you finish defining what the report will look like, Report Writer asks if you want it to send a form feed at the end of a report. If you are going to make multiple copies, this is the only way to make sure each copy begins at the top of the next page.

The form feed cannot be changed by the Revise option. Therefore, it is recommended that you select it unless you are **certain** it is unnecessary. Note that the top-of-form must also be properly set on your printer.

F. Report Specification Naming

Report Writer asks you for the name of the report specification that you will have created. The default is the name of the database, creating a file called **databasename.RPT**. You may specify a different name if you want, but it cannot exceed 8 characters.

If there is already a report specification with the same name, Report Writer asks for permission to overwrite the existing specification with the new one. The following message appears:

filename RPT exists already
Do you wish to destroy and re-create (Y/N)?

Answer **Y** if you wish to overwrite the old file. If you answer **N**, you are then asked to supply the name of a file where the report specification may be saved.

If a specification with a different name already exists for the current database, Report Writer links the new report specification to the database in the data dictionary without destroying the old report specification. Thus, a database can have more than one report specification, but only one report specification can be linked to the database at any one time.

Multiple reports can be accessed either by updating the data dictionary (data.dic) and changing the contents of the **REPORT** data item, or by entering the same database in the data dictionary several times, each time with a different title and report specification. This is illustrated on the top of the next page by a sample listing of the DIC command.

Title	Data	Form	Def	Index	Report
DATABASE	DATABASE	DATABASE	DATABASE		DATABASE
REPORT1	DATABASE	DATABASE	DATABASE		DATABASE
REPORT2	DATABASE	DATABASE	DATABASE		DATABASE

After naming the report specification, Report Writer will ask:

Do you wish to save the report form? (Y/N)

The report form is the model form that was built and used to create the report specification. It is advised that you save it, in case the report must be respecified. The name under which you save the report form **MUST NOT** be the same as that of the database. The report form will be a **.FRM** file, and would overwrite the **.FRM** file that already exists for that database.

G. Fine Tuning the Report Specification

If you used the short method of creating the report specification, you are given the option to make changes right after the creation process. Report Writer asks if you want to make a revision; if you respond in the affirmative, the system automatically takes you into the report specification revision process. (See the section entitled "REVISING A REPORT SPECIFICATION" for details on the revision process.) Since the short method of creating a report specification uses many defaults from the model form, you may want to alter the line spacing, item positioning, and/or heading and text positioning.

You may want to print the report before fine tuning it. If so, specify that you don't want to revise the report specification at this moment. If your report looks good, you will probably want to **D**escribe the report specification and save it with your system documentation. If you are not satisfied with the appearance of the report, you may **R**evise the report specification.

REVISING A REPORT SPECIFICATION

You may revise information that is contained in your report specification. Enter the revision process by one of the following ways:

1. Type an **R** (for the Revise function) when at the Report Writer menu.

2. After using the short method of creating a report specification, answer in the affirmative when asked if a revision is desired.

3. Supply the [R] option when entering Report Writer (detailed later in the section "USING REPORT WRITER WITHOUT GOING THROUGH THE MENU")

When revising a report specification, the Report Writer does not make reference to the model report form, so you cannot add or delete information on the form. If you make changes to the report form, you must re-create the report specification. You will also be unable to change an item's type (Text, Data, etc.), but this causes no problem since Report Writer can already tell which type an item is. If you desire to change an item's type, you must re-create the specification.

During the revision process, you will be shown information contained in your report specification, line by line. The cursor stops at the end of each line, called specification lines, so that you may make revisions. If the information is satisfactory, enter either <RETURN> or the down-cursor key (both provide the same function here). If you wish to revise information on the line, position the cursor over the information to be revised by using the back-cursor key, and revise the line by typing new information over the item(s) to be changed. Move to the next specification by pressing <RETURN> or the down-cursor key.

If you decide you want to move back up to the previous specification line, use the up-cursor key. When the report items are displayed and you enter the up-cursor key to step back to a previous item, a back slash (\) will appear before the previous specification line is displayed. If you are at the first specification line of a model line and press the up-cursor key, the Report Writer will step back to the start of specifications for the previous model line.

In order to allow you maximum flexibility during report specification revision, the Report Writer performs only minimal

checks for accuracy. It is possible for you to enter information during the revision process which will cause errors when you run the report. The most common error occurs when item starting positions are altered; this may result in the item values overlapping when a report is run.

The last screen during the revision process will display equations for Computed Items. You may change the equations at this time if you desire. After all specification have been displayed and revised, press any key to return to the Report Writer menu.

DESCRIBING A REPORT SPECIFICATION

You may get a description of the report specification at any time by selecting the **D** (Describe) option from the Report Writer menu. Report Writer asks if the output is to be sent to the printer:

Send output to printer (Y/N)?

If you answer **Y**, the description will be printed (as in the example in Appendix F). If you need to stop printing before the entire description has been printed, type **Control-C** (hold the CTRL key and press the C key). It will take a few seconds for the printer to stop printing information already obtained from the computer.

If you answer **N**, the report specification description will be displayed in the same manner as when you were **R**evising the report specification. However, since you cannot change information during the **D**escribe option, all information about each line is displayed on the screen at once. Press any key to see the next model line. After all model lines have been described, the equations for computed items are displayed. Press any key at this point to return to the Report Writer menu. (See Appendix F for a description of the WAREHSE report specification.)

PRINTING A REPORT

Once you have the report specification created and revised the way you want it, you may print a report by selecting the **P** (Print) option from the Report Writer menu. You are asked:

Select all records for report (Y/N)?

If you answer **N**, you will be asked for selection criteria:

Enter selection criteria or End <C/R>:

Answer in the form of a WHERE condition (e.g., PRICE<10.00). You will then be asked:

Send output to printer (Y/N)?

If you answer **N**, the report appears on the display. Since it's difficult to read the report as it scrolls up the screen, you may wish to stop the screen from scrolling by typing a **Control-S** (hold the CTRL key down and press the S key). To resume the display after a Control-S, type **Control-Q**. If during the process of listing the report to the screen, you wish to terminate printing, type a **Control-C**. You are then prompted to press any key to continue. Once you press a key, you are returned to the Report Writer menu.

If you answer **Y**, you will be asked:

Repeat first few lines for forms alignment (Y/N)?

This feature allows you to verify that the paper is properly aligned in the printer. Enter **Y**, if you want to check paper alignment. You are then asked for the number of copies of the report to be printed:

Enter number of copies to print (1-99) [1]:

The default in Report Writer is one copy. If you want multiple copies of the report to be printed, enter the number of copies desired. As is shown in the prompt, you may specify up to 99 copies.

If you had entered **Y** to the forms alignment question, the first few lines of the report will be printed, and you will be asked:

Forms alignment OK (Y/N)?

If you are not satisfied with the alignment of the paper, answer **N** to the question and realign the paper in the printer. The first few lines of the report will be printed again, and the question will be repeated. When you are satisfied with the forms alignment, answer **Y** to the question, and the rest of the report will be printed, from that point on.

USING REPORT WRITER WITHOUT GOING THROUGH THE MENU

Throughout this chapter, we have described how to use the Report Writer functions by going through the Report Writer menu. You may also run Report Writer without going through the menu by entering an option on the command line. The available options are:

[C]	Create new Report Specification
[D]	Describe existing Report Specification
[P]	Print a Report
[PF]	Print a Report, invoking forms alignment check
[R]	Revise existing Report Specification
[L]	List a Report on the Video Display

The Report Writer is invoked from the standard Condor prompt (e.g. **B>>** if B: is the current drive). Examples of accessing the Report Writer in this manner are as follows:

```
B>>REPORT WAREHSE [C]          (to create report spec)
B>>REPORT ORDERS [R]           (to revise report spec)
B>>REPORT GLJOURNAL [P]        (to print report)
```

Where WAREHSE, ORDERS, and GLJOURNAL are names of databases.

You can also specify selection criteria for printing when invoking the Report Writer in this manner. To do this, list the selection criteria before the print option. Examples:

```
B>>REPORT WAREHSE WHERE DIVNBR < 3 [P]
B>>REPORT PRODUCTS WHERE ALL [PF]
```

USING THE REPORT WRITER IN A COMMAND PROCEDURE

The Report Writer can also be invoked from a Condor command procedure, as outlined in Chapter 10. The command line will appear just as those which do not use the Report Writer menu, shown above. The same options are also used.

When **printing** a report from a command procedure, you will likely want it to run unattended. You may suppress the Report Writer prompts by specifying a WHERE condition and entering the **[P]** option on the command line. If you want all records to be selected, specify "WHERE ALL", "ALL", or "*". If you enter the **[PF]** option instead of the [P] option, the alignment question will be asked, and the command procedure will not run unattended.

Note also that the TITLE is cleared when the Report Writer is started. If you want the paper to go to the top of a new page before a report is printed, use the TITLE TOP command in the command procedure before the REPORT command.

REPORT WRITER SPECIFICATIONS AND HELPFUL HINTS

A. Maximum number of report specifications allowed.

The total number of individual specifications in a report is limited, depending on the number of data items in the database definition. Remember, each individual word or item in a model line becomes a specification. The following table shows how many specifications are allowed.

Number of Data Items in the Database File	Maximum Number of Report Specification Records
1 - 63	128
64 - 127	96

Because the total number of specifications is limited, you may want to combine separate words or phrases of titles and other text into a single specification, rather than have each word be an independent specification. To do this, place quotes around text in your report form. Text in quotes will be broken up into a series of 15-character specifications. This reduces the total number of specifications in the report and makes it easier to position the model line.

B. Length and width of paper report is to be printed on.

Normally, paper which is 8.5 inches wide by 11 inches long is used for printing a report, and a printer which prints 10 characters per inch of page width, and 6 lines per inch of page length is assumed. If you use this page size and printer, the default values in the Report Writer will work correctly.

The following table lists the default values:

Item of Information	Range	Default
Page Length (lines)	(1-127)	[66]
Left Margin (column)	(1-132)	[1]
Right Margin (column)	(1-127)	[80]
Top Margin (line)	(1-127)	[1]
Bottom Margin (line)	(1-127)	[60]

If you are using paper of a different size or a printer with different characteristics, you may want to replace the default values. Values for several common paper sizes and printer types are listed below. Use these values instead of the default values if you use these paper/printer combinations. If your paper/printer combination is not listed, find those that have similar characteristics to yours and combine the length and margin specifications to meet your needs.

Paper/Printer Details	Pg. Length	Rt. Margin	Bot. Margin
11-inch long paper, 8 lines/inch printer	88	80	80
8.5-inch wide paper, 12 char/inch printer	66	96	80
14-inch wide paper (132 column computer paper)	66	132	80
14-inch long (legal size) paper, 6 lines/inch printer	78	80	72

C. Reports wider than 80 columns.

A column width of 80 characters is assumed when using the short method of creating a report specification. For reports which are going to exceed 80 characters in width, **you must use the long method.**

To specify a report wider than 80 columns, set the right margin at least 80 characters beyond the left margin. Report Writer recognizes the difference between the margins and allows you to print with a width greater than 80 if this information is specified at the beginning of report specification **while using the long method.** (Note: If you do not specify the extended width at this time, you must re-create the report specification.)

On your report model form, you should either:

- Pack the fields and information into a single 80-column line on your report form, expanding them into their proper printing positions when you revise the report specification; or
- Use a second line, and let the items or text wrap around onto the second line.

When the report is specified to be greater than 80 columns, model lines in the report form automatically wrap around to the previous line. The wrap continues until the next model line fills the available space for column width.

Double backslashes ("\\") are useful in formatting your model form and report specification. When "\\" is encountered on a model line, the remainder of that line is ignored by Report Writer and text is stopped from wrapping. The next model line on the form is treated as a new report specification line.

D. Line Spacing/Positioning

When using the long method of creating a report specification, you are asked to specify the position and spacing for each line of the report. If the short method is used, you must revise the report specification to position and space the lines. Line positioning falls into three categories: 1) absolute line positioning, 2) relative line positioning, and 3) page lookahead.

1. Absolute line positioning

The prompts appear as follows:

Long Method:

Line number to skip to BEFORE printing line [0]:
Line number to skip to AFTER printing line [0]:

Short Method:

Line # Before	**Line # After**
0	**0**

Absolute line positioning allows you to specify the exact place on the page that a report line is printed. For instance, if you specify that Report Writer is to skip to line 3 before printing a heading line, that heading line is always printed on line 3 of a page. Absolute line positioning is useful for ensuring that page headings and similar material appear in the same position on all pages.

When using the long method, if you specify an absolute line position, you will not be prompted for relative line position and page lookahead for that line.

****
* **CAUTION** *
*
* If the report line is of the [EL] "every line" type, entering a number (other than 0) for an absolute postion causes Report Writer to skip to a new page for every record in the database. *
****

2. Relative line positioning

The prompts appear as follows:

Long Method:

Number of blank lines to insert BEFORE printing line [0]:
Number of blank lines to insert AFTER printing line [0]:

Short Method:

Space Before	Space After
0	0

Relative line positioning specifies the number of blank lines to appear before or after a line. This type of positioning is used most often with "before section", "after section", "page heading", and "last page" model lines.

3. Page lookahead

The prompts appear as follows:

Long Method:

Start new page if fewer than _____ lines remaining

Short Method:

Look Ahead
0

Page lookahead is a special feature that lets you prevent information being split across two pages, and also lets you begin each new section on a new page.

Use this function by specifying the number of lines that must remain on a page for the line to be printed. If fewer lines remain, Report Writer goes to the next page, prints page header information, and then prints the lines on the new page.

To start each new section on a new page, specify a number which is greater than 1 in the "before section" line.

To keep groups of "every line" information together on one page, use this feature in the first "every line" specification.

CHAPTER SUMMARY

* Report Writer allows you to specify the format and contents of database reports.

* Report Writer is accessed from the Condor prompt (B>>).

  - Options may be selected after the Report Writer menu is displayed.

  - Options may be specified in the **REPORT** command.

* A report is defined with two files:

  - a **REPORT MODEL FORM** file

  - a **REPORT SPECIFICATION** file

* A report model form is created in the same manner as in the **DEFINE** and **FORMAT** commands.

* **Short** and **Long** methods are available for defining a report specification. The long method queries you for more information and allows you to create reports which are wider than 80 columns.

* Report Writer can extract items from a database or calculate numbers from a given equation of items and/or constants.

* A report specification may be **R**evised or **D**escribed after it is created.

* Report Writer may be accessed using a command procedure.

CHAPTER 13

INDEXING YOUR DATABASE

In earlier chapters, you learned how to seek database records through functions like DISPLAY, DELETE, SELECT, and UPDATE. This is done by specifying a WHERE condition. You also learned to extract information from a database using LIST and PRINT commands. Before listing or printing, you sorted the database to organize its records in the proper sequence. In all of these cases, Condor finds its data by performing a sequential search (record-by-record). A sequential search is fine for small databases or those that don't change much, but in other cases, a database index is very helpful.

An index serves the same purpose for a database as a card catalog does for a library. Just as a card in the catalog is used to locate a book rapidly, an index is used to locate a record of information quickly. A card contains the title and location of a book and similarly, an index contains the "key" and location of a record. The "key" consists of the specific values of one or more data items in a record. Data items used to generate a key are specified when indexing a database.

There are numerous reasons that you might want to index a database:

1. You desire quick access to specified records.

2. You wish to see reports and listings sorted in key item sequence, and do not wish to re-sort the database file each time a report is produced.

3. You wish to place values automatically in order when entering into a database.

4. You desire to keep records in order when appending, reading, deleting, and/or updating records in your database.

5. You want to prevent duplicate items in a database.

This chapter will show you how to index a database, check for duplicate keys, and use Condor commands in conjunction with Condor indexing capabilities. The chapter will also describe how an index is updated and show you how to invalidate in index.

SETTING UP YOU CONDOR 20-3 DISCS.

You may have seen in Chapter 12 that Condor 20-3 consists of Condor 20-2 plus database indexing capabilities and an advanced yet simple-to-use report writer. The procedure for setting up your discs is the same as is outlined in Chapter12. Before you begin to use the Condor Index capabilities, you must "install" the 20-2 disc and 20-3 disc onto Work Discs that you will use in everyday operation. If you have a Winchester or 8" disc drive (HP 9895), you will be able to place all of Condor 20-3 onto one disc drive (if there is space left on the disc). Chapter 2 of this manual specifies how to install your 20-2 and 20-3 discs so that you may begin using them.

SETTING UP OR RE-CREATING A DATABASE INDEX

An index may be set up for a new database or for an existing database. You also have the ability to re-create an index.

1. Indexing a new database.

You may establish an index for a database when you first DEFINE the database. After you create a database in Condor 20-3, you are asked:

Do you wish to create an index for this Database (Y/N)?

To set up an index at this time, answer **Y**. Whether you answer **Y** or **N**, you are asked if you would like a printed copy of the database attribute summary:

Do you want a printed copy of the data definitions (Y/N)?

Answer **Y** to send the attribute summary to the printer. Answer **N** to display the summary only on the screen.

If you had earlier specified that you wish to create an index for this database, you are next prompted for the item(s) which will be index keys:

Enter name of data-item(s) which is to be the key:

Enter 1 to 8 database items that you would like to use as keys. Separate each item by a comma. The combined length of these items **must not exceed 127 characters.**

Next, you are asked whether you would like to permit duplicate keys to be entered into the database. (The concept of duplicate keys is explained later in this chapter.)

Do you wish to permit duplicate keys (Y/N)?

Answering **Y** allows duplicate keys to be added to the database.

2. Indexing an existing database.

You can also index an established database. Simply enter the **INDEX** command along with the database name and Key Data Items, and the data in the database will be indexed. Indexing an existing database with very few records is a fairly short process. It may take a little more time to index a database with a large amount of record, since each record must be reviewed and an index established for it.

The format of the INDEX command is as follows:

INDEX database USING d1,d2,...,d8 [option]

D1,d2,...,d8 are called **Key Data Items.** You may specify up to eight data items from your database to be used as keys. The combined length of all Key Data Items **must not exceed 127 characters.**

Available options ([option]) are:

[D]	Permit Duplicate Keys
[U]	Update Index

If you do not specify key data items or an option within the INDEX command, you are prompted for them with the following messages:

Enter name of data-item(s) which is to be the key:

Do you wish to permit duplicate keys (Y/N)?

3. Re-creating an index.

If database is already indexed you have the ability to re-create the index. You may wish to do this:

- Change key data items that are indexed, and/or
- Allow/disallow duplicate key entries

Use the INDEX command to re-create an index. The database must already exist and be indexed.

Example:

Type: **INDEX database**

Since the database is already indexed, Condor issues the following message:

Index exists already - Options: Destroy(D), Re-create(R) or End <C/R>:

Type: **R**

Condor asks for key data items:

Enter name of data-item(s) which is to be the key:

Enter up to 8 key data items. Finally, you are asked if duplicate entries are to be allowed:

Do you wish to permit duplicate keys (Y/N)?

DUPLICATE KEYS

In many cases, you do not want to allow duplicate keys in a database index. For example, a personnel database should probably not allow duplicate employee numbers. No two customer numbers should be the same in a customer address database. For these cases, you do not need to specify non-duplicate keys when setting up an index, as Condor defaults to that.

However, there are cases, when you want to permit duplicate keys. If you are indexing a purchase order database by VENDOR, you may have more than one purchase order with the same VENDOR. In order to tell the INDEX command to allow duplicate keys, specify the **[D]** option with the INDEX command, or answer **Y** to the duplicate key prompt when creating a database.

UPDATING AN INDEX

An index is automatically updated when you use ENTER, APPEND, READ, EMPTY, and DELETE commands with a database. It is also automatically modified when you use UPDATE to delete data in the database. **A data item used as a key cannot be altered by a CHANGE, COMPUTE, POST, or UPDATE command.**

Each time a database is used, the index is checked to make sure it is current with the data. If it is found not to be current, an error message is displayed, and the command is aborted. The **[U]** option may be used to update an existing index. It takes about the same amount of time to update an index as it does to create an index on the existing database.

INVALIDATING AN INDEX

If a database index is no longer needed, you may invalidate it in one of two ways:

- Delete the index file from the dictionary using the INDEX command.
- Re-sort or reorganize a database.

