PEARL LEVEL 3

USER'S GUIDE

November, 1980
Revised

T A B L E   O F   C O N T E N T S

T A B L E   O F   C O N T E N T S

CHAPTER 1. INTRODUCTION

WELCOME TO PEARL LEVEL 3 - FOR PROFESSIONAL PROGRAMMERS

1.1 THE POWER OF PEARL

PEARL, developed by Computer Pathways Unlimited, Inc., is an application generator. You and PEARL interact. You define the specifications of your application (such as complex financial and business applications, multiple file accounting systems). PEARL will generate much or all of the program code needed to implement the application you have defined.

PEARL is "menu" driven -- you define the application and PEARL provides the software solution, in many cases, ready-to-use.

PEARL Level 3 can create the following programs: Menu Selection, File Update/Edit, Report, Edit Control System Data, File Reorganizatioon (indexed files only) and a General Report Writer. These programs will enable you to define and cross-index data elements between multiple files within a single system.

With PEARL Level 3, you can define reports using data from multiple files, extend the standard program menu, and define the interrelationshps between data elements in different files. You can also post journal files to a master file, provide extended report generation and supply multiple index keys for a file.

PEARL Level 3 is designed primarily for use by professional programmers and systems development teams. PEARL Level 3 supports development of complex applications. Because PEARL 3 can do everything that PEARL Levels 1 and 2 can do, you do not have to be a programmer to use it. You can use PEARL 3 to generate a very simple application such as an address list, or you can develop a very complex application requiring the skills of a professional programmer.

1.2 WILL PEARL RUN ON YOUR SYSTEM?

Yes, if you have the following capabilities on your microcomputer system:

1. CP/M* Operating System.
2. 56K of memory.
3. CBASIC-2** Compiler, version 2.03 or later.
4. CRUN2.COM**, version 2.05 or later.
5. At least two floppy disk drives with at least 150K capacity per disk.

*CP/M is a trademark of Digital Research.
**CBASIC and CRUN2.COM are trademarks of Software Systems.

CHAPTER 2. SYSTEM OVERVIEW

PEARL has been developed as a support tool for the design, development, and implemention of applications on microcomputers. The operating system used with PEARL is CP/M. CBASIC-2 is used as the compiler and language because the features provided by this language also support the development of business applications on microcomputers.

2.1 MANUAL ORGANIZATION.

This manual has been organized into five sections.

2.1.1 PART 1 - GETTING STARTED

This section shows you how to configure your working diskettes. You are then lead through a step-by-step definition and generation of a sample system. The example system used is a "Customer Contact File". It is an enchanced version of the application outline in the PEARL Level 2 Manual. Although this example may not be of interest to you as a user, it is important to go through it and generate the application because it will give you the necessary understanding of how you can generate your own system.

2.1.2 PART 2 - SYSTEM DEFINITION.

This section describes in detail the menus and the options provided to allow you to define a system. Each entry in the PEARL main menu is expanded into a full chapter of explanation.

2.1.3 PART 3 - LISTING SYSTEM GENERATION CONTROL DATA.

This section describes in detail the listings of the control data to be used to generate the your user defined system.

2.1.4 PART 4 - SYSTEM GENERATION AND EXECUTION.

This section describes the steps needed to generate a system, to compile the system, and to then place the system in a production status.

2.1.5 PART 5 -- APPENDICES.

Each appendix provides you with supporting information to give you a better understanding of the design and structure of the PEARL-generated applications.

2.2 STEPS TO DESIGN AND DEVELOP AN APPLICATION.

PEARL is a tool to support application development. In order to effectively use PEARL, you must carefully consider your needs, or the needs of the end user for whom the application is being developed. These needs can then be expressed in terms of a system design to be used to define the PEARL control data. The effectiveness of the end system will depend on how well each of the following development phases are completed:

<u>Phase I - System Design.</u>

You will decide the nature of the information to be kept in the file for your application (as described in Chapter 4). PEARL-generated applications are centered on a data file containing a series of data records. The format of each data record is identical. Each record contains the data elements you specify when you define the application. Hence, you must decide what the data elements might be. For example: a last name, a street address, date last called, amount owed, invoice number, etc. Each data element has a type, such as date, character string, money amount, etc. During the process of defining an application to PEARL, you will be prompted to enter descriptions of all the data elements the data record will contain.

<u>Phase II - RUN PEARL.</u>

1. You will put your DEFINITION and APPLICATION SYSTEM DISK (drives A and B respectively) into your computer and RUN PEARL (as explained in Chapter 4). You will sequentially select options from the <u>Main Menu</u>.

2. You will define a system. You will be prompted to enter a system name, what to name the Main Menu Program for your new system, the version number, etc.

3. You will define the application file. You will be asked for such things as the name to give your application, which disk drive it will be kept on, etc.

4. You will define the data elements that will make up the records on the application file.

5. For data elements representing an array of finite data possibilities, each data possibility is defined both as a number and by its phrase definition.

6. You will define your reports. A description of entry is discussed in detail in Chapter 4, Step 7. Chapter 10 discusses all of the report options available.

7. You will print out a list of what you have defined to be sure it contains all the data elements you want and that they are properly defined. There is a length limit as to how long a record can be, and you must insure you have not exceeded this limit.

8. You may need to go back and edit your definitions if necessary, and then print them out. You may repeat this procedure until you are satisfied with your new system design.

Phase III - Generate Your Programs.

PEARL Main Menu option 15, System Generation, suboption 1 will generate all the programs for the system you have defined.