1. Deleting the index file.

A database index **(.IDX)** file may be deleted from the dictionary by using the INDEX command and choosing the option to destroy the index. The database must already exist and be indexed.

Example:

Type: **INDEX database**

Condor responds with the following prompt:

Index exists already - Options: Destroy(D), Re-create(R) or End <C/R>:

Type: **D**

Condor lets you know when it has finished destroying the index file by displaying:

Done.

In order to preserve the database, this is the only way that an index file should be destroyed.

2. Invalidating the index with SORT or REORG.

Use of the SORT or REORG commands also invalidates an index. By changing the sequence of the records in the database, the index is no longer accurate. To reconstruct the index, use the INDEX command with the [U] option.

COMMANDS THAT AFFECT INDEXED DATABASES

Since indexes are used primarily to provide fast access to the data in a database, the commands on which you can specify a WHERE condition (DISPLAY, DELETE, SELECT, and UPDATE) check to see if the database is indexed, and, if it is, whether the index can be used to make record access faster. This is done by determining if Key Items are used in the WHERE condition and there are no OR clauses.

The DELETE and SELECT commands impose an additional restriction on index usage. In order for these commands to take advantage of indexed access, the WHERE condition must specify a particular value (IS or = condition) or a limited range (surrounded by both GT and LT conditions).

The LIST and PRINT commands will also utilize an index. If a database is indexed, the information can be shown in key sequence by specifying the [I] (Index) option within the LIST or PRINT command.

CHAPTER SUMMARY

* The **INDEX** command allows you to index a database for:

  - o Quick record access
  - o Listing and printing of a database in key item sequence without re-sorting
  - o Preventing duplicate items

* Data item names in a database are used as keys.

* New or existing databases may be indexed.

* Duplicate keys may be allowed by using the [D] option in the INDEX command.

* An index may be invalidated by using the INDEX command or by re-sorting or reorganizing the database.

* A database index is automatically updated with many Condor commands. The [U] option allows you to update an index at any time.

GLOSSARY OF TERMS

Alphabetic Data	Characters a to z, A to Z, "-", "'"
Boot-up	The HP Series 100 startup procedure. At boot-up, the CP/M operating system is loaded and the WELCOME Menu is displayed.
Command Procedure	A set of Series 20 commands, stored in a disc file, which can all be executed in sequence by a single command.
Computed Items	Items not contained in the database but computed by Report Writer in 20-3. A computational formula is user supplied.
Control Items	In the Condor 20-3 Report Writer, a data item which indicates that a new section is to begin with a change in value. More than one control item may be specified.
Data Base	A database is an organized collection of related information that can be retrieved and manipulated on demand. A database is stored as a collection of disc files describing the screen form, data items, data item definitions, and data.
Database1	A term used in Condor Series 20 command line syntax, indicating the "destination" database.

Database2	A term used in Condor Series 20 command line syntax, indicating the "source" database.
Data Definition	The attributes of a data item (i.e. type of data, length in bytes, minimum and maximum values, and default values). Data definitions are stored in a disc file and are associated with the database dynamically.
Data Dictionary	A Condor Series 20 database file which describes the databases stored on a disc. The records of the Data Directory describe the files associated with each database defined.
Data Directory	The header record of **.DAT** files. The header contains the number of data items, bytes per record, and number of records in the database.
Data File	A disc file containing the data values for a database.
Data Item	The smallest unit of information accessible in a database. Examples are CITY, AGE and NAME. In a Condor 20-3 report specification, a data item signifies to be taken from the database.
Data Item Name	Each data item in a database has an associated data item name that is used to identify it. The name designates a fixed amount of storage within each database record used to store the data item's value. Data item names are included in the database form and are described by data definitions.
Data Item Size	The number of bytes of disc storage allocated to a data item. Also called Field Size.
Data Item Type	Specification in a data item definition indicating whether the data to be stored in a data item is Alphabetic (A type), Alphabetic-Numeric (AN type), Numeric (N type), Dollar ($ type), or Date (J type, where J indicates a Julian Date).

Data Value	Specific alphabetic characters, numbers or special characters ("+", "-", ".", etc.) stored in the allocated disc space of a data item. Examples: New York, 59.56, John Doe, etc.
Default Value	Predefined data values, which will be entered in a data field while entering data if no other value is entered.
Edit Criteria	Predefined restrictions for entering data such as minimum/maximum values, default values and data types. Condor Series 20 prohibits entry of data not meeting edit requirements and signals the user if errors occur.
Field	A specific amount of space allocated within a record.
File	Allocated space on a disc to be used for storage of data, data definitions, forms, command procedures, etc. A data base file is a collection of data base records.
Form	The screen display format into which database data is entered, and in which database data is displayed.
Format	The process of "painting" a screen form on the HP Series 100 screen for later data entry use.
Input	Refers to the process of entering data into a database, or entering data in a file; or describes a set of data which is to be or has been entered into the computer.
Julian	A numeric representation of a date. Dates are entered into Series 20 and displayed in the form MM/DD/YY. Internally, the date is stored in a Julian representation where January 1, 1900 (01/01/00) is day 1 Julian, and each consecutive date is represented as the next consecutive number.

Master/Transaction	Database qualifiers (master database, transaction database) used within this manual to clarify transaction processing. The master database is updated by a transaction database. For example, journal transactions update a general ledger.
Matching Field	Synonymous with "Matching Data Item". Refers to data item naming and data item definition requirements. A data item in one database must be defined with the same name and data definitions as in another database.
Model Lines	A line created by the user in setting up a report format. Each report may have up to 24 model lines. Condor 20-3 Report Writer uses model lines to generate default values and determine which items are to appear.
Record	A collection of data items. The contents of a record refers to the collection of data values stored in a record.
Report Form	The .FRM file that contains model lines used to create a report specification, using the 20-3 Report Writer.
Report Specification	An .RPT file that contains all the information needed to describe or print a report. Created by using the Condor 20-3 Report Writer. Individual line specifications are derived from model lines.
Section	In the Condor 20-3 Report Writer, a group of related information in a report.
Section Change	In the Condor 20-3 Report Writer, caused when the value of a control item changes.
System Values	Date ($DATE) and page number ($PAGE). Used in the Condor 20-3 Report Writer
Text	Titles, column headings, labels, and descriptive information.

APPENDIX A

CONDOR SERIES 20 COMMAND DESCRIPTIONS

SERIES 20 COMMAND DEFINITIONS

TABLE OF CONTENTS

PAGE

SUMMARY OF COMMANDS A-2
COMMAND DESCRIPTION AND USAGE A-3
COMMAND SYNTAX A-5
ABORT A-7
APPEND A-8
* CHANGE A-9
* COMBINE A-10
COMPARE A-11
COMPUTE A-12
COPY A-15
DATE A-16
DBMS A-17
DEFINE A-18
* DELETE A-20
DESTROY A-21
DIC A-22
DIR A-23
* DISPLAY A-24
EDIT A-26
EMPTY A-27
ENTER A-28
FORMAT A-30
HELP A-32
** INDEX A-32.1
* JOIN A-33
LIST A-34
LOG A-36
POST A-37
PRINT A-39
* PROJECT A-41
READ A-42
RENAME A-43
REORG A-44
** REPORT A-44.1
RESTART A-45
RUN A-46
SAVE A-49
SELECT A-50
SET A-52
SORT A-54
STAX A-55
SYSTEM A-56
TABULATE A-57
TITLE A-59
UPDATE A-60
WRITE A-61

(*) These commands are included only in Condor 20-2 & 20-3
(**) These commands are included only in Condor 20-3

	COMMAND	DESCRIPTION
	ABORT	Stops a RUN command
	APPEND	Attach the records of database2 to database1
*	**CHANGE**	Change data item values in a database
*	**COMBINE**	Attach records of the databases, creating a RESULT database
	COMPARE	Compare data item values in two databases for (not) matching conditions and create a RESULT database
	COMPUTE	Compute data item values in a database
	COPY	Copy a database or file
	DATE	View or enter date
	DBMS	Loads CONDOR SERIES 20 system
	DEFINE	Create a new database, redefine a database, describe a database
*	**DELETE**	Delete records of a database meeting specified conditions
	DESTROY	Eliminate a database or file
	DIC	View entries in the data dictionary
	DIR	View the list of files in the data directory
*	**DISPLAY**	View selected records of a database
	EDIT	Built in text editor to create RUN files
	EMPTY	Eliminate all records in a database
	ENTER	Insert new records into a database
	FORMAT	Create or revise a database (or report) form or a HELP screen
	HELP	Assist operator in selecting procedures
**	**INDEX**	Create or rebuild a database index
*	**JOIN**	Attach data items of two databases by matching data item values creating a RESULT database
	LIST	View records of a database in sequential or indexed order
	LOG	Log a new disk in the computer
	POST	Update data item values in one database with those from another
	PRINT	Print records of a database in sequential or indexed order
*	**PROJECT**	Create a RESULT database from selected data items of a database
	READ	Transfer records from an ASCII file to an existing database
	RENAME	Change the name of a database or file
	REORG	Reorganize the structure of a database, adding or deleting data items
**	**REPORT**	Create, revise, print a report
	RESTART	Continue processing of an interrupted command procedure
	RUN	Start processing a command procedure having directives
	SAVE	Save a RESULT database
	SELECT	Select database records meeting specified conditions, creating a RESULT database
	SET	Set DBMS operating parameters
	SORT	Sort database records by data item values
	STAX	View or print statistics of data item values
	SYSTEM	Exit from Condor
	TABULATE	Summarize and print specified data items
	TITLE	Print report headings
	UPDATE	Change data item values in a database meeting specified conditions
	WRITE	Transfer records from a database to an ASCII sequential file

(*) These commands are included only in Condor 20-2.
(**) These commands are included only in Condor 20-3.

COMMAND DESCRIPTION AND USAGE

The database manager commands require that the database management system be in memory. It can be loaded by typing **DBMS** or pressing the **CONDOR DBMS** function key (if installed). The user should always record the current date at startup time, (see **DATE** command).

The operator is always prompted on the console screen with the characters x>>, where x is a disk drive letter (A - H) which the user has defined as the current disk drive. For example, if the prompt is A >>, the current disk drive is A. To select another disk drive as the current disk drive, type the disk drive letter, a colon, and carriage return, E.G., **B:< C/R >**. Typically the user may select another disk drive where more disk space is available. Commands may be entered without specifying the disk drive on which they reside (see **SET** command).

The following general syntactical rules apply to command statements.

1. { } - (braces) indicate a character, word, or phrase is optional. The braces are not entered.

2. [ ] - (brackets) indicate a character or word is optional and that when the option is used, brackets must also be entered.

3. ( ) - (parenthesis) indicate that the enclosed text is for clarification and is never entered.

4. / - (slash) indicates that only one of the items listed should be entered. Exceptions are when the slash is used in a date or as the division symbol in the **COMPUTE** command.

5. **UPPER CASE BOLD** is entered exactly as shown.

6. **Lower case bold** represents variable names for which the user substitutes in actual usage.

7. Where applicable, commands can address up to 32 operands. Operands are data item names and constants. Data item names are denoted by d1, d2 ...,dn. Delimiters, logical or arithmetic operators, command names, or filenames do not count as operands. For example, A 32 data item sort can be initiated by the command **SORT CLIENTS BY NAME, CITY, . . . d32.**

8. Data item names must be referenced as they appear in the brackets of the Form File (**.FRM**). They may be abbreviated by the * or ? symbols. However, when using the **COMPUTE** command, they must be spelled out in full to avoid confusion with the arithmatic operators.

9. Operands must be less than 16 characters in length when defined in the command line. Operands with embedded blanks (e.g., **JOHN SMITH**) must be enclosed in single or double quotes in order to be interpreted as one operand (e.g., **'JOHN SMITH'**).This rule does not apply to data entered using the **ENTER** or **UPDATE** commands. Database Titles must be referenced by no more than 8 characters.

10. All command lines are limited to 127 characters.

11. Upper and lower case letters are always treated the same for search and compare operations. Hence, they may be entered either way. They are displayed as entered.

12. Work or RESULT sets are created under the name **R$.ext** (**R$.DAT, R$.DBM, R$.DEF, R$.FRM, R$.ASC**). They will always be created on the current drive. A blank disk can be inserted into an unused drive and selected as the current drive if more work space is desired. These files can be renamed if they are to be saved for later use. In command procedures, filenames of the format **$$.ext** must be typed as **$$$$.ext** so that the "$" symbol will not be interpreted as a batch parameter.

13. The system files, **DEF.CVL, DEF.DEF, DEF.FRM, DIC.DEF, DIC.FRM, DIR.DEF, DIR.DIR** and **DIR.FRM** must always reside on drive A. The system will only search drive A for these files regardless of the current drive or the master drive.

CONDOR SERIES 20 COMMAND SYNTAX

Command statements consist of a verb-like command, database or file name, and in some commands, additional qualifying information, and an optional switch. The qualifying information describes database or data items and logical/arithmetic operators for adding, comparing, asssigning item, etc. The snytax for the command statement is as follows:

Syntax:	COMMAND {Dr:}Database/File {Qualifying Information} [option]
	[ ] indicates optional; /indicates alternate choice
Example:	**UPDATE B:EMPLOYEE WHERE NAME IS W.B.SMITH**
COMMAND:	One of the CONDOR SERIES 20 instructions described in Appendix A of this manual.
[Dr:]	Indicates optional drive designation. In the example, database EMPLOYEE resides on drive B.
Database/File	The database name of file name.ext (.ext is a three letter file name extension used to describe the type of file). Database refers to a CONDOR SERIES 20 title. Each database is composed of a **.FRM** file (screen format), a **.DEF** file (data item definitions), and **.DAT** file (data). Database names can not exceed eight characters. Filename.ext describes a single disk file. The name should not exceed eight characters in length and the .ext three characters. The extensions of **.DAT, .DBM, .DIR, .FRM, .DEF, .DIC, .DIR, .HLP, .SUB,** and **.CMD,** are standardized and will be described in other sections of this manual.
Qualifying Information	A user specified list of data item names, constants and operators which varies according to the CONDOR SERIES 20 COMMAND used. Options are described later in Appendix A within the COMMAND description. CONDOR SERIES 20 wild characters are extremenly helpful in expressions. The wild characters "*" allows you to shorten a data item name such as ADDRESS to ADDR* in a command line. The wild character "?" instructs the computer to accept any value in the data item's corresponding position. For example, if a data item is always stored with a person's first and second initials followed by a last name (e.g. W.B.SMITH), then the instruction "...WHERE NAME IF ????SMI* instructs the computer to search for all last names beginning with SMI. * and ? can be used for any data item name or any Alpha or Alphanumeric data value. There are some restrictions indicated in Appendix A.
[option]	A user specified **option** switch which alters the operation of an DBMS command. For example, in the **SORT** command, the switch [D] instructs the **SORT** program to sort the database in descending sequence. CONDOR SERIES 20 command options are described later in Appendix A. Note, the option character must be capitalized.

	COMMAND	SYNTAX / DESCRIPTION
	ABORT	**ABORT**
	APPEND	**APPEND** database1 database2
*	**CHANGE**	**CHANGE** database {ST d1 is value1 AND d2 is value2...}
*	**COMBINE**	**COMBINE** database1 database2
	COMPARE	**COMPARE** database1 database2 {NOT} MATCHING d1, d2,...,d32
	COMPUTE	**COMPUTE** database ST dr = d1 opr d2...d31
	COPY	**COPY** destination-database/file = source-database/file {OK} **COPY** drive: = source-database/file {OK}
	DATE	**DATE** {mm/dd/yy}
	DBMS	**DBMS**
	DEFINE	**DEFINE** database {a.dat,b.frm,c.def (,d.rpt,e.idx)}
*	**DELETE**	**DELETE** database {WHERE condition}
	DESTROY	**DESTROY** database/file
	DIC	**DIC** {drive:}
	DIR	**DIR** {drive:} {filename.ext}
*	**DISPLAY**	**DISPLAY** database {WHERE condition}
	EDIT	**EDIT** filename {.CMD}
	EMPTY	**EMPTY** database {OK}
	ENTER	**ENTER** database {[option]}
	FORMAT	**FORMAT** formfile {.FRM/.HLP}
	HELP	**HELP** helpfile {.HLP}
**	**INDEX**	**INDEX** database USING d1,d2,...d8 {[option]}
*	**JOIN**	**JOIN** database1 database2 MATCHING d1,d2,...,d32 {[option]}
	LIST	**LIST** database BY d1,d2,... {AND COMPUTE statistics d3,d4,...}} {[option]}
	LOG	**LOG** {drive: {diskname} }
	POST	**POST** database1 database2 MATCHING d1,d2,... {opr d3,d4...}
	PRINT	**PRINT** database {BY d1,d2,... {AND COMPUTE statistic d3,d4,...}} {[option]}
*	**PROJECT**	**PROJECT** database by d1,d2,...,d32
	READ	**READ** database {filename.ASC}
	RENAME	**RENAME** newname oldname {OK}
	REORG	**REORG** database
**	**REPORT**	**REPORT** database {WHERE condition} {[option]}
	RESTART	**RESTART** drive:
	RUN	**RUN** procedure {.CMD} {param1 param2 ... param9}
	SAVE	**SAVE** database/file {OK}
	SELECT	**SELECT** database {WHERE condition}
	SET	**SET** {function status}
	SORT	**SORT** database BY d1,d2,...,d32
	STAX	**STAX** database BY d1,d2,...,d32 {[option]}
	SYSTEM	**SYSTEM** or **SYS**
	TABULATE	**TABULATE** database {BY d1,d2,... {AND COMPUTE statistics d3,d4,...}} {[option]}
	TITLE	**TITLE** {'text' or "text",PAGE,DATE,SKIP,LINE,TOP,nn}
	UPDATE	**UPDATE** database {WHERE condition}
	WRITE	**WRITE** database {filename.ext}

(*) These commands are included only in Condor 20-2.

(**) These commands are included only in Condor 20-3.

ABORT

PURPOSE: Stops a RUN command procedure

SYNTAX: **ABORT**

EXAMPLE: **ABORT**

USAGE:

This command may be used to exit from applications by placing it as one of the selection in a HELP screen.

APPEND

PURPOSE: Attach records of one database to another

SYNTAX: **APPEND database1/file1 database2/file2**

Where the records of **database2/file2** are appended to the records of **database1/file1. Database2/file2** is unchanged.

EXAMPLES:
```
APPEND  B:PARENT  C:DAILYINT
APPEND  B:DEFSET.DEF  B:PARTDEF.DEF
APPEND  B:CLIENTS.DAT  R$.DAT
```

USAGE:

The records of database2 are appended to the records of database1. Database2 is unchanged.

This command requires that both files have identical record sizes and field numbers as shown in their respective file directories. Otherwise, the append will be aborted.

Only files with extensions of .DEF, .DIC and DAT may be appended. If no extension is specified, the name is considered to be a database title and the appropriate data file is read from the dictionary.

When the appended records cannot fit onto one diskette (i.e., no space left), an error message will be issued, and the **APPEND** will terminate, resulting in no records appended to database1/file1.

The drive must always be specified if not the current drive.

CHANGE

PURPOSE: Change data item values in a database

SYNTAX: **CHANGE database {ST d1 IS value1 AND d2 IS value2...}**

Where value1, value2,... can be constants or other data item values

EXAMPLES:
CHANGE CLIENTS ST DATE = 1/1/79
CHANGE CLIENTS ST STATUS IS 1 AND DATE IS 06/01/81
CHANGE ORDERS ST COMPANY IS 'CONDOR CORP'
CHANGE CLIENTS ST STATUS2 = @STATUS1

USAGE:

This command changes the value of one or more data-items in each record of a database. The data-item value on the left side of the verb is changed to the new value on the right side of the verb. The verb can be the word **IS** or the equality symbol (=). The verb must be preceeded and followed by one or more spaces.

Data-items can be replaced with other data-item values within the same record. This requires that the data-item on the right side of the verb be preceeded with the commercial at symbol (@), which distinguishes the data-item from a constant. For example, if **STATUS1** and **STATUS2** are data-items, the command:

CHANGE CLIENTS ST STATUS2 IS @STATUS1

will assign the value of **STATUS1** to the value of **STATUS2** in each record of the database **CLIENTS.**

Note: In order to perform this change, the data definitions of **STATUS1** and **STATUS2** must be identical (i.e., both must have the same field size and type).

Several data values may be changed at once by specifying a list of data items and values separated by the delimiter AND. A maximum of 16 data values may be changed at one time.

Caution should be exercized using this command because no edit checking is made for appropriate upper and lower bounds on the new value. For example, if the data-item **COMPANY** is a 4 byte alphanumeric field and the change specified is to set **COMPANY IS CONDOR,** only 4 characters **COND** will be stored and no error message provided.

COMBINE

Purpose: Attach records of two databases, creating a RESULT database

SYNTAX: **COMBINE database1/file1 database2/file2**

RESULT SET: **R$.DAT, R$.DEF, R$.FRM**

EXAMPLES: **COMBINE B:CLIENTS C:CLIENTS**
COMBINE B:CLIENTS.DAT C:CLIENTS.DAT

USAGE:

This command produces a union of two sets, i.e., a new set is created containing records from both **database1** and **database2.** The form and definition files from **database1** are copied as part of the result set. Both databases must have identical record types (i.e., both must have the same data definitions and in the same order) for combining, otherwise an error message will be displayed and the program aborted.

Individual files that have extensions of **.DEF, .DIC,** and **.DAT** may also be combined, in which case no form or definition files are created. In general, if there are i active records in **database1** and j active records in **database2,** there will be (i+j) records in the RESULT set. This command will not transfer the inactive (deleted) records from the two databases to the RESULT set.

If directory is full or disk space not available, an error message is displayed and the **COMBINE** is terminated with no result set written.

COMPARE

PURPOSE: Compare data item values in two databases for (not) matching conditions and create a RESULT database

SYNTAX: **COMPARE database1 database2 { NOT } MATCHING d1,d2,..., d32**

RESULT SET: **R$.DAT, R$.DEF, R$.FRM**

EXAMPLES:
```
COMPARE ORDERS TRANSACT MATCHING ACCOUNT
COMPARE ORDERS SHIPPING USING ACCOUNT AND SHIPPER
COMPARE ORDERS OLDORDER NOT MATCHING ACCOUNT
COMPARE ORDERS OLDORDER ACCOUNT
```

USAGE:

The **COMPARE** command compares specified data items in database1 to the corresponding data items in database2. **USING** and **BY** are synonyms for **MATCHING.**

The MATCHING condition produces a RESULT set of database1 records whose data item values are also contained in database2. The **NOT MATCHING** condition produces a RESULT set of database1 records whose data item values are **not** contained in database2.

In example 1,records from **ORDERS** with **ACCOUNTS MATCHING** those in **TRANSACT** are chosen and written to the RESULT set. In example 3, records from **ORDERS** with **ACCOUNTS NOT MATCHING** those in **OLDORDER** are chosen.

Note: Data items being compared in the two data bases, must be defined as having the same name, size, and field type.

EXAMPLE: **COMPARE TRANSACT ACCOUNT MATCHING ACCTNO**

would write to the RESULT set only those transactions having a valid account number.

By using the **NOT MATCHING** condition, records requiring review would be written to the **RESULT** set.

Note: COMPARE ACCOUNTS TRANSACT USING ACCTNO is not the same as **COMPARE TRANSACT ACCOUNTS USING ACCTNO.**

In the first statement **ACCOUNTS** records **MATCHING** those in **TRANSACT** are written to the RESULT set. In the second statement, **TRANSACT** records **MATCHING** those in **ACCOUNT** will be written to the RESULT set.

COMPUTE

PURPOSE: Compute data item values in a database

SYNTAX: **COMPUTE database ST dr = d1 opr d2... d31**
COMPUTE database ST d1 opr d2... d31 = dr

Where d1, d2,... d31 are operands which can be integers (constants) or data items (variables) and dr is the data item name for the computed result. There may be up to 32 terms including the result data item.

opr are the following arithmetic operators:

+ Addition
- Subtraction
* Multiplication
/ Division

EXAMPLES:
```
COMPUTE  ORDERS ST SUBTOTAL=QUANTITY*PRICE
COMPUTE  ORDERS ST PROFIT=PRICE-COST
COMPUTE  ORDERS ST DISCOUNT=PRICE/4+12.00
```

USAGE:

The **COMPUTE** command is used to calculate a quantity, and store the result in a numeric field for each record in the data base. Using this command properly requires a careful understanding of the syntax; otherwise, unexpected results may be obtained.

Caution must be exercised in using this command because of its global effects.

Order of evaluation:

The result data item must be already defined in the data base record. The terms in the expression are evaluated from left to right as they appear in the equation. **There is no priority given to operands or items in parentheses.** Parentheses and spaces are optional and ignored, however they may be used for making the expression more readable. For example, the expressions below are identical:

```
PRICE = 7 - 4 * AMOUNT + 2          (1)
PRICE = ( 7 - 4 ) * AMOUNT + 2      (2)
PRICE = 7 - ( 4 * AMOUNT ) + 2      (3)
PRICE = 7 - ( 4 * AMOUNT + 2 )      (4)
PRICE = 7 - 4 * ( AMOUNT + 2 )      (5)
```

The above expressions are all evaluated identically as illustrated in (2), i.e., from left to right.

Identification of terms:

The terms in the expression are identified by their first character. If it is a digit (0-9), the term is considered a constant and will be interpreted as an integer. If the first character is a letter (A-Z), the term is considered a data item and the contents interpreted as an integer. Since the multiplication sign (*) is also a wild character symbol, data item names are required to be spelled out as they appear in the form of the database.

If the data item name has embedded blanks or any of the characters (+,=,/,-), the name must be enclosed in quotation marks (") or literals (') so it can be treated as a single name. For example, the item name **ENROLLED-DATE** must be entered as **'ENROLLED-DATE'** or **"ENROLLED-DATE"**, otherwise it would be treated as two terms, **ENROLLED** minus **DATE.** A minimum of three operands and two operators (including the equals sign) is considered a valid equation.

Data item types:

The data items referenced in the expression must be numeric, dollar, or date. Alphabetic and Alphanumeric data items cannot be used in the **COMPUTE** statement.

Numeric conversion and overflow:

Arithmetic is performed as integer arithmetic and fractions from division are truncated.

EXAMPLE: **9 / 10 = 0**

All operands are normalized to a four byte integer value and then operated upon by the operators. This means that data items having different sizes may be used in the arithmetic. For example a number in a two byte item may multiply a number in a 1 byte item. However, there are restrictions in the multiplication and division operations. These operations must not produce intermediate results greater than four bytes. For example in the expression:

A*B/C=D

A*B must not exceed four bytes, even though after dividing by **C** it may produce a result that fits in four bytes. If these restrictions are violated, an overflow error will occur and that computation will not be performed.
If the result of the computation is a number greater than the defined result item size, an overflow error message will be listed, and the result will not be stored.

Date (Julian) data items:

Two julian data items may be subtracted giving the difference between the dates in days.

EXAMPLE: **COMPUTE TIME ST DAYS = FINISH - START**

A constant number of days may be added or subtracted from a julian data item, yielding a new date.

EXAMPLE: **COMPUTE TIME ST FINISH = START + 30**

A date format cannot be entered as a constant, since all constants are assumed to be integers.

> 1/1/80 would be interpreted to mean 1 divided by 1 divided by 80. Since the date is stored in Julian, 010180 would also not result in the data 1/1/80 being used in the calculation.

Dollar ($) data items

Dollar items are stored as 100 times the dollar value (i.e. in cents). The user must use care to remember this when using the **COMPUTE** command.

Dollar constants must be entered with two decimal places. For example, $123.00 would be entered as 123.00 not 123.

The examples below illustrate use of dollar and numeric data items yielding correct results:

```
$ = $ + $
$ = $ - $
$ = $ * $ / 100
$ = $ / $ * 100
$ = $ * n
$ = $ / n
n = $ * $ / 10000        (May frequently cause overflow)
n = $ / $
```

The **COMPUTE** command can also be used in an interactive mode by issuing the command **COMPUTE database.** The equation can be entered after the prompt, the computations will be performed, and the results stored. When completed, the operator will be prompted for another equation.

COPY

PURPOSE: Copy a database or file.

SYNTAX: **COPY destination-database/file = source-database/file { OK }**
COPY drive: = database/file { OK }

EXAMPLES: **COPY B:RESULT=C:RESULT**
COPY B:=C:RESULT
COPY B: NEW.DAT C:CLIENTS.DAT
COPY DUPLCATE.DAT=CLIENTS.DAT

USAGE:

This command is used for copying databases and individual files from one drive to another. It may also be used to make duplicate copies of a database or file on the same drive under a different name. A filename is distinguished from a database name by the presence of an extension name (e.g., **CLIENT.DAT**). If the source drive has no data dictionary file (**DATA.DIC**) then all names in the command are interpreted as filenames.

The form, definition and data files of a database are copied when the title of a database is referenced in the command. If the destination - name is the same as the source-name, then all files are named as shown in the source dictionary. If no destination-name is specified (e.g., **COPY B:=C:RESULT**), the copy destination-name is assumed to be the same as the source-name. When the source and destination names are different, the destination files are all renamed with the destination name, except with the appropriate extensions supplied. If the destination drive has a data dictionary, a new dictionary record will be written corresponding to the filenames and title of the copied database. If no data dictionary exists on the destination copied drive, a data dictionary (DATA.DIC) will be created and a new dictionary entry written.

Database or filenames that exist already on the destination drive will not be erased unless a confirmation is given by the user. The confirmation may be given in the command line for batch operations by typing **OK** following the command line.

Wildcard characters (*) and (?) cannot be used with the **COPY** command.

DATE

PURPOSE: View or enter date.

SYNTAX: **DATE {current-date}**

EXAMPLES:
```
DATE
DATE 4/1/79
DATE 4-1-79
DATE 040179
```

USAGE:

Following the command **DATE**, the current date, if any, is displayed and then a prompt for a new date will occur. A <C/R> will leave the date unchanged. If the current date is specified in the command line then it is recorded as the current date with no prompting. Dates may be entered in the formats **mm/dd/yy**, **mm-dd-yy**, or **mmddyy** and will be stored in Julian as a 3 byte binary number.

Note: Leading zeros in the month and day need not be typed except in the mmddyy format.

New files created during the session will have the current date stored in their directory. The date of creation of a database can be displayed by listing the directory of the file, (e.g., **LIST CLIENTS.DIR**)

See the **SET** command for additional **DATE** options.

DBMS

PURPOSE: Loads CONDOR SERIES 20 system.

SYNTAX: **DBMS**

EXAMPLE: **DBMS**

USAGE:

The **DBMS** is the main body of the SERIES 20 system. It must be in memory in order for all the SERIES 20 commands to operate. Typing **DBMS** or the **CONDOR DBMS** function key (if installed) starts the system.

One feature of SERIES 20 is that many other non-SERIES 20 programs (utilities or application programs) may be executed without returning to CP/M. These programs must have an extension of .COM. They are executed by typing **$program** (eg $PIP). Once DBMS is in memory there is 32K of user space in which to load programs and data. If the utility or application program is greater than 32K, the user can return to CP/M by executing the **SYS** command, and restore the full user memory space CP/M allows. If the user needs to use the SERIES 20 commands, DBMS must be loaded again.

DEFINE

PURPOSE: Creates a new database, redefines a database, describes a database.

SYNTAX: **DEFINE database {A.DAT, B.FRM, C.DEF (,D.RPT, E.IDX)}**

EXAMPLES: **DEFINE CLIENTS**
DEFINE INVENTRY
DEFINE QUICKDB EXISTS.DAT, EXISTS.FRM, EXISTS.DEF

USAGE:

The **DEFINE** command creates a database by creating a data definition file **(.DEF)**, a data file **(.DAT)** and writing a new record to the data dictionary **(DATA.DIC)**. The format used for creating a database is defined by a form file **(.FRM)**.

EXAMPLE: **DEFINE B:CLIENTS**

If the command above is entered, the **DEFINE** command searches drive B for a file called **CLIENTS.FRM**. If the form file is not found, then two prompts were made: The first prompt asks if a new form is to be created. Typing **Y** will automatically load the **FORMAT** program. If **N** is entered, then the name of an existing form file will be requested. The extension **.FRM** need not be supplied as it is assumed. The second prompt asks if a new definition file is to be created. If **Y** is entered, then data definitions must be supplied as described below. If **N** is entered, then the name of an existing definition file must be entered. The extension **.DEF** is not required, as it is assumed. The form and definition data item counts must be the same, or an error message is displayed.

A database can be defined quickly by using existent .DAT, .FRM and/or **.DEF** files (most likely attached to another database). After specifying the database to be created, list the files to be attached to this new database:

DEFINE NEWDB OLDDB.FRM, OLDDB.DEF.

The definitions are supplied following the prompt of a data item name. Data item names are displayed one at a time and a data type must be entered.

Valid data types are:

A	Alphabetic
AN	Alphanumeric
N	Numeric
$	Dollar
J	Julian
R	Required - used with any of the above.

Following entry of the data type a carriage return may be entered and the system will supply the remainder of the definition. Following entry of the last data item type, the **{{End}}** is displayed and a prompt asking if the definitions are OK. Entering **Y** saves the definitions. Entering **N** permits editing the definitions. If **N** is entered, the definitions will be displayed

for each data item as entered. If the definition is satisfactory, enter a carriage return to review the definition. If the definition is not satisfactory, move the cursor using the left arrow key to locate the cursor over the value to be edited, and type over it with the new value. When finished editing, enter a carriage return and the next definition will be displayed. Continue to edit until the **{{End}}** is reached and the **"Definitions OK (Y/N)"** will be displayed.

Reply **Y** to save the definitions. After the definitions are saved, in 20-3, a prompt asks if an index is desired for the database. Another prompt is displayed asking if a description of the database is desired. Enter **Y** for a description. A further prompt asks if the descriptions are to be directed to the printer (**Y**) or console (**N**).

An empty database can be redefined. To redefine a database containing records, use the **WRITE** command to save the data before redefining, and the **READ** command to restore the data after redefining. If you wish to add or delete fields with a database having records but not change the field definition, use the **REORG** command.

Redefine a database by entering the **DEFINE** command and selecting the Redefine option. In the redefine mode, the previous definitions will be displayed. They may be edited as described above. In order to edit a specific data item enter a **#**. **A>** prompt will be displayed. Enter the data item name to be changed and a carriage return or another **#** to end the edit mode.

Once a database has been defined, the user can update the edit criteria at any time. If there are no records in the database, then the redefine option can be used, otherwise the **UPDATE** command must be used on the definition file.

EXAMPLE: **UPDATE CLIENTS.DEF WHERE FIELD IS 3 4 5**

CAUTION: Only the MIN, MAX, and DEFAULT values should be updated.

The database definitions may be displayed anytime by choosing the describe (D) option. The user will be prompted to send the result to the printer (Y) or console (N). A database need not be empty to display the data definitions.

DELETE

PURPOSE: Delete records of a database meeting specified conditions.

SYNTAX: **DELETE DATABASE {WHERE condition}**

EXAMPLES: **DELETE CLIENTS WHERE DATE IS LT 1/1/79**
DELETE CLIENTS ST DATE IS GT 5/1/78
DELETE CLIENTS ST ACTIVITY IS TERMINATED
DELETE EMPLOYEE WHERE ALL

WHERE condition:

A WHERE condition is one or more comparisons joined by AND or OR. Both AND and OR may not be used together in one command. A comparison is a phrase comparing a data item with one or more values:

	data-item	comparison-verb	value {,value...}
EXAMPLE:	**STATE**	**IS**	**MI,IN**

A comparison-verb compares the value of the data item with one or more values to determine if the WHERE condition will be satisfied.

Comparison-verbs are:

Verb	Synonym
=	EQ, IS, ARE, EQUALS
<>	NE, IS NOT, ARE NOT
<	LT, IS LT, IS NOT GE, IS <
>	GT, IS GT, IS NOT LE, IS >
<=	LE, IS LE, IS NOT GT, IS <=
>=	GE, IS GE, IS NOT LT, IS >=
*	ALL, WHERE ALL

Values may be constants or another data-item value. Data item values are specified by preceeding the data item name with the "at sign" (**@**).

EXAMPLE: **SHIP.ADDRESS IS @BILL.ADDRESS**
The commas separating the values are the same as typing "OR data item IS".

EXAMPLE: **DELETE CLIENTS WHERE STATE IS MI,IN**

is the same as

DELETE CLIENTS WHERE STATE IS MI OR STATE IS IN

USAGE:

Caution should be exercised when using this command because of its global effects. When a deletion occurs, the record becomes inactive and will appear as if its not in the file. The deletion of a record does not affect the directory record count. This count indicates the total of inactive and active records. Sorting the file will remove all inactive records.

DESTROY

PURPOSE: Eliminate a database or file.

SYNTAX: **DESTROY database/file {OK}**

EXAMPLES: **DESTROY CLIENTS**
DESTROY PARTS OK
DESTROY PARTS.DEF

USAGE:

The **DESTROY** command erases all files of a database permanently and deletes the title from the data dictionary. A confirmation is always required to proceed. If the database title is different from any of the files making up the database (i.e., the data, definition or form filenames), the user will be prompted to erase these as well. If the confirmation OK is given in the command line, no prompts will be requested and all files associated with the database will be erased, regardless of the names.

If a database is using a file that is being used simultaneously by another database (e.g. **.DEF**, **.FRM**, or **.DAT** file), that file will not be destroyed when **DESTROY** is performed.

Individual files can be erased by specifying the extension name. All files on a drive can be erased by typing **DESTROY *.***

Wild card matches to database titles (e.g. CLI*) are not allowed.

CAUTION: Once a database or file has been **DESTROYED**, there is no way to recover the data.

If no dictionary is on the drive specified the argument is treated as a filename regardless of an extension.

DIC(tionary)

PURPOSE: **View entries in the data dictionary**

SYNTAX: **DIC {drv:}**

EXAMPLES: **DIC**
DIC B:
DIC C:

USAGE:

The **DIC**(tionary) command will give a listing of the data dictionary on any drive. If no drive is specified the current drive is assumed. The listing will appear in four columns with the title, dataset, definition set, and form filenames listed. The dictionary may also be listed by issuing the command,

LIST DATA.DIC BY TITLE, DATA, DEFINITION, FORM.

PURPOSE: View the list of files in the disk directory.

SYNTAX: **DIR {drv:}**

EXAMPLES:
```
DIR
DIR B:
DIR C:*.DAT
DIR CLIENTS.*
```

USAGE:

The DIR command will display the file directory of all files on a specified drive or a set of files matching certain names. The * is used as a wild card match, whereby all letters following the name are to qualify. The questions mark "?" is a wild card character matching only the character in the position of the name.

EXAMPLE: **DIR CL?ENTS.***

will display all files such as:

```
CLIENTS.DAT
CLIENTS.DEF
CLYENTS.DAT
CLYENTS.DEF
```

The file entries are displayed in four columns.

DISPLAY

PURPOSE: View selected records of a database.

SYNTAX: **DISPLAY database {WHERE condition}**

EXAMPLES:
```
DISPLAY CLIENTS WHERE NAME IS SMI*
DISPLAY CLIENTS WHERE STATE IS MI
DISPLAY DEFSET.DEF ST FIELD IS 1 2 4
DISPLAY DATA.DIC WHERE TITLE IS CLIENTS
DISPLAY CLIENTS ST ENROLL.DATE GT @TERM.DATE
DISPLAY CLIENTS WHERE ALL
```

WHERE condition:

A WHERE condition is one or more comparisons joined by AND or OR. Both AND and OR may not be used together in one command. A comparison is a phrase comparing a data item with one or more values:

	data-item	comparison-verb	value {,value...}
EXAMPLE:	**STATE**	**IS**	**MI,IN**

A comparison-verb compares the value of the data item with one or more values to determine if the WHERE condition will be satisfied.

Comparison-verbs are:

Verb	Synonym
=	EQ, IS, ARE, EQUALS
<>	NE, IS NOT, ARE NOT
<	LT, IS LT, IS NOT GE, IS <
>	GT, IS GT, IS NOT LE, IS >
<=	LE, IS LE, IS NOT GT, IS <=
>=	GE, IS GE, IS NOT LT, IS >=
*	ALL, WHERE ALL

Values may be constants or another data-item value. Data item values are specified by preceeding the data item name with the "at sign" (@).

EXAMPLE: **SHIP.ADDRESS IS @BILL.ADDRESS**

The commas separating the values are the same as typing "OR data item IS".

EXAMPLE: **DISPLAY CLIENTS WHERE STATE IS MI,IN**

is the same as

DISPLAY CLIENTS WHERE STATE IS MI OR STATE IS IN

USAGE:

The typical use for this command is to make an inquiry about a single record or a set of records meeting the stated search criteria. If a record is found that meets the search condition, it will be displayed on the console screen and may be printed by typing the Print function key. The user can continue searching for other records that meet the same condition or stop the search. The operator can not make changes to the database using the display command (see update command).

When specifying more than one value in a relational clause (e.g., **AGE IS 10, 12, 14** or **AGE IS 10 12 14**), the comma or space is interpreted as meaning logically a non-exclusive OR. The interpretation for this condition is **AGE IS 10 OR AGE IS 12 OR AGE IS 14.** It is considered ambiguous to include both logical operators (AND, OR) in the same command line. Hence the clause **AGE IS 10 20 AND SEX IS MALE** is ambiguous because it could mean we are looking for either male 10 year olds or male 20 year olds; or interpreting it another way, any 10 year old regardless of sex, or male 20 year olds.

The database can also be a definition or dictionary file. It may also be the directory of the data file. In these inquiries type the appropriate extension **.DEF, .DIC,** or **.DIR.**

PURPOSE: Create or revise a command procedure file

SYNTAX: **EDIT filename {.CMD}**

EXAMPLES: **EDIT COMMAND.CMD**
EDIT C:SALES.CMD

USAGE:

The **EDIT** command allows the user to create or revise an existing command procedure. The usage of **EDIT** command is similiar to **FORMAT** command.

There are two editing modes permitted with the **EDIT** command, the **replace** mode and **insert** mode. The **replace** mode is the default and this allows the user to type over anything on the screen, thereby replacing the current character with the new character. The arrow keys move the cursor to the appropriate screen position so that the replacement can be made. A carriage return puts the cursor on the next line in the left column. Typing the **insert character (INS CHAR)** function key will change the user from **replace** mode to **insert** mode. Typing the **insert character (INS CHAR)** key a second time returns to **replace** mode.

In the insert mode, any character typed will be inserted at the cursor position shifting the remainder of the line to the right, and advance the cursor to the right.

Typing the **delete character (DEL CHAR)** will delete a character at the current position, shifting the remainder of the line to the left. Typing the **insert line (INS LINE)** key in either mode will insert a new line, moving all the text down one line. Typing the **delete line (DEL LINE)** key in either mode will delete the current line from the cursor to the end of the line.

If the file specified does not exist, a message **New File** is displayed; otherwise the existing file will be edited. The user can decide at anytime during the **EDIT** process to keep the original and not use the current revision by typing the **A** for **Abort** function key. If it is desired to keep the original as well, before revising, make a copy of the original using a new name and edit the copy. (see the **COPY** command).

Unlike the **FORMAT** command, the **EDIT** command permits you more than 23 lines. You may page forward and back in the command file by using the **Next Page** and **Previous Page** function keys. You may go to the top or bottom of the command procedure file by typing the **Top Page** or **Bottom Page** function keys. After you have entered the 60th command, the screen does not scroll. Type the **REFRESH** function key to get the screen to scroll again.

If editting a new file, the video screen will be blank except for a prompt line at the bottom of the screen. Enter the commands on the screen exactly as you wish to see them.

If revising a command procedure, simply enter new information where desired on the screen. Delete undesired information by replacing it with spaces or deleting the characters.

Note: Unless explicitly specified, the new file will reside on the current drive. Form (.FRM) and Help (.HLP) files should be created and revised using **FORMAT** command. If the command procedure file is on another drive than the current drive, the drive must be specified. Use the **UPDATE** command to make changes to data files.

EMPTY

PURPOSE: Eliminate all records in a database

SYNTAX: **EMPTY database {OK}**

EXAMPLES: **EMPTY TRANSACT**
EMPTY B:CLIENTS OK

USAGE:

The **EMPTY** command will delete all records of the data base and reset the directory record count to zero. Note: The records are not recoverable. Either data base names or data file names may be specified in this command. The data filename and drive must be explicitly specified in this command, if not the current drive.

The user is requested to give a confirmation before the database records are deleted by typing Y <C/R> to a prompt. The confirmation can also be given in a batch procedure by typing the word OK following the database name. A space must separate the data base name from the confirmation.

ENTER

PURPOSE: Insert new records into a database

SYNTAX: **ENTER database [option]**

OPTIONS: [D] Means set auto default mode on
[R] Means set auto repeat mode on

EXAMPLES: **ENTER CLIENTS**
ENTER CLIENTS.DEF [R]
ENTER DATA.DIC

USAGE:

Once data is entered into the field, entering a carriage return will position the cursor to the next field. If the data entered fills the entire field, then a carriage return is automatically supplied by the system, and the user should not supply another one. Entering a back arrow will move the cursor backwards and entering a forward arrow moves the cursor forward. Entering the Return Tab, or down arrow, moves the cursor to the beginning of the next field. Entering the Shift Tab, or up arrow, moves the cursor to the beginning of the previous field.

In the entry mode the system checks for consistancy between the entered data and the edit criteria defined in each field of the definition file. Should the entered value not be within the defined range or meet the field type specification, the console will beep and the error will be indicated at the bottom of the screen. The cursor will automatically reset to the begining of the field that was in error and the user must correct the error to meet the edit criteria in order to proceed. The range error may be overridden by entering the override function key before entering a carriage return. The user must be cautious, since it is possible to truncate the data this way. In the standard mode of data entry (i.e., no switch options used), if only a carriage return is entered for the current field, the default value (if it is non-blank), will be inserted into the field, and the cursor advanced to the next field.

Two alternate entry modes are provided other than the standard mode. These are referred to as the auto-default mode and auto-repeat mode. When the auto-default mode is on, the default values that have been pre-defined in the definition file are automatically inserted when the cursor has been advanced to that field, without pausing first for operator input, as in the standard mode. If a default value is non-blank, the value is inserted and the cursor is automatically advanced to the next field. If the default value is blank, the cursor will be positioned at the beginning of the field. A back arrow will allow the operator to go back to the previous field if the value is to be changed.

The auto-repeat mode [R] permits the values entered in the previous record to be used in the following record without retyping. The screen is not cleared as in the standard mode when the continue option is selected. The auto-repeat mode is turned on or off by typing the **Auto-Repeat** function key. The auto-default mode is turned on or off by typing the **Auto-Default** function key. These modes can also be turned on when calling the enter program by invoking then as switch options, as shown above.

If the user wishes to abort the data entry mode while filling in the form, the Abort function key may be typed and the current record will not be saved. Any previous records entered that session will be saved. Normally, the user will exit from the enter routine using the save-end option (**End** function key).

If after entering a new record, the user has been signalled that there is no space left on the diskette or the database is full, a new database should be started on a new diskette. In order to be compatible with the record type of the full database, use the define command to build a new database and specify the form and definition file to be used as the full database (see **DEFINE** command).

Note: When given a "no space left on diskette" message, the last record entered must be re-entered on the new diskette. A database can hold up to 32,767 records.

FORMAT

PURPOSE: Create or revise a form or HELP screen.

SYNTAX: **FORMAT filename {.FRM}**

EXAMPLES: **FORMAT NEWFORM.FRM**
FORMAT NEWFORM
FORMAT C:OLDFORM.FRM
FORMAT HELPFIL.HLP
FORMAT INFORM.FRM OUTFORM.FRM

USAGE:

The **FORMAT** command allows the user to create or revise an existing **FORMAT** for data entry and retrieval. The **FORMAT** command is used in conjunction with the **DEFINE** command to create a database.

There are two editing modes permitted with the **FORMAT** command, the **replace** mode and **insert** mode. The **replace** mode is the default and this allows the user to type over anything on the screen, thereby replacing the current character with the new character. The arrow keys move the cursor to the appropriate screen position so that the replacement can be made. A carriage return puts the cursor on the next line in the left column. Typing the **insert character (INS CHAR)** function key will change the user from **replace** mode to **insert** mode. Typing the **insert character (INS CHAR)** key a second time returns to **replace** mode.

In the **insert** mode, any character typed will be inserted at the cursor position shifting the remainder of the line to the right, and advance the cursor to the right.

Typing the **delete character (DEL CHAR)** key will delete a character at the current position, shifting the remainder of the line to the left. Typing the **insert line (INS LINE)** key in either mode will insert a new line, moving all of the text down one line. Typing the **delete line (DEL LINE)** key in either mode will delete the current line from the cursor to the end of the line.

The file specified does not exist, a message **New File** is displayed; otherwise the existing file will be edited. The user can decide at anytime during the **FORMAT** process to keep the original and not use the current revision by typing the **Abort** function key. If it is desired to modify a form but keep its original, the user can do one of two things: make a copy of the original (see the **COPY** command) and modify it, or a form name, different from the original can be specified with the **FORMAT** command: **FORMAT ORIGFORM.FRM COPYFORM.FRM.**

Pressing the **Grid** softkey will paint a grid on the screen, making it easier to line up titles and items. The grid will not appear in a saved form. The **Refresh Screen** softkey will erase the grid from the screen, leaving titles, data items, etc. untouched.

If formatting a new file, the video screen will be blank except for prompts line at the bottom of the screen. Enter the form on the screen exactly as you wish to see it. Data item names are enclosed in brackets followed by

underscores.

EXAMPLE: **[ACCOUNT]**_____

Comments or other reference information may be shown on the screen by simply typing it in the desired location on the screen.

If revising a form procedure, simply enter new information where desired on the screen. Delete undesired information by replacing it with spaces or deleting the characters.

If a new form file is being formatted, the user will be prompted with an option to define a new database. If the Y is pressed, the **DEFINE** program is automatically loaded and run (see **DEFINE** command).

The **FORMAT** command can also be used to create a **HELP** file. Note, however, that files created by **FORMAT** do not contain a CR-LF characters. (See **HELP** command.)

PURPOSE: Assist operator in selecting procedures

SYNTAX: **HELP {helpfile} {.HLP}**

EXAMPLE: **HELP**
HELP BILLING

USAGE:

The **HELP** command assists the user by initiating a program, or procedure, selected from a menu on the console screen. The command is quite useful where the operator does not know the names of the data bases involved in the operations, the command syntax, or data item names.

Functionally, the **HELP** command reads a help file whose format is described below and displays this file on the console. The name **SYSTEM.HLP** is used as the default name of the help file when none has been specified.

The help file is created with a text editor or by using **FORMAT helpfile.HLP** If using the format program, specify the file extension as .HLP (e.g. **FORMAT** billing.HLP).

A HELP file has the following format:

Description 1 [Command Line 1]
Description 2 [Command Line 2]

Description n [Command Line n]

Note: The command lines are enclosed in the square brackets. Only the command descriptions are displayed after the **HELP** command. A number n that is entered, corresponding to the nth bracketed statement, will initiate that command statement according to the actual syntax between the brackets. Numbers begin with 1 and can be as large as 127. One HELP may call another HELP, or a command procedure. All program files with an extension of **.COM** or **.DBM** may be called. **.COM** programs must be preceeded with a dollar sign ($). Commands which are intrinsic to DBMS cannot be called (e.g. **DIC, DIR, LOG, SET, RESTART, SYS**) unless they are in a command procedure.

An example of a HELP file which handles financial accounting under the name **ACCOUNTS.HLP** may be constructed as follows:

HELP ACCOUNTS
1. Review of accounts on file **[PRINT ACCOUNT BY NAME, ADDRESS]**
2. Accounts receivable **[RUN ACCREC]**
3. Accounts payable **[HELP PAYABLES]**
4. Summary of Accounts by state **[TABULATE ACCOUNT BY STATE]**

Note: This HELP can call other HELP files, **PAYABLES.HLP**

A command line may also contain the **RUN** command. Using this facility, extensive command procedures may be executed very easily, global parameters can be used for the command procedure and chained help screens may be used.

INDEX

PURPOSE: Create or Update a database Index

SYNTAX: **INDEX database (USING d1,d2,...,d8) [option]**

OPTIONS: **[D]** Means permit Duplicate Key Values
[U] Means Update Index

EXAMPLES: **DEFINE CLIENTS**
DEFINE INVENTRY
DEFINE QUICKDB exists.dat,exists.frm,exists.def

USAGE:

The INDEX command permits creation and updating (**[U]**) by using a key of up to eight data-items. The database file may contain duplicate key values if the **[D]** option is specified.

No more than one index may be created for one data (**.DAT**) file. You may update an index without specifying the key items on the command line.

Indexes are used (and updated) by the **APPEND**, **DELETE**, **EMPTY**, **ENTER**, and **UPDATE** commands. Indexes are used for quick access to specified records (through use of a WHERE condition) by the **DISPLAY**, **DELETE**, **SELECT**, and **UPDATE** commands if the WHERE condition specifies selection by Key Items and no OR clauses are used. **DELETE** and **SELECT** also require the search to be bounded. Examples of bounded searches are:

NAME IS SMITH
NAME IS > A AND NAME < D

Limitations:

Up to eight Data Items may be specified as an index key.
Combined length of Key Data Items may be up to 127 bytes.

The Index is updated or used by the following commands: **APPEND**, **DELETE**, **DISPLAY**, **EMPTY**, **ENTER**, **INDEX**, **LIST**, **PRINT**, **SELECT**, and **UPDATE**.

JOIN

PURPOSE: Attach data items of two databases by matching data item values.

SYNTAX: **JOIN database1 database2 MATCHING d1, d2,... d32 [option]**

OPTIONS [D] means store database2 default values in result record if database1 record has no match.

RESULT SET: **R$.DAT, R$.DEF, R$.FRM**

EXAMPLES: **JOIN ACCOUNTS ORDERS MATCHING ACCOUNT-NO**
JOIN ACCOUNTS ORDERS BY ACCOUNT-NO [D]
JOIN PRODUCTS PRICES BY ITEM-NUMBER

USAGE:

JOIN creates a new database containing data-items from both database1 and database2. Records from each are joined together whenever a match occurs between specific data-value(s) stored in dataset1 common to those data-value(s) dataset2. Specific data-items to be matched are specified in the command line. These will be referred to as the "common data-item". A requirement for the **JOIN** is that the common data-item have identical data definitions in both databases (i.e., both have the same field size and are the same type).

A form file and definition file are created as part of the Result set. They contain the labels and data definitions of both databases. The result record will have its data items ordered with the data items from database1 followed by those of database 2. The common data-item which would be redundant in database2 is removed so that the resultant record has d1+d2-K fields and 1+R1+R2-C bytes, where d1 and d2 are the total number of data items in database1 and database2, K is the number of common data-items specified in the command line; R1, R2 is the total field sizes of database1 and database2, C is the total field size of the common data-items. This type of join is called a "natural join", i.e., the join occurs on equal data-values and the redundant data values are removed. In general, if there are n occurences of the common data-item in database1 and m occurences of it in database2, then n*m result records are produced.

A special type of join may be invoked by specifying the switch [D] following the command line. In this case, if no match occurs between any one record in database1 with database2, a record will be forced to appear in the result set with the Default values of database2 joined to the unmatched database1 record.

Joins are restricted to a maximum result record size of 1024 bytes, 127 data-items, 32767 records, and a single CRT screen. The result form is automatically edited and compressed so that it only contains the labels and underscores from the input forms (see PROJECT command).

LIST

PURPOSE: View records of a database in sequential order.

SYNTAX: **LIST database/file { BY d1, d2,...**
{ AND COMPUTE statistic d3, d4,...} } [option]

Where **statistic** can be one of the following words:
TOTAL (TOT), SUBTOTAL (SUBTOT), MIN, MAX, AVERAGE (AVE, AVG), STAX

The word **COMPUTE** may be abbreviated by using the symbol **@**.

The word **STAX** is used to indicate total, min, max, average

Options: **[A]** Means accumulate subtotals
[P] Means print
[X] Means suppress label headings in columnar format

EXAMPLES:

LIST CLIENTS
LIST CLIENTS BY NAME, CITY AND STATE
LIST CLIENTS NAME, AND CITY AND STATE
LIST CLIENTS BY NAME AND COMPUTE TOTAL WAGE, HOURS
LIST CLIENTS NAME @ TOT WAGE SUBTOTAL USING DEPT
LIST CLIENTS.DEF
LIST CLIENTS.FRM
LIST REPORT.SUB

USAGE:

Several output formats can be generated with the **LIST** command. Delimiters may be a comma, one or more spaces or AND (separated by a space on both sides). The output may be sent to the printer instead of the console by using the option [P] (see also **PRINT** command).

The output formats are classified as:

1. Screen
2. Columnar
3. Columnar with statistic
4. ASCII

The screen format is produced by the statement **LIST** database. **LIST** database displays one record at a time starting from the first record of the file. The next record can be displayed or previous (last) record can be displayed by pressing the appropriate keyboard characters as prompted. The **TITLE** command can be used to set spacing (see **TITLE** command). Dictionary and Definition files and Database Directories may be listed or printed by including the extension names.

EXAMPLES: **LIST DATA.DIC**
LIST CLIENTS.DIR
LIST CLIENTS.DEF

The columnar format will produce a list of records containing only the data-items specified. This list will produce three columns. Each column is labeled as NAME, STATE and PHONE. the label names must correspond to those in the form file for the database. The label heading may be suppressed by using the [X] switch. The user may specify up to 32 data-items in any order. Each column is assigned a number of printing positions equal to one plus the number of underscores or label length depending on which is larger, as defined in the form. For example, if the field name has 23 underscores, 24 printing positions are assigned. If the field has 2 underscores, then 6 printing positions are assigned because the label defines the size.

EXAMPLE: **LIST CLIENTS BY NAME , STATE AND PHONE**

The columnar with statistic format can supply totals, and other statistics at the end of the list, as well as subtotals at level breaks. Each time the dept value changes, a subtotal is printed. It is necessary that the database be sorted by the subtotal data items, either ascending or descending sequence is acceptable. If it is not sorted an error message is printed, indicating that the database is not in logical sequence. The database must be sorted if this error occurs (see **SORT** command).

The user can specify more than one data-item per statistic and more than one statistic, but not one statistic following directly another statistic.

EXAMPLE: **COMPUTE TOTAL WAGE, HOURS AND AVERAGE WEEKS, HOURS**

The above example is acceptable, but the following example is an error.

EXAMPLE: (ERROR) **COMPUTE TOTAL, AVERAGE HOURS**

The number of printing positions is determined as in format 2, except that columns with totals are given at least 13 printing positions. In formats 1, 2, and 3, numeric, dollar, and date field types are right justified, with blanks to the left. Alphabetic, and alphanumeric field types are left justified, with blanks to the right.

Certain errors that can occur are indicated by either a series of periods..... or a series of asterisks **********. The periods imply that the form indicates too few underscores to actually print the output. The asterisks indicate that the total, subtotal, or average has overflowed (exceeded maximum allowed 10 digit number).

The ASCII Format will list an ASCII file (e.g. a text file, command file, form file or help file). No interpretation is done in this case, except to expand the tab characters.

EXAMPLE:
LIST MENU.HLP
LIST START.SUB
LIST LEDGER.CMD
LIST.LEDGER.FRM

Any file that has an extension other than .COM, .CVL, .DBM, .DEF, .DIR, .DAT, or .IDX will be listed as ASCII.

PURPOSE: Log a new disk in the computer.

SYNTAX: **LOG** or **LOG DISK** or **LOGDISK {Dr:diskname}**

EXAMPLES: **LOG DISK**
LOGDISK
LOGDISK B:
LOG C:DISKNAME

USAGE:

The **LOGDISK** command is to be issued whenever a diskette is changed from a disk drive. This is equivalent to a disk reset in CP/M (Control-C). If a disk is changed and the new disk is not logged in, it can only be read and not written to. A write error will occur if attempting to write to a diskette not logged in.

A user message, asking for diskette to be inserted, will be displayed if a drive is designated with the LOGDISK command. If a disk name is also listed, it will be included in the suer message.

Note: **Control-C** will not do a disk reset when in DBMS.

POST

PURPOSE: Update data item in one database with those from another

SYNTAX: **POST database1 database2 MATCHING d1 ,d2,... {opr d3, d4,... opr d5, d6,... d32}**

Where **opr** can be any of the following operators:

REP, REPLACE, ADD, SUM, SUB, SUBTRACT

RESULT SET: **R$.DAT, R$.DEF, R$.FRM**

EXAMPLES:

POST INVENTRY TRANSACT MATCHING PARTNO AND ADD QUANTITY AND REPLACE DAT
POST ACCOUNTS RECEIPTS USING ACCTNO, ADD AMOUNT, QUANTITY
POST ACCOUNTS RECEIPTS ACCTNO ADD ORDERS SUB STOCK REP DATE

USAGE:

The user can reference up to 32 data items providing all the fields named in the command line exist in both data bases. Also required, is that data items referenced, are defined as having the same field name, type, and size in both data bases. Each operator encountered in the command line performs the same update operation on every data item in the list until a new operator is encountered.

EXAMPLE:

POST ACCOUNTS RECEIPTS BY PARTNO AND REPLACE DATE, PAYMENT

is the same as,

POST ACCOUNTS RECEIPTS BY PARTNO AND REPLACE DATE, REPLACE PAYMENT

The use of **COMMAS** and the word **AND** are optional. Commas and spaces are interchangable.

Any data item belonging to both data bases may be specified in the command line. However, only data items declared as numeric (N) or dollar ($) may be preceded by an arithmetic operator (e.g. ADD). A **REPLACE** may be done on any field type. If an error condition exists because of an invalid operation on a non-numeric field, the program will abort and no updates will be made.

The updating sequence occurs by matching data item value of database1 and database2 records. When a match occurs, a copy of the existing database1 record is written out to **R$.DAT.** The database1 record is then updated as indicated by the operators in the commmand line. If an overflow error occurs in any field of the record, the database1 record is not updated, and not written to **R$.DAT.** An overflow error will produce a message on the console screen. The operator will be required to intervene and either choose to abort the program or continue on, thereby skipping updating that record. If the abort option is chosen, no result set is written.

Note: A portion of the database1 may get updated if previous updates had been done without error.

If no data items or operators are specified in the command line, the command will default to a complete replacement operation.

EXAMPLE: **POST ACCOUNTS RECEIPTS USING ACCTNO**
will cause the default

AND REPLACE d1,d2,..
where d1,d2,... are all the data items in database2.

Note: When using as above, all data items named in database2 must also be in database1. The program will abort if this condition is not true.

The result data base created provides a convenient way to maintain an audit trail and to also verify that all database2 records have been updated. The RESULT set **R$.DEF** and **R$.FRM** are exact copies of the database1 definition and form files, respectively.

The **POST** command will perform in the shortest time whenever the two data bases are sorted by the matching data items specified in the command line. An insignificant decrease in performance is observed with databases which have only a few records out of sequence. That is, if a few records have been added to a master file that was previously sorted, sorting again and running the post program will not significantly improve the performance of the **POST** command.

PRINT

PURPOSE: Print records of a database in sequential order.

SYNTAX: **Print database/file {BY d1, d2,...**
{AND COMPUTE statistic d3, d4,...} } [option]

Where **statistic** can be one of the following words:
TOTAL (TOT), SUBTOTAL (SUBTOT), MIN, MAX, AVERAGE (AVE, AVG), STAX

The word **COMPUTE** may be abbreviated by using the symbol **@**.

The word **STAX** is used to indicate total, min, max, ave

Options: **[A]** Means accumulate subtotals
[X] Means suppress label headings in columnar format

EXAMPLES:
```
PRINT  CLIENTS
PRINT  CLIENTS BY NAME, CITY AND STATE
PRINT  CLIENTS  NAME, AND CITY AND STATE
PRINT  CLIENTS  BY  NAME  AND COMPUTE TOTAL WAGE, HOURS
PRINT CLIENTS NAME @ TOT WAGE SUBTOTAL USING DEPT
```

USAGE:

Several output formats can be generated with the **PRINT** command. Delimiters may be a comma, one or more spaces or the conjunction AND (delimited by a space on both sides).

The output formats are classified as:

1. Screen
2. Columnar
3. Columnar with statistic
4. ASCII

The screen format is produced by the statement **PRINT** database. The **TITLE** command can be used to set spacing (see **TITLE** command). Dictionary and definition files and Database Directories may be listed or printed by including the extension names.

EXAMPLE:
```
PRINT DATA.DIC
PRINT CLIENTS.DIR
PRINT CLIENTS.DEF
```

The columnar format will produce a list of records containing only the data-items specified. This list will produce three columns. Each column is labeled as NAME, STATE and PHONE. the label names must correspond to those in the form file for the database. The label heading may be suppressed by using the [X] switch. The user may specify up to 32 data-items in any order. Each column is assigned a number of printing positions equal to one plus the number of underscores or label length depending on which is larger, as defined in the form. For example, if the field name has 23 underscores, 24 printing positions are assigned. If the field has 2 underscores, then 6 printing positions are assigned because the label defines the size.

EXAMPLE: **PRINT CLIENTS BY NAME , STATE AND PHONE**

The columnar with statistic format can supply totals, and other statistics at the end of the list, as well as subtotals at level breaks. Each time the dept value changes, a subtotal is printed. It is necessary that the database be sorted by the subtotal data-items, either ascending or descending sequence is acceptable. If it is not sorted an error message is printed, indicating that the database is not in logical sequence. The database must be sorted if this error occurs (see **SORT** command).

The user can specify more than one data-item per statistic and more than one statistic, but not one statistic following directly another statistic.

EXAMPLE: **COMPUTE TOTAL WAGE, HOURS AND AVERAGE WEEKS, HOURS**

The above example is acceptable, but the following example is an error.

EXAMPLE: (ERROR) **COMPUTE TOTAL, AVERAGE HOURS**

The number of printing positions is determined as in format 2, except that columns with totals are given at least 13 printing positions. In formats 2 and 3, numeric, dollar, and date field types are right justified, with blanks to the left. Alphabetic, and alphanumeric field types are left justified, with blanks to the right.

Certain errors that can occur are indicated by either a series of periods..... or a series of asterisks **********. The periods imply that the form indicates too few underscores to actually print the output. The asterisks indicate that the total, subtotal, or average has overflowed (exceeded maximum allowed 10 digit number).

The ASCII Format will **PRINT** an ASCII file (e.g. a text file, command file, form file or help file). No interpretation is done in this case, except to expand the tab characters.

EXAMPLE:
PRINT MENU.HLP
PRINT START.SUB
PRINT LEDGER.CMD
PRINT.LEDGER.FRM

Any file that has an extension other than .COM, .CVL, .DBM, .DEF, .DIR, .DAT, or .IDX will be printed as ASCII.

To **PRINT** to a file you will need to use the **SET PRINTER** file name command (see **SET** command).

PROJECT

PURPOSE: Create RESULT database from selected data items of a database

SYNTAX: **PROJECT database BY d1,d2,d3,..., d32**

RESULT SET: **R$.DAT, R$.DEF and R$.FRM**

EXAMPLES: **PROJECT CLIENTS BY NAME CITY STATE**
PROJECT EMPLOYEE USING NAME AND SALARY
PROJECT DEPT NAME, DEPTNO

USAGE:

For each record of a database, specific data-items are extracted resulting in a new dataset containing only the desired data-items. The result database contains the form and data definitions of the selected data-items. The edit criteria is preserved as well.

The form created with a projection will have labels of the data-items as they appear in the database form, followed by the number of underscores allocated to the data-value. The record size of the result dataset is equal to the sum of the individual field sizes of each of the data-items projecting.

It is possible to **PROJECT** the same data-item more than once. If groups of data item names appear more than once on the form, **PROJECT** will create one record for each repetition of the group, having duplicated the values for the data items which are not repeated. Any repeating data item must have the same definition (i.e. type size). For example, if the data item **QUANTITY** appears six times in the format, all six occurrences must have the same definition. If there are repeating groups, all repeated data items must have the same number of repetitions. In the command **PROJECT ORDER BY ORDER.NO QUANTITY PRICE**, if **QUANTITY** and **PRICE** are repeating groups, and **QUANTITY** appears six times, **PRICE** must also appear six times. **ORDER.NO**, since it appears only once, will appear in each of the six records written.

In general if there are n records in the database, and r repeated groups, there will be n*r records in the result set. All records are projected, including inactive records which remain inactive in the result set.

READ

PURPOSE: Transfer records from an ASCII file to an existing database

SYNTAX: **READ database { input - filename }**
If no input file is specified, **R$.ASC** is read

EXAMPLES: **READ CLIENTS**
READ CLIENTS APPLCANT.ASC

USAGE:

The **READ** command is useful for converting existing data files that are in ASCII format to a SERIES 20 Database. The database must be defined previously (see **DEFINE** command). The ASCII data file can contain either fixed or variable length records according to any of the formats described below:

	FORMAT	RECORD TYPE
1.	Fixed length	Continuous stream of string characters, no binary characters separating fields or records.
2.	Fixed length	String characters with < C/R > or < C/R > < L/F > separating records, no field delimiters.
3.	Fixed or variable length	Strings enclosed in quotes ("), commas separating fields, < C/R > or < C/R > < L/F > separating records.
4.	Variable length	Strings separated by tabs (ASCII 09), records separating by < C/R >.
5.	Fixed or variable length	String characters with < C/R >< L/F >, separating fields, additional < C/R >< L/F > separating records.

Note: All ASCII data files must be terminated with a control-Z (ASCII 01AH), indicating end of file.

All input records are formatted and encoded by the database form and definition files. The input record is formatted according to the number of underscores given to each data item in the form. For example; if a data item called NAME has 12 underscores in the form, the input string will be read as if it is 12 characters in length. If the string is delimited as described above and is less than 12 characters, the remainder of the field will be blank filled. If the string is greater than 12 bytes and delimited, an error message will be displayed on the console indicating which ASCII record number was in error. This record will be skipped and not read into the database. If the input record contains binary data or a quote character (") other than field or record delimiters, it too will cause an error and not read into the database. Each field will be encoded as described in the definition file and checked for consistency with the declared field type. If they do not agree, an error message is displayed and the record skipped.

RENAME

PURPOSE: Change the name of a database or file.

SYNTAX: **RENAME newname=oldname**

EXAMPLES: **RENAME CLIENTS = APPLCNTS**
RENAME NEW.DAT = OLD.DAT
RENAME D:NEW = D:OLD

USAGE:

The **RENAME** command changes the name of a database title, including the names of the data, form and definition files. If a database is using a file that is being used simultaneously by another database (e.g. **.DEF**, **.FRM**, or **.DAT** file), that file will not be renamed when **RENAME** is performed.

Files and databases may only be renamed on the same drive. Filenames are distinguished from database titles by using an extension name (e.g., **CLIENT.DAT**). If no dictionary is on the drive in which the renaming is done, the name is always considered to be a filename. Attempting to **RENAME** a file or databae title to a name which exists already will be denied. An existing file, or database must be destroyed first before using **RENAME**.

REORG

PURPOSE: Reorganize the structure of a database adding and deleting data items.

SYNTAX: **REORG database {formname.FRM} [option]**

OPTIONS: **[R$]** Means create result set

RESULT SET: **R$.DAT, R$.DEF, R$.FRM** (if option chosen)

EXAMPLES:
```
REORG  CLIENTS
REORG  CLIENTS  NEWFORM.FRM
REORG  CLIENTS  NEWFORM.FRM [R$]
```

USAGE:

The **REORG** command permits the user to change the order of, and add or delete data items in a database. Each record of the database will be rearranged into a new format defined by a different form file (not the current database form). If no form file is specified, the **REORG** program prompts the operator for a form file name. In order to use the **REORG** program, the user first creates the desired form using the format command. The data-items that are to be rearranged or saved must have the same label names and have the same number of underscores as the original database form. After designing the new format, save it and end the format program. Do not choose the option to define a new database. After the format is completed, use the **REORG** command.

The original database is reorganized according to the new format and a result set created. If the new format has new data item names, the user will be prompted to supply appropriate definitions as done in the define program. New data items will have new data values initialized with nulls. If any data items have been deleted, the user will be prompted to confirm the deleted ones. The user is prompted by the system to choose the option whether or not to replace the original database with the result set.

In batch operation, if the new form name is supplied on the command line, the user will not be prompted unless new definitions are required. If no new definitions are required, the operator will not be prompted.

Note: In batch, unless the switch option **[R$]** is specified in the command line, the original database is replaced by the result set.

REPORT

PURPOSE: Create, Revise, or Describe a Report Specification
Print or List a Report

SYNTAX: **REPORT database [option]**
or
REPORT report-name WHERE condition [option]
(WHERE condition is defined in the DISPLAY command)

EXAMPLES: **REPORT WAREHSE**
REPORT WAREHSE [C]
REPORT ACCOUNTS WHERE PURCHASE < 1/1/83 [PF]

USAGE:

The **REPORT** command permits the user to create special format reports, having full control of placement of information on the page, handle special page sizes, use pre-printed forms, and specify multiple level control breaks.

If no option is given, the user is presented a menu of options (Report Menu). The menu is the same as above except that the **[P]**, **[PF]**, and **[L]** options are shown as one choice. The user is asked if the report is to be printed and, if so, whether forms alignment is desired.

Except when the user specifies the **[P]** or **[PF]** option on the command line, the user is returned to the menu after completion of each option selected.

A report specification **(.RPT)** file is created by first creating a report form. The report contains model lines, each of which represents a line to be printed in the report. The report form is similar to an entry form, except that placement of data to be printed is dexcribed by placing thedata item name where it is to be printed instead of placing underscroes in the form.

After the report form is created, the user is asked to specify characteristics of each model line in the report form. The user may select the **short** or **long** method for specifying characteristics. The short method asks very few questions and allows a report specification to be defined quickly, with later fine tuning necessary for line spacing. The long method requires more input from the user and allows detailed fine tuning during the definition process. In both methods, default values are presented for most values requested, The defaults are determined from either the most frequently used values or information contained on model lines in the report form.

In the short method, only **Line Control** and **Computed Item Specifications** are requested. When using the long method, the following information is requested:

Page Layout Specification
Line Control Specification
Item Control Specification

Page Layout specification requires the **page length (1-127 lines)**, **Top Margin**, **Bottom Margin**, **Left Margin**, and **Right Margin.** A page may be up to 127 lines long and up to 132 columns wide.

Line Control Specification may be **First page (FP)** only, **Page Heading (PH)**, **Every Line (EL)**, **Before (BS)** and **After Section (AS)**, and **Last Page (LP)** only. The user may specify placement of the line by positioning to a specified line number before and after printing or, instead of line number placement, the number of blank lines to leave before and after printing. A look-ahead capability allow assurance that information printed near the bottom of the page will appear reasonable. For Before and After Section line types, the user also specifies which data items (**control items**) cause a section change.

Item Control Specification consists of describing the **printing position, printing width, print format,** and **print function.** Items may be **Text, System values** for **Data** and **Page Number, Data Items** or **Computed Items** which are not contained in the database file. An equation must be given for Computed Items.

Allowable Print Formats are:

SD - System Default - simulates the **LIST** command
RJ - Right Justify
LJ - Left Justify
LZ - Print Leading Zeroes
BZ - Blank if Zero
FD - Floating Dollar sign
LS - Leading Stars
NP - Not Printed

Print functions are required for data items in After Section and Last Page lines:

VAL - Print the Value of the Item
SUB - Subtotal - reset to zero at each section change
TOT - Total - not reset to zero to each section change
MIN - Minimum value in the section
MAX - Maximum value in the section
AVE - Average value in the section

Note: When Printing or Listing a report, the database file must be in sort by the Control Items specified.

RESTART

PURPOSE: Continue processing of an interrupted procedure

SYNTAX: **RESTART {drv:}**

EXAMPLES: **RESTART**
RESTART B:

USAGE:

The RESTART command can be used whenever a running command procedure was aborted, either by a system error, operator performing a control-C, a hardware reset, a power failure, or hardware failure. The active batch file called **R$.DBM** contains the remaining job stream and the **RESTART** will permit continuation of the batch procedure using this file. If the batch file **R$.DBM** is on a drive other than the current drive, the drive must be specified when giving the restart (e.g., **RESTART D:**). The user should sign onto the drive that was current at the time of abort, then issue the **RESTART DRV:** command. The user must be cautioned that if a command is aborted by a hardware reset or failure before the current operation is finished, strange results may be produced if a **RESTART** is attempted. It is suggested that in this case the user should determine the consequence of continuing the batch by issuing a **RESTART.** The command line analyzer has been designed, so that Control-C aborts the batch procedure only between commands, thereby permitting a proper **RESTART.** Exceptions are the **LIST** and **PRINT** commands which may be aborted during execution.

RUN

PURPOSE: Start processing a command procedure having directives

SYNTAX: **RUN batch {.CMD} {param1 param2,...param9}**

EXAMPLES:
```
RUN GLEDGER
RUN REPORT JUNE AUG
RUN ORDERS
```

USAGE:

The **RUN** command can be used to process a sequence of commands that are in a text file. The command sequence is created by using the system editor (e.g. CP/M ED.COM).

The output created from the command file is a work file under the name **R$.DBM.** This file is executed until an abort or error condition occurs. The command file created by the system editor must have an extension of **.CMD.** The format of the file is as follows:

```
COMMAND LINE1.......
COMMAND LINE2.......
      .
      .
COMMAND LINE127......
```

Up to 127 lines may be in the command file.

Statements other than command lines may be included in the file, such as comment lines or **RUN** directives. Comment lines are lines which have a semicolon (;) as the first character of the line. These lines are included for programmers' comments and will not appear in the **R$.DBM** work file. The **RUN** directives are preceeded by an asterisk (*) and a keyword. The allowable **RUN** directives are:

```
*IF condition
*ENDIF

*ECHO ON
*ECHO OFF

*MESSAGE message line
*GET {param1,param2,...param9}
*LET param = value

*END
*ABORT
```

The ***IF** directive can be used to test for certain conditions, especially in situations where there are parameters to be passed to the command stream. The ***IF condition** can be any logical condition as defined in the SELECT command except parameter numbers are used instead of data-item names.

EXAMPLE:
```
*IF $1 > 40
      SELECT CLIENTS WHERE STATE IS MI, IN
*ENDIF
```

The above will include the statement **SELECT CLIENTS WHERE STATE IS MI, IN** if the parameter $1 is greater than 40. If this condition is not true, then every statement between the ***IF** and ***ENDIF** is excluded from processing. The ***IF** and ***ENDIF** directives may be nested up to 7 levels.

The ***GET** directive permits the user to enter parameter values from the keyboard into the command file at run time. For example, if the directive:

```
*GET $4
```

appears in the run file, the user will be prompted with a colon(:) and a value must be entered followed by a carriage return. The value entered will be assigned to the parameter value $4. More than one value may be entered, providing there occurs an equal number of parameters specified in the ***GET** directive. The parameters may be seperated by either blanks or commas. Parameters that have embedded blanks or commas must be enclosed in single or double quotes. For example, the directive:

```
*GET $3,$5
```

will prompt the user as shown below and the following could be entered:

```
:500,"Los Angeles"
```

Any parameters specified in the ***GET** and ***LET** directives supercede those on the run command line. For example, if the user types **RUN REPORT JUNE AUG,** and the directive ***GET $2** appears in the run file, the value **AUG** will be superceded by whatever value the operator types in response to the ***GET $2.** The parameter $1 will still be assigned **"JUNE".**
The directive ***MESSAGE** (or***MESS, *MSG**) will display a message on the console screen at run time. If the message is blank, then only a carriage return line feed is issued. The ***MESSAGE** can be included between an ***IF** and ***ENDIF** condition, so that the message is only printed if the condition is true.

The ***LET** directive permits assignment of a value to a parameter within the command procedure. The equals sign (=) must be surrounded by spaces.

EXAMPLES:
```
*LET $4 = N
*LET $5 = "LOS ANGELES"
*LET $5 = $4
```

In the last example, **$4** must have been previously defined in a ***GET** or ***LET** directive.

The ***ECHO OFF** can be used to turn off the display of the batch stream before it actually starts executing.

Note: The echo is always turned back on if an error or abort, condition occurs. It also is turned back on when the command procedure is completed or a **HELP** command is encountered.

Parameters to be passed to the command file are preceeded with a single dollar sign ($) and a digit between 1 and 9. These are replaced with the parameter values on the command line following the batch filename. If there are more or fewer parameters in the command line than appear in the command file, an error condition occurs, aborting the **RUN**. The current date may be passed to any of the commands of the command file by including the parameter "$TODAY" in the command. It too will be replaced with the current sign on date.

The ***ABORT** directive will abort the **RUN** program if encountered, and return control to the operator. No command file will be processed in this case. The ***END** directive, when encountered, will not include any statements that follow. The job stream is started as soon as the ***END** is encountered. An ***END** is optional at the end of the command file.

PURPOSE: Save a RESULT database.

SYNTAX: **SAVE database/file {OK}**

EXAMPLES: **SAVE THIS**
SAVE CLIENTS OK
SAVE DATAONLY.DAT

USAGE:

The SAVE command copies the result set files **R$.DAT, R$.DEF,** and **R$.FRM** and gives them the new name specified in the command. In addition, the new-name is used as the title of a new database and writes this title to the data dictionary. If the database exists previously, then a confirmation is required to erase and replace this name with the new database. The confirmation may be given at the end of the command line in batch mode by specifying **OK.** Files may be saved individually, in which case no dictionary record is entered. Filenames are distinguished from database names by specifying an extension.

EXAMPLE: **SAVE CLIENTS.DAT**
would copy the file **R$.DAT** to the file CLIENTS.DAT.

PURPOSE: Select datatase records meeting logical conditions, overwriting the current **RESULT** database.

SYNTAX: **SELECT database {WHERE condition}**

RESULT SET: **R$.DAT, R$.DEF** and **R$.FRM**

EXAMPLES: **SELECT EMPLOYEE WHERE SEX IS MALE**
SELECT EMPLOYEE WHERE AGE IS GT 30
SELECT EMPLOYEE.DEF WHERE FIELD IS LT 10
SELECT CLIENTS WHERE ALL

WHERE condition:

A WHERE condition is one or more comparisons joined by AND or OR. Both AND and OR may not be used together in one command. A comparison is a phrase comparing a data item with one or more values:

	data-item	comparison-verb	value {,value...}
EXAMPLE:	**STATE**	**IS**	**MI,IN**

A comparison-verb compares the value of the data item with one or more values to determine if the WHERE condition will be satisfied.

Comparison-verbs are:

Verb	Synonym
=	EQ, IS, ARE, EQUALS
<>	NE, IS NOT, ARE NOT
<	LT, IS LT, IS NOT GE, IS <
>	GT, IS GT, IS NOT LE, IS >
<=	LE, IS LE, IS NOT GT, IS <=
>=	GE, IS GE, IS NOT LT, IS >=
*	ALL, WHERE ALL

Values may be constants or another data-item value. Data item values are specified by preceeding the data item name with the "at sign" (**@**).

EXAMPLE: **SHIP.ADDRESS IS @BILL.ADDRESS**

The commas separating the values are the same as typing "OR data item IS".

EXAMPLE: **SELECT CLIENTS WHERE STATE IS MI,IN**

is the same as

SELECT CLIENTS WHERE STATE IS MI OR STATE IS IN

USAGE:

When using this command, records meeting the search condition are selected from the parent database and written to a new data file called **R$.DAT.** Duplicate copies of the parent definition and form files will be created under the names **R$.DEF, R$.FRM.** The new data file, **R$.DAT,** will contain all the items of the parent set in this case.

Note: Select will only output active records, thus can be used to remove inactive records from a file without sorting. In this case, it is up to the user to erase the original set following the select and rename the result set to that of the original.

In order to use the selected database the user must enter a record into the dictionary calling the names of the output files or by renaming or copying them onto a new diskette, where a dictionary entry may already exist. In general, one can reserve a dictioentry under the title **RESULT** and specify the data, form, and definition as **R$.** These files are created on the current disk drive.

PURPOSE: Set DBMS operating parameters.

SYNTAX: **SET {function status}**

EXAMPLES:
```
SET AUTOSEL OFF
SET ECHO OFF
SET MASTER B:
SET ABORT OFF
SET DATE DMY
SET PRINTER B:REPORT
```

USAGE:

The set command will set the state of several system functions. These include:

ABORT ON/OFF	=	**Overrides batch abort if error.**
AUTOSEL ON/OFF	=	**Automatic drive search for command**
DATE MDY/DMY/YMD	=	**Day, month, year sequence**
ECHO ON/OFF	=	**Crt echo of program display and msgs**
MASTER DRV:	=	**Search drive (A-H) for command**
PRINTER FILENAME {.PRT}	=	**Send printer output to a disk file**

The **ABORT ON/OFF** (default is ON) is set to OFF if the automatic batch abort is to be overridden (i.e., batch will not abort) when an error occurs in batch. The only way to stop a batch when **ABORT OFF** is set is to use a control-C or reset the machine. Normally the system should be run with **ABORT ON** (default). This way any errors encountered in batch will not be propagated.

The **AUTOSEL ON** permits the system to search the master drive (default is A) for the entered command if it is not found on the current drive. When **AUTOSEL OFF**, only the current drive is searched for the command.

The master drive can be set to any drive (A-H), in which case the command will first look to the current drive and then to the master drive. When **SET MASTER drv:** is given, AUTOSEL is set to ON automatically.

The **SET ECHO OFF** is used primarly in batch procedures to inhibit echoing of messages and command lines on the CRT. The ECHO is set ON automatically when a command procedure is completed, a HELP screen is displayed or a system error occurs. When SERIES 20 is initially loaded, AUTOSEL, ABORT and ECHO are both set to ON, and the MASTER is drive A:.

The **SET DATE** permits user to set date format for either entry or display. The following formats are displayed when set as shown:

```
MDY ->  mm/dd/yy
DMY ->  dd.mm.yy
YMD ->  yy-mm-dd
```

The delimiter, if entered, can be (/), (.) or (-) on any of the above formats. It will be displayed with the delimiter shown above according to the type set.

The date format is only an entry and display parameter and does not affect the value stored. Therefore, it may be enetered with the **SET DATE** command in one format and displayed later with another.

SET PRINTER is used to send output which would normally go to the printer to a disk file instead. It is useful for two purposes. The primary purpose is to include a report from DBMS with text from a word processor. Once the DBMS report is placed in a disk file, it may be merged with text using the word processor. The second purpose is to create a trace of commands executed by typing a **Control-P** prior to starting a command procedure. **Control-P** should be typed again after tracing is complete.

If no file extension is given, **.PRT** is used.

Printer output may be reset to the printer by the **SET PRINTER** command without a disk file name.

SORT

PURPOSE: Sort database records by data item values.

SYNTAX: **SORT database BY d1 , d2,...d32 [option]**

OPTION: **[D]** sort in descending order

RESULT SET: File is re-written

EXAMPLES: **SORT CLIENTS BY NAME**
SORT CLIENTS BY CITY STATE AND ZIP-CODE

USAGE:

The sort will order d1, as the major sort element then d2, then d3, and so on. Up to 32 items may be specified, delimiters may be spaces, commas or AND. If [D] follows the command line, the file will be sorted in descending order.

A temporary work file (**R$.SRT**) is created on the current disk drive for all sorts and the data file is rewritten in the order sorted. The work file is erased at the end of the sort. If work space is in question, it is a good idea to put a blank disk into an unused drive, log onto the drive and start the sort to allow the maximum amount of room to hold the work file. There must be 1.0 times the space of the original file available on the sort disk or an out of space error will occur and the sort will be aborted.

The sort program will also clean up the file by sifting out previously deleted (inactive) records.

The maximum file size is 4000K bytes (4 megabytes) for the **SORT** program. However, a file to be sorted cannot extend beyond the limitation of disk drive storage. Larger files may be sorted by using the select command to break the file into smaller sections and then sort the sections. Use the **APPEND** or **COMBINE** commands to merge sections.

STAX(tistics)

PURPOSE: View or print statistics of data item values

SYNTAX: **STAX database BY d1, d2,... d32 [option]**

OPTIONS: **[A]** means display only the averages
[C] means display only the count
[M] means display only the minimum and maximums
[T] means display only the totals
[P] means send the output to the printer

If no option is specified all statistics are displayed on the console screen

EXAMPLES:
```
STAX EMPLOYEES BY AGE
STAX EMPLOYEES BY WAGE AND AGE [P]
STAX EMPLOYEES WAGE [T] [M] [P]
```

USAGE:

As an example, one can determine the sales expenses and costs for a data base called ACCOUNTS. The user would type the command:

```
STAX ACCOUNTS BY SALES AND COSTS
```

This would give the following output:

	SALES	COSTS
Minimum	3.00	4.00
Maximum	55.00	220.00
Total	10345.00	5435.00
Average	41.38	21.74

Record Count = 250

A maximum of 32 data items can be indicated for the **STAX** command. Alpha or alphanumeric fields cannot be used for the **STAX** command. An overflow can occur if the total or calculation of the average produces a number greater than 4 bytes. A series of asterisks are printed in this case.

SYSTEM

PURPOSE: Exit from DBMS

SYNTAX: **SYS** or **SYSTEM**

EXAMPLES: **SYS**
SYSTEM

USAGE:

The user can exit from the DBMS by using the **SYS** command, at which time all the addresses and memory space will be restored to its original state.

TABULATE

PURPOSE: Summarize and print specified data items

SYNTAX: **TABULATE database BY d1, d2,...**
{AND COMPUTE statistic d3, d4,... ,d32} [option]

Where **statistic** can be one of the following words:
TOTAL (TOT, ADD, SUM), MIN, MAX, AVERAGE (AVE, AVG)

The word compute may be addreviated with "@"

OPTIONS: **[A]** Means accumulate subtotals
[P] Means print results on printer
[S] Means create result set

RESULT SET: **R$.DAT,R$.DEF,R$.FRM** (if option chosen).

EXAMPLES:
```
TABULATE  CLIENTS BY STATE
TABULATE  CLIENTS BY STATE AND COMPUTE TOTAL WAGE [P]
TABULATE  CLIENTS BY STATE @ TOT WAGE, AVERAGE HOURS
```

USAGE:

The database must be sorted by the tabulating key. i.e., all the fields named before the COMPUTE must be in sort. The tabulating key can be in either ascending sequence or descending sequence. If it is not, an error message is printed and the program is aborted. The database should be sorted in this case (see **SORT** command).

The output generated is a statistic summary either count or totals, averages, min, and max values. In the first example, the output will be shown as:

STATE	{COUNT}
INDIANA	20
MICHIGAN	25
Total	45

In the third example, the output will be shown without the count as:

STATE	WAGE {Subtotal}	HOURS {Average}
INDIANA	2301.03	38
MICHIGAN	2698.07	49
Total	4999.10	

Note: Only totals are provided for count and subtotal columns. The column spacing is set by adding one to the number of underscores or characters in the label found in the form for that field, which ever is greater. (The extra 1 is added in order to maintain at least one space column seperation.) If the data item is in the compute predicate, and if the number of underscores or label length is not at least twelve, then 12 + 1 =13 printing positions are assigned. The dummy field COUNT will always get 12 + 1 = 13 printing positions.

TITLE

PURPOSE: Print report headings.

SYNTAX: **TITLE {'text', or "text", PAGE, DATE, TOP, SKIP, LINE, nn}**

Where clause consists of one or more of the characters or words defined below:

P or **PAGE** means print page n, where n is the current page
L or **LINE** means carriage return-line feed
D or **DATE** means prints current signon date
T or **TOP** means do a form feed after nn lines
S or **SKIP** means skip a line, i.e. insert a blank line
nn = number of lines between titles (default is 66) (2-99)

EXAMPLES:
```
TITLE 'FINANCIAL REPORT FOR ABC COMPANY AS OF', DATE
TITLE LINE,LINE, 'TWO LINE',LINE,' FOR A TWO LINE TITLE
TITLE TOP,44
TITLE "The above title does a form feed every 44 lines'
TITLE
```

USAGE:

The **TITLE** command prints the title after **nn** lines have been printed. The lines can be set between 2 and 99 (default is 66) The page number and signon date can be printed on each page of the report by including the **P** or **PAGE**, and **D** or **DATE** words. These values are printed wherever these words appear in the title line.

EXAMPLE: **TITLE 'THIS IS',PAGE,' AND TODAY IS',DATE**

will print

THIS IS PAGE 1 AND TODAY IS 4/23/83

The **SKIP** or **S** option can be used to insert blank lines every **nn** lines.

EXAMPLE: **TITLE SKIP,2**

Will insert a blank line every 2 lines, therefore double spacing the printed output.

Notes:

1. Each **TITLE** command clears the previous **TITLE** line.
2. The **TITLE** command executes from left to right; therefore, **TITLE "HEADING",TOP,48** will print the heading, then go to the top of the next page. **TITLE,TOP,48,"HEADING"** will go to the top of the next page, then print the heading.
3. The **SKIP** and **TOP** options cannot be used in the same TITLE command.

UPDATE

PURPOSE: Change data item values in a database meeting specified conditions.

SYNTAX: **UPDATE database {WHERE conditon}**

EXAMPLES:
```
UPDATE CLIENTS WHERE NAME IS SMI*
UPDATE CLIENTS NAME IS 'DOE JANE'
UPDATE DEFSET.DEF ST FIELD IS 1 2 4
UPDATE DATA.DIC WHERE TITLE IS CLIENTS
UPDATE CLIENTS.DIR
UPDATE CLIENTS WHERE ALL
```

WHERE condition:
A WHERE condition is one or more comparisons joined by AND or OR. Both AND and OR may not be used together in one command. A comparison is a phrase comparing a data item with one or more values:

data-item comparison-verb value {,value...}

EXAMPLE: **STATE IS MI,IN**

A comparison-verb compares the value of the data item with one or more values to determine if the WHERE condition will be satisfied.

Comparison-verbs are:

Verb	Synonym
=	EQ, IS, ARE, EQUALS
<>	NE, IS NOT, ARE NOT
<	LT, IS LT, IS NOT GE, IS <
>	GT, IS GT, IS NOT LE, IS >
<=	LE, IS LE, IS NOT GT, IS <=
>=	GE, IS GE, IS NOT LT, IS >=
*	ALL, WHERE ALL

Values may be constants or another data-item value. Data item values are specified by preceeding the data item name with the "at sign" (@).

EXAMPLE: **SHIP.ADDRESS IS @BILL.ADDRESS**

The commas separating the values are the same as typing "OR data item IS".

EXAMPLE: **UPDATE CLIENTS WHERE STATE IS MI,IN**
is the same as
UPDATE CLIENTS WHERE STATE IS MI OR STATE IS IN

USAGE:

The command is very similar to the **DISPLAY** command with the exception that the user can make changes to records in the database by typing the Revise function key. When desired data items have been changed, the user may exit from the screen by entering the EXIT function key. At this point the user may continue or end updating by entering a function key as prompted at the bottom of the screen. After each record is updated the changed record is recorded into the file.

WRITE

PURPOSE: Transfer records from a database to an ASCII sequential file.

SYNTAX: **WRITE database {output - filename} [option]**

OPTION:

None Fixed length, continuous ASCII stream-no separators between data items. Numeric data items are right justified.

[A] Abridged - variable length - data items separated by tab (ASCII 09), records separated by <C/R>. All trailing blanks are truncated.

[B] Basic - variable - data items enclosed in quotes (") and separated by commas, records separated by <C/R> - <L/F>. Numeric fields will not be surrounded by quotes.

[R] RPG - Fixed length - no separators between data items, records are separated by <C/R> - <L/F>. Numeric fields are right justified.

[M] Mail List - Variable length - data items separated by <C/R> <L/F>, records separated by an additional <C/R> <L/F>.

[G] Graphics - Fixed length - separators are double spaces between data items, records are separated by <C/R> - <L/F>. Numeric fields are right justified.

EXAMPLES: **WRITE CLIENTS**
WRITE CLIENTS REPORT
WRITE CLIENTS REPORT1 [B]

Note: All ASCII files will be terminated with control-Z (**ASCII 01AH**), indicating the end of file.

Note: If no output-filename is specified, output will be written to **R$.ASC**

USAGE:

The **WRITE** command is useful for converting a SERIES 20 Database into an **ASCII** data file that can be read by other application programs or report generators (e.g., applications written in a host language such as Basic, Fortran, etc).

Four different types of output formats may be generated by chosing the appropriate option **[option]** as described above.

The B format is compatible with Microsoft's BASIC as well as CBASIC. All data items should be input as string variables and numeric items converted to the appropriate storage type.

APPENDIX B

CONDOR SERIES 20 SPECIFICATIONS

DATABASE, RECORD AND DATA ITEM SIZES*

Database size:	32,767 records maximum per database
Record size:	2 to 1024 bytes per record. In each record, the first byte is automatically assigned as a status byte.
Data items:	1 to 127 data items per record.
A or **AN Data:**	1 to 127 bytes per data item
N or **$ Data:**	1 to 10 digits (stored as a 1 to 4 byte integer.
J Data:	Julian Date, 3 bytes per field.

* Note: maximum size file you can back-up is 248K bytes on 5 1/4" disks and 1.1 M bytes in the 8" disks, unless you have a utility to copy parts of a file on one disk.

DATA TYPES

AN - Alphanumeric:	a to z, A to Z, 0 to 9, symbols
A - Alphabetic:	a to z, A to Z, " " (space), "'" (single quote), "." (period), "-" (hyphen)

DATA TYPES continued

N - Numeric:	+/-	127 requires 1 byte
	+/-	32767 requires 2 bytes
	+/-	8388607 requires 3 bytes
	+/-	2148373647 requires 4 bytes
$ - Dollars:	+/-	$1.27 requires 1 byte
	+/-	$327.67 requires 2 bytes
	+/-	$83886.07 requires 3 bytes
	+/-	$21483736.47 requires 4 bytes
J - Date:	Format **MM/DD/YY** for input/output; stored as 3 bytes	
R:	Required Entry	

DATABASE, FILE AND DATA ITEM NAMES

Database Name: 1 to 8 characters

Data Item Name: 1 to 15 characters

File Names:

xxxxxxxx.FRM	--	Form File
xxxxxxxx.DEF	--	Data Item Definitions
xxxxxxxx.DAT	--	Data File
xxxxxxxx.CMD	--	Command Procedure
xxxxxxxx.HEP	--	HELP Screen File
xxxxxxxx.DBM	--	Series 20 Command
DATA.DIC	--	Data Dictionary

Database and Data Item names should avoid the following characters:

* (asterisk)	[(left square bracket)
? (question mark)	] (right square bracket)
} (right bracket)	" (double quote)
< (less than)	' (single quote)
> (greater than)	

These characters have special significance in some Series 20 commands, and using them in database or data item names may cause ambiguity in interpreting these commands. If a database or data item name must contain one of these characters, the name should be enclosed in quotation marks every time it is used in a Series 20 command. It is also best if names do not contain embedded blanks. Again, if embedded blanks must be used, the name must be enclosed in quotation marks to avoid having it interpreted as two separate names.

RESERVED WORDS

In Condor Series 20 commands, certain words have special meaning. For example, the word **ADD** in the **POST** command instructs the computer to add two data values. If the word **ADD** is used as a data item name, it is highly probable that the command line will be misinterpreted, causing an erroneous calculation to be performed. For this reason, words having special meaning to Series 20 are designated as Reserved Words, and their use is restricted; that is, data item names should not be Reserved Words.

Condor Series 20 Reserved Words are:

{ (Compute)
ADD
AND
ARE
AVERAGE
AVE
AVG
BY
COMPUTE
COMP
EQ (Equal)
EQUAL
GE (Greater or Equal)
GT (Greater Than)
IS
LE (Less or Equal)
LT (Less Than)
MATCHING
MAX
MIN
NE (Not Equal)
NOT
OR
REPLACE
REP (Replace)
STAX
ST (Such That)
SUBTOTAL
SUBTOT (Subtotal)
SUBTRACT
SUB (Subtract)
SUM
TOTAL
TOT
USING
WHERE
WHOSE

APPENDIX C

CONDOR SERIES 20 FILE FORMATS

The **READ** and **WRITE** commands are the interface between Condor 20 and the "outside world" of ASCII CP/M files. These files in turn can be processed or created by other HP Series 100 applications, such as Series 100/Word, Wordstar/100, Series 100/Graphics, Link/125, and user-written applications. The **WRITE** command supports six different file formats, selected by the **[option]** in the command. These formats are described below:

DEFAULT FORMAT — No Option

* Continuous character stream format
* Each data item occupies the same number of bytes in the file as the length of its data entry field on the database form
* Numeric data items are right justified
* No separator characters between data items
* No separator characters between records

Default format may be most appropriate for stream-oriented file processing by applications written in PASCAL or ASSEMBLY language. It is also a highly efficient format for backing up database contents, or for temporary storage during database reorganization.

ABRIDGED FORMAT — [A] Option

* Variable-length record format
* Data items occupy only as many bytes as required to hold the data; leading and trailing blanks are truncated
* Data items separated by TAB characters (HEX 09)
* Records separated by <CR> character (HEX 0D)

Abridged format may be most appropriate for applications processing where the application program does not know the length of each data item in advance. The file can be searched for the delimiter characters (TAB and <CR>) to determine the location of individual data items and records within the file.

BASIC FORMAT — [B] Option

* Variable-length record format
* Data items occupy only as many bytes as required to contain the data; leading and trailing blanks are truncated
* Data items are enclosed in quotation marks (")
* Data items separated by commas
* Records separated by <CR> <LF>

Basic format is used for data exchange between databases and applications programs written in Basic and between Mailmerge/100 for Wordstar/100. The format used on the file corresponds exactly to the format used by Basic's file I/O commands, allowing direct processing of the file by Basic programs, using file I/O statements. Mailmerge/100 can automatically read this format to couple with a document designed in Wordstar/100.

GRAPHICS FORMAT — [G] Option

* Fixed-length record format
* Each data item occupies as many bytes on the file as the length of its data entry field on the database form
* Numeric data is right justified
* Data items are separated by two spaces
* Records are separated by a <CR> <LF> sequence

Graphics format is used to incorporate data extracted from databases into graphic data files, or word processor documents. The data file resembles the body of a row/column report, with records appearing in successive records of the file, and data item values aligned beneath one another in subsequent records, to form columns. The file can be read directly by Series 100/Graphics, Series 100/Word, Wordstar/100, or the CP/M text editor, displayed using the CP/M TYPE command, etc.

MAIL LIST FORMAT — [M] Option

* Variable-length record format
* Each data item occupies as many bytes on the file as the length of its data entry field on the database form
* Numeric data is right justified
* Data items are separated by a <CR> <LF> sequence
* Records are separated by an additional <CR> <LF> sequence

Mail list format is used to pass data from databases into Series 100/Word's

Mail Merge facility. The resulting file is in exactly the format expected for input as the "mailing list" file in the Mail Merge macro. The data items in the database become, in sequence, the variable parameters used in the form letter.

RPG FORMAT — [R] Command

* Fixed length records structure
* Each data item occupies as many bytes on the file as the length of its data entry field on the database form
* Numeric data is right justified
* No separator characters between data items
* Records are separated by a <CR> <LF> sequence

RPG format is used to exchange data between databases and applications written in languages that use a fixed-length record structure for normal file I/O. Languages such as COBOL, RPG, and FORTRAN can easily process data files using this format.

APPENDIX D

EXAMPLE DATABASES USED IN THIS MANUAL

```
Attribute summary of Data Base CARDFILE

1.NAME: AN,20,0,20,"ABC                "
2.TITLE: AN,20,0,20,"                   "
3.FIRM: AN,30,0,30,"                   "
4.STREET: AN,30,0,30,"                    "
5.CITY: AN,20,0,20,"                   "
6.STATE: AN,2,0,2,"                   "
7.ZIPCODE:  N,3,-8388607,8388607,"                    "
8.BUS-PHONE: AN,10,0,10,"                    "
9.HOME-PHONE: AN,10,0,10,"                     "
10.REV-DATE:  J,3,01/01/00,12/31/99,"                       "

Record Size (Bytes) = 149
Total Records = 18
```

```
**********************************************************************
**                                                                  **
**            ***** HP 125 CARDFILE DATABASE *****                  **
**                                                                  **
**    [NAME]:        ____________________                           **
**    [TITLE]:       ____________________                           **
**    [FIRM]:        ______________________________                 **
**                                                                  **
**    [STREET]:      ______________________________                 **
**    [CITY]:        ____________________                           **
**    [STATE]:       __      [ZIPCODE]:    _____                   **
**                                                                  **
**                                                                  **
**    [BUS-PHONE]:   __________                                     **
**    [HOME-PHONE]:  __________      [REV-DATE]:  ________          **
**                                                                  **
**********************************************************************
```

```
Attribute summary of Data Base ORDERS

1.ORDER.NO: AN,8,8,8,"                    "
2.CUST.NO: AN,6,0,6,"                   "
3.PRODUCT.NO: AN,6,0,6,"                    "
4.QUANTITY:  N,2,0,32767,"                    "
5.UNIT.PRICE:  $,4,.00,5000.00,"                      "
6.ORDER.TYPE:  N,1,0,99,"01                   "
7.DATE:  J,3,01/01/00,12/31/99,"                      "
8.REGION: AN,4,0,4,"                  "
9.TAX:  $,4,.00,5000.00,"                     "
10.AMOUNT:  $,4,.00,10000.00,"                        "

Record Size (Bytes) = 43
Total Records = 5
```

```
**********************************************************
**                                                      **
**   ORDER FORM:     (Enter Data Below:)                **
**                                                      **
**********************************************************
**                                                      **
**          Enter [ORDER.NO]:          ________         **
**          Enter [CUST.NO]:             ______         **
**                                                      **
**          Enter [PRODUCT.NO]:          ______         **
**          Enter [QUANTITY]:               ___ units   **
**          Enter [UNIT.PRICE]:       $ _______         **
**          Enter [ORDER.TYPE]:             __          **
**          Enter [DATE]:              ________         **
**          Enter [REGION]:               ____          **
**                                                      **
**********************************************************
**                                                      **
**          Computed [TAX]:             $ _______       **
**                                                      **
**          Computed [AMOUNT]:        $ _________      **
**                                                      **
**********************************************************
```

```
Attribute summary of Data Base REGIONS

1.REGION: AN,4,0,4,"                    "
2.REGION.NAME: AN,12,0,12,"                      "
3.NAME: AN,20,0,20,"                    "
4.ACTUAL.MTD:  $,4,-21474836.47,21474836.47,"                      "
5.ACTUAL.YTD:  $,4,-21474836.47,21474836.47,"                      "
6.QUOTA.MTD:  $,4,-21474836.47,21474836.47,"                     "
7.QUOTA.YTD:  $,4,-21474836.47,21474836.47,"                     "
8.QUOTA%.MTD:  $,3,-83886.07,83886.07,"                  "
9.QUOTA%.YTD:  $,3,-83886.07,83886.07,"                  "
10.AMOUNT:  $,4,-21474836.47,21474836.47,"                     "

Record Size (Bytes) = 63
Total Records = 12
```

```
**********************************************************************
**                                                                  **
**              ***** REGIONS DATABASE ****                         **
**                                                                  **
**  [REGION]:       ____                                            **
**  [REGION.NAME]: ____________                                     **
**  [NAME]:         ____________________                            **
**                                                                  **
**********************************************************************
**                                                                  **
**  [ACTUAL.MTD]: __________   [ACTUAL.YTD]: __________             **
**  [QUOTA.MTD]:  __________   [QUOTA.YTD]:  __________             **
**  [QUOTA%.MTD]:     ______   [QUOTA%.YTD]:     ______             **
**                                                                  **
**  [AMOUNT]:     __________                                        **
**                                                                  **
**********************************************************************
```

```
Attribute summary of Data Base CUSTOMER

1.CUST.NO: AN,6,0,6,"                    "
2.FIRM: AN,30,0,30,"                    "
3.NAME: AN,20,0,20,"                    "
4.STREET: AN,30,0,30,"                    "
5.CITY: AN,20,0,20,"                    "
6.STATE: AN,2,0,2,"                    "
7.ZIP.CODE: AN,5,0,5,"                    "
8.REGION:  A,10,0,10,"                    "
9.OFFICE:  A,10,0,10,"                    "
10.SALES.REP:  A,20,0,20,"                    "
11.AMOUNT:  $,4,-21474836.47,21474836.47,"                    "
12.REV.DATE:  J,3,01/01/00,12/31/99,"                    "

Record Size (Bytes) = 161
Total Records = 0
```

```
**********************************************************************
**                                                                  **
**                  ***** CUSTOMER DATABASE *****                   **
**                                                                  **
**    [CUST.NO]:      ______                                        **
**    [FIRM]:         ------------------------------                **
**    [NAME]:         --------------------                          **
**                                                                  **
**    [STREET]:       ------------------------------                **
**    [CITY]:         --------------------                          **
**    [STATE]:        __      [ZIP.CODE]:    _____                  **
**                                                                  **
**                                                                  **
**    [REGION]:       ----------                                    **
**    [OFFICE]:       ----------                                    **
**    [SALES.REP]:    --------------------                          **
**    [AMOUNT]:       ------------                                  **
**    [REV.DATE]:     --------                                      **
**                                                                  **
**********************************************************************
```

```
Attribute summary of Data Base PRODUCTS

1.PRODUCT.NO: AN,6,0,6,"                    "
2.PRODUCT.LINE: AN,3,0,3,"                    "
3.DESCRIPTION: AN,20,0,20,"                    "
4.UNIT.PRICE:  $,4,-21474836.47,21474836.47,"                    "
5.COST:  $,4,-21474836.47,21474836.47,"                    "
6.QTY.ON.HAND:  N,4,-2147483647,2147483647,"                    "
7.INV.VALUE:  $,4,-21474836.47,21474836.47,"                    "

Record Size (Bytes) = 46
Total Records = 0
```

```
**************************************************************
**                                                          **
**            ***** PRODUCTS DATABASE *****                 **
**                                                          **
**        [PRODUCT.NO]:                      ______         **
**        [PRODUCT.LINE]:                       ___         **
**        [DESCRIPTION]:    ____________________            **
**                                                          **
**        [UNIT.PRICE]:                 $ ________         **
**                                                          **
**        [COST]:                       $ ________         **
**                                                          **
**        [QTY.ON.HAND]:                  ________         **
**                                                          **
**        [INV.VALUE]:                $ ___________         **
**                                                          **
**************************************************************
```

APPENDIX E

CONDOR SERIES 20 COMMAND QUICK REFERENCE

COMMAND	SYNTAX
ABORT	**:ABORT**
APPEND	**:APPEND** database1 database2
CHANGE*	**:CHANGE** database {ST d1 is value1 **AND** d2 is value2...}
COMBINE*	**:COMBINE** database1 database2
COMPARE	**:COMPARE** database1 database2 {NOT} MATCHING d1,d2,...d32
COMPUTE	**:COMPUTE** database ST dr = d1 opr d2...d31
COPY	**:COPY** destination-database/file = source-database/file {OK}
	:COPY drive: = source-database/file {OK}
DATE	**:DATE** {mm/dd/yy}
DBMS	**:DBMS**
DEFINE	**:DEFINE** database {*}
DELETE*	**:DELETE** database {WHERE condition}
DESTROY	**:DESTROY** database/file {OK}
DIC	**:DIC** {drv:}
DIR	**:DIR** {drv:} {filename.ext}
DISPLAY*	**:DISPLAY** database {WHERE condition}
EDIT	**:EDIT** filename {.CMD}
EMPTY	**:EMPTY** database {OK}
ENTER	**:ENTER** database [option]
FORMAT	**:FORMAT** formfile {.FRM/.HLP}
HELP	**:HELP** {HELPFILE} {.HLP}
INDEX**	**:INDEX** database [option]
JOIN*	**:JOIN** database1 database2 MATCHING d1,d2,...d32 [option]
LIST	**:LIST** database {BY d1,d2,... {**AND COMPUTE** statistic d3,d4...}}[option]
LOG	**:LOGDISK** or **LOG DISK** {drive.disk name}
POST	**:POST** database1 database2 MATCHING d1,d2,...{opr d3,d4...}
PRINT	**:PRINT** database {BY d1,d2,...{**AND COMPUTE** statistic d3,d4...}}[option]
PROJECT*	**:PROJECT** database BY d1,d2,...d32
READ	**:READ** database {filename.ASC}
RENAME	**:RENAME** newname oldname {OK}
REORG	**:REORG** database {formfile.FRM} [option]
REPORT**	**:REPORT** database {WHERE condition} [option]
RESTART	**:RESTART** {drv:}
RUN	**:RUN** procedure {.CMD} {param1 param2...param9}
SAVE	**:SAVE** database/file {OK}
SELECT	**:SELECT** database {WHERE condition}
SET	**:SET** {function status}

* These commands are included only in Condor 20-2 and 20-3.
** These commands are included only in Condor 20-3.

SORT	:**SORT** database BY d1,d2,...d32
STAX	:**STAX** database BY d1,d2,...d32 [option]
SYSTEM	:**SYSTEM** or **SYS**
TABULATE	:**TABULATE** database {BY d1,d2,...{AND COMPUTE statistic d3,d4...}}[option]
TITLE	:**TITLE** {'text' or "text", PAGE, DATE, SKIP, LINE, TOP, nn}
UPDATE	:**UPDATE** database {WHERE condition}
WRITE	:**WRITE** database {filename.ASC} [option]

APPENDIX F

REPORT WRITER EXAMPLE

This appendix is supplied to step you through a complete example of creating a report specification using the short method. Once you have created a report specification using the short method, you will be able to use the long method as well. The example contained is the same report found in Chapter 12, the Warehouse Report (Fig. F-1).

Before beginning, you will want to:

1. Place the Condor 20-3 in drive A, and a scratch disc in drive B. (If another drive is used in place of B, substitute that letter throughout the example.)

2. Enter Condor 20-3 by pressing the **"Condor 20-3"** softkey.

3. On drive B, create the WAREHSE database and database form as found in Fig. F-2. Use the command **DEFINE WAREHSE** to perform that. You may defer entering records into the database until a later time. (If you need assistance in creating the database and form, Chapter 3 displays how to create a database.)

WAREHOUSE UPDATE REPORT

Item Number	Item Description	Prev. On Hand	Shipped	Received	Current On Hand	Unit Price	Total Value
DIVISION 1	WAREHOUSE 1						
1	SHOVELS	25	7	4	22	5.80	127.60
2	TROWELS	20	100	60	-20	3.33	-66.60
5	RED STONES	87	91	23	19	4.71	89.49
7	YELLOW STONES	97	7	14	104	4.58	476.32
8	BLACK STONES	50	19	23	54	4.88	263.52
			WAREHOUSE 1		TOTAL		890.33
DIVISION 1	WAREHOUSE 2						
1	SHOVELS	27	9	14	32	5.80	185.60
2	TROWELS	15	6	0	9	3.33	29.97
5	RED STONES	93	75	12	30	4.71	141.30
7	YELLOW STONES	47	14	11	44	4.58	201.52
8	BLACK STONES	3	8	31	26	4.88	126.88
			WAREHOUSE 2		TOTAL		685.27
			DIVISION 1	TOTAL			1575.60
DIVISION 2	WAREHOUSE 1						
11	BLUE CEMENT	12	10	2	4	3.40	13.60
12	YELLOW CEMENT	30	10	5	25	4.50	112.50
8	BLACK STONES	50	1	34	83	4.50	373.50
			WAREHOUSE 1		TOTAL		499.60
DIVISION 2	WAREHOUSE 2						
11	BLUE CEMENT	40	12	3	31	3.40	105.40
			WAREHOUSE 2		TOTAL		105.40
			DIVISION 2	TOTAL			605.00
			GRAND TOTAL				2180.60

(Fig. F-1)

SCREEN FORMAT OF WAREHOUSE DATABASE

```
[DIVNBR] _                [WHSENBR] _
 (division number)         (warehouse number)

[NO] __                   [ITEM] _____________
 (item number)             (item description)

[PREQTY] __               [SHIPPED] ___          [RCVED] __
 (previous quantity)       (number shipped)       (number received)

[PRICE] ______
```

****

Number of underscores in screen format

[DIVNBR]	1
[WHSENBR]	1
[NO]	2
[ITEM]	13
[PREQTY]	2
[SHIPPED]	3
[RCVED]	2
[PRICE]	6

****

Attribute summary of Data Base WAREHSE

```
1.DIVNBR: N,1,1,5,"        "
2.WHSENBR: N,1,Ø,5,"        "
3.NO: N,1,Ø,127,"         "
4.ITEM: AN,13,Ø,13,"        "
5.PREQTY: N,1,Ø,127,"         "
6.SHIPPED: N,1,Ø,127,"          "
7.RCVED: N,1,Ø,127,"        "
8.PRICE: $,3,.ØØ,327.67,"             "

Record Size (Bytes) = 23
Total Records = Ø
```

(Fig. F-2)

CREATING THE WAREHOUSE REPORT

Conventions used in the example that follows:

1. All information to be supplied by you are highlighted in bold characters.
2. Report Writer prompts are printed without bold characters.
3. Instructional comments are displayed with bold characters surrounded by parentheses.

B>>REPORT WAREHSE

```
                CONDOR SERIES 2Ø RDBMS REPORT WRITER
                        Version 1.xx**xx

                        Choose option

        Create New Report Specification          (C)
        Describe existing Report Specification   (D)
        Revise existing Report Specification     (R)
        Print or Display Database Report         (P)

        Enter option or End <C/R>:C
```

Busy

Do you wish to use the short method (Y/N)?**Y**

Do you with to create a new report format (Y/N)?**Y**

(The screen will clear. Duplicate the form in Fig. F-3 and press the End softkey.)

```
                        WAREHOUSE UPDATE REPORT

                                                              PAGE   $PAGE

Item     Item          Prev.                                  Current  Unit     Total
Number   Description   On Hand   Shipped     Received         On Hand  Price    Value
DIVISION [DIVNBR] WAREHOUSE [WHSENBR]
  [NO]  [ITEM]          [PREQTY] [SHIPPED]    [RCVED]        [CURQTY] [PRICE]  [VALUE]

                                    WAREHOUSE [WHSENBR] TOTAL                 [VALUE]
                              DIVISION [DIVNBR] TOTAL                         [VALUE]
                         GRAND TOTAL                                          [VALUE]
```

(Fig. F-3)

Do you wish to use the short method (Y/N)?**Y**

(Report Writer gives you a second chance to specify or not specify the short method after you have created a new report format or revised an existing one.)

>1. WAREHOUSE UPDATE REPORT

Describe when this line is to be printed:

```
First Page only          (FP)
Last Page only           (LP)
Every Page Heading       (PH)
Once Every Line          (EL)
On Break-Before Section  (BS)
On Break-After Section   (AS)

Enter code              [ PH ]: <RETURN>
```

(The above line is a page heading. It is to be printed at the top of every page.)

OK (Y/N)? **Y**

(If you incorrectly specified when the line is to be printed, you may enter "N" to any of the "OK (Y/N)" prompts and alter your response.)

```
>2.                                    PAGE    $PAGE

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ PH ]: <RETURN>

 OK (Y/N)? Y


>3.Item  Item      Prev.          Current  Unit     Total

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ PH ]: <RETURN>

 OK (Y/N)? Y


>4.Number Description On Hand Shipped  Received On Hand Price
Value

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ PH ]: <RETURN>


 OK (Y/N)? Y
```

```
>5.DIVISION [DIVNBR] WAREHOUSE [WHSENBR]

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ PH ]:BS   <RETURN>

        (The above line is to be printed before every section.)

 OK (Y/N)? Y

Which data-item is to cause a section break to occur:DIVNBR
Enter any other or <C/R>:WHSENBR
Enter any other or <C/R>:  <RETURN>


>6. [NO] [ITEM] [PREQTY] [SHIPPED] [RCVED] [CURQTY] [PRICE]
[VALUE]

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ BS ]:EL  <RETURN>

 OK (Y/N)? Y


Item 6:[CURQTY]

Describe Type of Variable:

  Numeric    (N)
  Dollar     ($)
  Date       (J)

Enter Code [ N ]: <RETURN>
CURQTY=PREQTY - SHIPPED + RCVED
```

```
Define printing width - Enter column size [ 6 ]:3
```

(Above is the formula value for CURQTY. CURQTY is the Current Quantity on hand. If you press <RETURN> without supplying the formula for this or any other computed item, Report Writer asks you to enter a calculation expression.)

```
Item 8:[VALUE]

Describe Type of Variable:

  Numeric    (N)
  Dollar     ($)
  Date       (J)

Enter Code [ N ]:$  <RETURN>
VALUE=CURQTY * PRICE
Define printing width - Enter column size [ 5 ]:7
```

(VALUE is the dollar value of the current inventory.)

```
>7.   WAREHOUSE [WHSENBR] TOTAL          [VALUE]

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ BS ]:AS  <RETURN>

 OK (Y/N)? Y

Which data-item is to cause a section break to occur:WHSENBR
Enter any other or <C/R>: <RETURN>

Item 2:[WHSENBR]

Describe Value to Print:

  Average value      (AVG)
  Maximum value      (MAX)
  Minimum value      (MIN)
  Subtotal           (SUB)
  Accumulated total  (TOT)
  Last value read    (VAL)

  Enter code        [ SUB ]:VAL
```

```
Item 4:[VALUE]
Define printing width - Enter column size [ 5 ]:7

Describe Value to Print:

  Average value      (AVG)
  Maximum value      (MAX)
  Minimum value      (MIN)
  Subtotal           (SUB)
  Accumulated total  (TOT)
  Last value read    (VAL)

  Enter code        [ SUB ]: <RETURN>


>8.   DIVISION [DIVNBR] TOTAL          [VALUE]

Describe when this line is to be printed:

   First Page only          (FP)
   Last Page only           (LP)
   Every Page Heading       (PH)
   Once Every Line          (EL)
   On Break-Before Section  (BS)
   On Break-After Section   (AS)

   Enter code              [ AS ]:<RETURN>

 OK (Y/N)? Y

Which data-item is to cause a section break to occur:DIVNBR
Enter any other or <C/R>: <RETURN>

Item 2:[DIVNBR]

Describe Value to Print:

  Average value      (AVG)
  Maximum value      (MAX)
  Minimum value      (MIN)
  Subtotal           (SUB)
  Accumulated total  (TOT)
  Last value read    (VAL)

  Enter code        [ SUB ]:VAL
```

```
Item 4:[VALUE]
Define printing width - Enter column size [ 5 ]:7

Describe Value to Print:

  Average value      (AVG)
  Maximum value      (MAX)
  Minimum value      (MIN)
  Subtotal           (SUB)
  Accumulated total  (TOT)
  Last value read    (VAL)

  Enter code        [ SUB ]: <RETURN>


>9.   GRAND TOTAL              [VALUE]

Describe when this line is to be printed:

   First Page only           (FP)
   Last Page only            (LP)
   Every Page Heading        (PH)
   Once Every Line           (EL)
   On Break-Before Section   (BS)
   On Break-After Section    (AS)

   Enter code               [ EL ]:LP  <RETURN>

 OK (Y/N)? Y

Item 3:[VALUE]
Define printing width - Enter column size [ 5 ]:7

Describe Value to Print:

  Average value      (AVG)
  Maximum value      (MAX)
  Minimum value      (MIN)
  Subtotal           (SUB)
  Accumulated total  (TOT)
  Last value read    (VAL)

  Enter code        [ TOT ]: <RETURN>


Enter name of output report [WAREHSE]:<RETURN>
Busy
```

Do you wish to save the report form created (Y/N)?**Y**

Enter filename of report form or End <C/R>:**WHSERPT.FRM**

(This is your report model form. It is advised that you save it, in case the report must be respecified. Always, the name under which you save the form MUST NOT be the same as the database form. This would destroy the database form used for data entry and updating.)

****** YOU HAVE FINISHED CREATING THE REPORT SPECIFICATION ******

Now, anytime you wish to print a report of the data in your database, simply select the [P] (for print) option in the Report Writer menu.

Since the short method doesn't ask you to specify line spacing for the report, you probably want to revise the specification so that it appears just like the report in Fig. F-1. The report specification should end up looking like the description which immediately follows in this appendix.

Also, before you print, you should do two things:

1. Make sure you have data in your database.

2. Sort your database by the control items which are used in your report specification to designate section breaks. If your report appears to have incorrect section changes, this is the probable cause.

Report Specification for WAREHSE

Page Format

Physl Lines	Left Margin	Right Margin	Top Margin	Bottom Margin
66	1	80	1	60

>1.WAREHOUSE UPDATE REPORT

Print on
(PH) - Every Page Heading

Line # Before	Line # After	Space Before	Space After	Look Ahead
1	0	0	0	0

Item	Type	Format	Column	Length	Print
WAREHOUSE	TEXT	SD	24	9	VAL
UPDATE	TEXT	SD	34	6	VAL
REPORT	TEXT	SD	41	6	VAL

>2.PAGE $PAGE

Print on
(PH) - Every Page Heading

Line # Before	Line # After	Space Before	Space After	Look Ahead
0	0	1	0	0

Item	Type	Format	Column	Length	Print
PAGE	TEXT	SD	40	4	VAL
$PAGE	TEXT	SD	47	5	VAL

>3.Item Item Prev. Current Unit Total

Print on
(PH) - Every Page Heading

Line # Before	Line # After	Space Before	Space After	Look Ahead
0	0	1	0	0

Item	Type	Format	Column	Length	Print
Item	TEXT	SD	1	4	VAL
Item	TEXT	SD	9	4	VAL
Prev.	TEXT	SD	22	5	VAL
Current	TEXT	SD	52	7	VAL
Unit	TEXT	SD	63	4	VAL
Total	TEXT	SD	71	5	VAL

>4.Number Description On Hand Shipped Received On Hand Price Value

Print on
(PH) - Every Page Heading

Line # Before	Line # After	Space Before	Space After	Look Ahead
0	0	0	0	0

Item	Type	Format	Column	Length	Print
Number	TEXT	SD	1	6	VAL
Description	TEXT	SD	9	11	VAL
On	TEXT	SD	22	2	VAL
Hand	TEXT	SD	25	4	VAL
Shipped	TEXT	SD	31	7	VAL
Received	TEXT	SD	42	8	VAL
On	TEXT	SD	52	2	VAL
Hand	TEXT	SD	55	4	VAL
Price	TEXT	SD	63	5	VAL
Value	TEXT	SD	71	5	VAL

>5.DIVISION [DIVNBR] WAREHOUSE [WHSENBR]

Print on
(BS) - Break Before Section
Control Items
DIVNBR WHSENBR

Line # Before	Line # After	Space Before	Space After	Look Ahead
0	0	1	1	0

Item	Type	Format	Column	Length	Print
DIVISION	TEXT	SD	1	8	VAL
DIVNBR	DATA	SD	10	1	VAL
WAREHOUSE	TEXT	SD	19	9	VAL
WHSENBR	DATA	SD	29	1	VAL

>6.[NO] [ITEM] [PREQTY] [SHIPPED] [RCVED] [CURQTY] [PRICE] [VALUE]

Print on
(EL) - Every Line

Line # Before	Line # After	Space Before	Space After	Look Ahead
0	0	0	0	0

Item	Type	Format	Column	Length	Print
NO	DATA	SD	3	2	VAL
ITEM	DATA	SD	8	13	VAL
PREQTY	DATA	SD	23	2	VAL
SHIPPED	DATA	SD	32	3	VAL
RCVED	DATA	SD	44	2	VAL
*CURQTY	DATA	SD	52	3	VAL
PRICE	DATA	SD	63	6	VAL
*VALUE	DATA	SD	71	7	VAL

```
>7.WAREHOUSE [WHSENBR] TOTAL [VALUE]

Print on
(AS) - Break After Section
Control Items
WHSENBR

    Line #  Line #  Space    Space   Look
    Before  After   Before   After   Ahead

       Ø       Ø       1       1       Ø

 Item                    Type  Format    Column  Length  Print
 WAREHOUSE               TEXT  SD          37       9     VAL
 WHSENBR                 DATA  SD          47       1     VAL
 TOTAL                   TEXT  SD          57       5     VAL
*VALUE                   DATA  SD          71       7     SUB

>8.DIVISION [DIVNBR] TOTAL [VALUE]

Print on
(AS) - Break After Section
Control Items
DIVNBR

    Line #  Line #  Space    Space   Look
    Before  After   Before   After   Ahead

       Ø       Ø       1       1       Ø

 Item                    Type  Format    Column  Length  Print
 DIVISION                TEXT  SD          28       8     VAL
 DIVNBR                  DATA  SD          37       1     VAL
 TOTAL                   TEXT  SD          46       5     VAL
*VALUE                   DATA  SD          71       7     SUB

>9.GRAND TOTAL [VALUE]

Print on
(LP) - Last Page Only

    Line #  Line #  Space    Space   Look
    Before  After   Before   After   Ahead

       Ø       Ø       1       Ø       Ø

 Item                    Type  Format    Column  Length  Print
 GRAND                   TEXT  SD          23       5     VAL
 TOTAL                   TEXT  SD          29       5     VAL
*VALUE                   DATA  SD          71       7     TOT

*  COMPUTED DATA-ITEMS

CURQTY=PREQTY - SHIPPED + RCVED
VALUE=CURQTY * PRICE

Done.
```

INDEX

ABORT command 10-14,A-7
*ABORT directive 10-7
Abort during a command procedure 7-6
Aborting a command procedure 10-7
Aborting a HELP screen 10-14
Absolute line positioning.....12-25
Alphabetic data item 3-7
Alphanumeric data item 3-7
Alternate database descriptions 9-9
APPEND command 6-5, A-8
Appending databases 6-5
Approximate matches 5-4
Arithmetic 6-11
AS (After Section) line printing spec'n.....12-11
Assigning parameter values 10-7
Audit trail 11-7
AUTO-DEFAULT feature 4-6
AUTO-REPEAT feature 4-6
AUTOSELECT feature 7-7
Averages, in Report Writer.....12-14
BASIC files, reading 8-6, C-2
BASIC files, writing 8-6, C-2
BIRTHDAY database 2-4
BS (Before Section) line printing spec'n.....12-11
BZ (Blank if Zero) printing format.....12-12
Calculated report item.....12-12
CHANGE command 6-12, A-9
Changing a database name 7-3
Changing a form 9-3
Changing data item values 6-12
Changing discs 7-1
Changing the data dictionary 9-8
COBOL files, reading and writing 8-6, C-3
Columnar reports 5-10
COMBINE command 6-6, A-10
Combining databases 6-5
Command prompt 2-7
Command echo 7-6
Command procedure, aborting 10-7
Command procedure, conditional 10-6
Command procedure, comments 10-5
Command procedure, editing 10-10

Command procedure, ending 10-7
Command procedure, messages 10-5
Command procedure, restarting 10-10
Command procedures 10-1
Command, Series 20 2-7
Command syntax A-5
Commands, quick reference E-1
Commands, summary A-2
Comments, in command procedures 10-4
COMPARE command 6-16, A-11
Comparing databases 6-16
COMPUTE command 6-12, A-12
Computed report item.....12-12
Computing data item values 6-12
Conditional processing 10-6
COPY command 7-3, A-15
Copying a database 7-3
CP/M files, writing 8-2
CP/M files, reading 8-3
Creating a HELP screen 10-13
Creating the RESULT database 6-2
Cumulative subtotals 5-14
Current disc drive 2-8
D (Duplicate keys) index option.....13-3
DATA, report item type.....12-12
Data dictionary, definition 9-7
Data dictionary, listing 7-2
Data item 3-2
Data item, definition 1-4
Data item definitions, printing 9-4
Data item name 3-2
Data item, report..... 12-9
Data item types 3-7
Data entry 4-1
Data entry field 3-5
Data validation 4-5
DATA.DIC file 9-8
Database, copying 7-3
Database, definition 1-2
Database, destroying 7-4
Database, erasing contents 7-4
Database, reorganizing 9-13
Database arithmetic 6-11
Database files, manipulating 9-3
Database form, definition 1-5
Database installation 2-1
Database inquiry 5-1
Database name 2-7, 3-2
Database names, changing 7-3
Database reports 5-5
Database sorting 5-6
Database structure 1-4,9-1
Databases, appending 6-5
Databases, combining 6-5

Databases, example D-1
Databases, listing 7-2
DATE command 7-5,A-16
Date, in report title 5-17
Date, setting Series 20 7-5
Date data item 3-7
Date format, setting 7-6
DBMS command 7-7,A-17
Default data values 4-5
DEFINE command 3-6,9-5,A-18
DELETE command 6-18,A-20
Deleting records 4-11,6-18
Deleting a database 7-4
Describing a report specification.....12-19
DESTROY command 7-4,A-21
Destroying a database 7-4
DIC command 7-2,A-22
DIR command 7-3,A-23
Disc, choosing 3-5
Disc contents, listing 7-3
Disc directory 7-3
Disc drive, current 2-8
Disc file, printing reports to a 7-5
Discs, switching 7-1
DISPLAY command 5-2,A-24
Dollar data item 3-7
$(dollar sign before commands) 7-8
Duplicating a database 7-3
Echo, setting 7-6
*ECHO directive 10-5
Echoing a command procedure 10-5
EDIT command 10-10,A-26
Editing a command procedure 10-10
EL (Every Line) line printing specification...12-11
EMPTY command 7-4,A-27
Emptying a database 7-4
*END directive 10-7
Ending a command procedure 10-7
ENTER command 4-1,A-28
Entering data 4-1
Entering matching records 11-6
Entering unique records 11-4
EQ search condition 4-10
Equation 6-12
Erasing a database 7-4
Erasing database contents 7-4
Errors, command procedure 7-6
Example databases D-1
Executing Series 20 2-4
Exiting Series 20 2-9,7-8
Expression 6-12
FD (Floating Dollar sign) printing format.....12-14
File formats C-1
Files, copying 9-3

Files, database 9-1
Files, listing 7-3
Form, changing 9-3
Form, creating 3-6
Form, database 1-5
Form, designing 3-4
Form, entering 3-6
Form, report.....12-3,12-9,12-17
Form, revising 9-3
Form feeds, report.....12-16
FORMAT command 9-3,A-30
Format, of WRITE command files C-1
FORTRAN files 8-6
FP (First Page only) line printing spec'n.....12-11
GE search condition 4-10
*GET directive 10-7
General Ledger example 11-1
Graphics, Series 100.....8-5
Grid softkey, database form.....3-6
Grid softkey, report form..... 12-10
GT search condition 4-10
HELP command 10-13,A-32
HELP screen, aborting 10-14
HELP screen, creating 10-13
HELP screen, definition 10-12
HELP screens 11-9
Host computer data 8-7
*IF directive 10-6
INDEX command 13-3,13-4,13-5,A-32.1
Index, existing database.....13-3
Index, invalidating.....13-5,13-6
Index, new database.....13-2
Index, re-creating.....13-4
Index, updating.....13-5
Inputting data 4-1
Inquiry 5-1
Insert mode (FORMAT command) 9-4
JOIN command 6-7,A-33
Joining two databases 6-7
LE search condition 4-10
*LET directive 10-7
Level 1, of Series 20 2-1
Level 2, of Series 20 2-1
Link/125 8-7
LIST command 5-9,A-34
Listing database contents 5-9
Listing databases on a disc 7-2
Listing files 7-3
LJ (Left Justify) printing format.....12-14
LOGDISK command 7-2,A-36
Logging in a disc 7-1
Long method report specification creation.....12-8
LP (Last Page) line printing specification....12-11
LT search condition 4-10

LZ (Leading Zeroes) printing format.....12-14
Mail Lists 8-3,9-11
Master database 6-13,11-3
Master disc drive, setting 7-7
Matching items 6-7
Matching records 11-4
MAX, in Report Writer.....12-14
Maximum value 3-9,4-5
*MESSAGE directive 10-5
Messages, command procedure 10-5
MIN, in Report Writer 12-14
Minimum value 3-9,4-5
Multiple discs 7-1
Naming a database 3-2
Naming data items 3-2
NE search condition 4-10
NP (Not Printed) printing format 12-14
Numeric data item 3-7
Option, in command 2-7
Overriding range checking 4-5
$PAGE 12-12
Page lookahead 12-26
Page numbering 5-18
Parameters, assigning values 10-7
Parameters, command procedure 10-5
Parameters, entering values 10-7
PASCAL files, reading and writing 8-6
PH (Page Heading) line printing spec'n 12-11
Plotting database data 8-5
POST command 6-13,A-37
Posting transactions 6-13
Posting a master database 11-7
PRINT command 5-7,A-39
Printing data item definitions 9-4
Printing reports to a disc file 7-5
Printing reports with Report Writer 12-19
PROJECT command 6-10,A-41
Projecting a database 6-9
Quick reference E-1
Quitting Series 20 2-9
Range checking 4-5
Range checking, overriding 4-5
READ command 9-14,8-3,A-42
Record, definition 1-4
Record selection 6-2
Redefining a database 9-9
RJ (Right Justify) printing format 12-14
Relational database operations 6-1
Relative line positioning 12-26
RENAME command 7-3,A-43
Renaming a database 7-3
Reserved words B-3
REORG command 9-13,A-44
Reorganizing a database 9-13

Repeating data item values 4-6
Replace mode (FORMAT command) 9-4
REPORT command 12-4,12-21,A-44.1
Report, columnar format.....5-10
Report, columnar with statistics 5-11
Report, printing to a disc file 7-5
Report, screen format 5-7
Report, summary format 5-14
Report, statistical 5-10
Report form 12-3
Report form, creating 12-8
Report formats 5-5
Report line spacing 5-18
Report specification 12-3,12-16
Report subtotals 5-13
Report title 5-17
Report totals 5-11
Report Writer 12-1
Report Writer menu 12-7
Reports, database 5-5
Reports wider than 80 columns 12-24
Required data item 3-8
RESTART command 10-10,A-45
Restarting a command procedure 10-10
RESULT database 6-1
RESULT database, creating 6-2
RESULT database, files 9-2
RESULT database, saving 6-3
Returning to CP/M 2-2
Revising data in a form 4-8
Revising a form 9-3
Revising a report specification 12-18
RPG files, reading and writing 8-6
RUN command 10-2,A-46
RUN command parameters 10-2
Running CP/M programs 7-8
Running a command procedure 10-2
SAVE command 6-3,A-49
Screen form 1-5
Screen format reports 5-7
SD (System Default) printing format 12-14
Search condition 4-10
Searching a database 5-1
Section breaks in a report 12-16
SELECT command 6-2,A-50
Selecting data items 6-2
Series 20 specifications B-1
SET command 7-6,A-52
SET ABORT command 7-6,A-52
SET AUTOSEL command 7-7,A-52
SET DATE command 7-6,A-52
SET ECHO command 7-6,A-52
SET MASTER command 7-7,A-52
SET PRINTER command 7-5,A-52

Setting the date 7-5
Setting the date format 7-6
Setting the master drive 7-7
Sharing a database form 9-9
Sharing database data 9-9
Sharing data item definitions 9-9
Short method report creation 12-8
Softkeys, data entry 4-3
Softkeys, display command 5-5
SORT command 5-6,A-54
Sorting a database 5-6
Spacing, in reports 5-18
Special characters 3-3
Specifications, Condor B-1
Specifications, Report item type 12-12
Specifications, Report line printing 12-10
Specifications, Report Writer 12-22,12-23
Startup 2-4
Statistical reports 5-16
STAX command 5-16,A-55
Structure of a database 1-4,9-1
Subtotals 5-13
Subtotals, cumulative 5-14
Subtotals in Report Writer reports 12-14
Summary of commands A-2
Summary reports 5-14
Switching discs 7-1
SYSTEM command 2-9,7-8,A-56
SYSTEM report item 12-12
TAB key 4-2
TABULATE command 5-14,A-57
TEXT report item 12-12
TITLE command 5-17,A-59
Title, report 5-17
$TODAY 12-12
Totals, in PRINT reports 5-11
Totals, in Report Writer reports 12-14
Transaction database 6-13,11-4
Transaction processing 11-1
Types of data items 3-7
U (Update index) index option 13-3
Unique records 11-3
UPDATE command 4-7,A-60
Updating a data base 4-7
Updating master databases 6-13
User programs 8-6
Validation, data 4-5
Verb 2-7
Viewing database records 5-14
Visicalc/100 8-10
Welcome Menu 2-2
Wild card match 5-4
Word/100 mailing lists 8-3,C-2
Word/100 documents 8-4

Wordstar/100 mailing lists 8-3,C-2
Wordstar/100 documents 8-4
WRITE command 9-14,8-2,A-61
WRITE command formats 8-2,D-1
WRITE file formats 8-3,D-1