Phase IV - Compilation.

PEARL Main Menu option 15, submenu Option 10 will set up a submit file so you can then insert your compile disk, submit the compile process and then run your new system.

Phase V - Backups.

You will backup all your disks and store them away in a safe place for protection in case something should happen to the originals.

CHAPTER 3. CONFIGURING YOUR WORKING DISKS

PEARL Level 3 is designed to run on a number of hardware configurations. At least one disk unit is required. However, the PEARL programs, control files, and generated source files are most easily managed when they are distributed over several drives. Approximately 600K of storage is required to maintain the PEARL programs, the run time CBASIC programs, the control files, and the source files for the common subroutines.

This section describes some sample configurations using a variety of computers and disk storage configurations. These include:

1. An ALTOS computer with two double density floppy disk drives and hard disk storage distributed across 6 units (approximately 9 megabytes each).

2. Any system using 8-inch single density floppy diskettes.

3. An ALTOS or TRS-80 Model 2 computer with two double density floppy disk drives.

   In addition to the above, special configuration directions are available for configuring PEARL on such systems as Superbrain, North Star Horizon, etc.

3.1 DISTRIBUTION DISKETTES

The distribution diskettes contain three logical sets of files. When distributed on single density, soft sectored 8-inch diskettes, each of three diskettes contain the following files:

<u>1. Definition Programs.</u>

PEARL3.INT	MAIN MENU PROGRAM
A000.INT	SYSTEM INITIALIZATION PROGRAM
A100.INT	FILE DEFINITION PROGRAM
A200.INT	DATA ELEMENT DEFINITION PROGRAM
A300.INT	PHRASE SELECTION DEFINITION PROGRAM
A400.INT	REPORT CONTROL DEFINITION PROGRAM
A400A.INT	REPORT DETAIL DEFINITION PROGRAM
A500.INT	MENU DEFINITION PROGRAM
A600.INT	POST/CLOSE COMPUTATIONS ENTRY PROGRAM
A800.INT	EDIT SYSTEM CONTROL DATA
AP0001.INT	LIST DATA ELEMENT CONTROL
AP0002.INT	LIST REPORT CONTROL
AP0003.INT	LIST MENU CONTROL
AP0004.INT	LIST POST/CLOSE COMPUTATIONS
AP0005.INT	CROSS FILE VALIDATION PROCESSING

2. Generation Programs.

A900.INT	GENERATION CONTROL PROGRAM
A900A.INT	GENERATE I/O ROUTINES
A900B.INT	GENERATE DISPLAY/EDIT ROUTINES
A900C.INT	GENERATE REPORT PROGRAMS
A900D.INT	GENERATE UPDATE/EDIT, REORGANIZATION MAINLINE
A900E.INT	GENERATE INITIALIZATION ROUTINES
A900F.INT	GENERATE MAIN MENU PROGRAM
A900G.INT	GENERATE POST/CLOSE ROUTINES
PLT031.NDX	PROGRAM LOGIC TABLES
PLT032.NDX	PROGRAM LOGIC TABLES

3. Common Source Utilities.

C24000E.BAS	POST/CLOSE REPORT WRITER
C82000E.BAS	CONSOLE INPUT ROUTINES
C82400A.BAS	CREATE QSORT CONTROL FILE
C82500A.BAS	CREATE SUBMIT CONTROL FILE
C86900E.BAS	FILE COPY ROUTINES
C90000E.BAS	SCREEN CLEAR, PAUSE, CHAIN UTILITIES
C90400E.BAS	CHAIN LIST PROCESSOR
CENTRYE.BAS	COMMON ENTRY ROUTINE
CHAIN.BAS	SET %CHAIN COMPILER DIRECTIVES
CINTL.BAS	TERMINAL CONFIGURATION INITIALIZATION
COM01E.BAS	COMMON VARIABLES
FNFILEE.BAS	FUNCTIONS FOR FILE STRING PROCESSING
IS73000.BAS	INDEX FILE INITIALIZATION
ISCAN.BAS	INDEX FILE I/O (READ ONLY)
ISUPDATE.BAS	INDEX FILE I/O
ISINTL.BAS	INDEX FILE CONTROL DATA
XSORTX.BAS	SORT EXIT UTILITY MAINLINE
SR10000.BAS	I/O ROUTINE FOR SORT CONTROL FILE
SR24000.BAS	DISPLAY EDIT FOR SORT CONTROL DATA

3.2 SETTING UP WORKING DISKETTES.

The following is a general description of the diskettes you will probably want to use during definition, generation, compilation, and placing an application into production. Depending upon the size the of system being generated and the storage capacity of your diskettes, you may have to alter the configuration somewhat as described in Section 3.3.

1. PEARL DEFINITION DISK

   This diskette will be used when defining a system, or anytime the PEARL programs are used.

2. PEARL GENERATION DISK

   This diskette will be used during system generation.

3. CONTROL DATA DISK

   This diskette will contain the PEARL control files. During generation processing, generated code will be stored on this diskette.

4. PEARL COMPILE MASTER DISK

   This diskette will be set up once with the programs required in order to compile a PEARL-generated system. Once set up, it will be copied onto the MAIN PROGRAM DISK for each generated system.

5. APPLICATION SYSTEM MAIN PROGRAMS

   This diskette will contain the common subroutines and will be used to hold the main program modules during the segment compile process after the system is generated.

6. GENERATED SUBROUTINES

   This diskette will contain the generated subroutines for the application being generated after being moved from the control data diskette.

7. APPLICATION SYSTEM DISK

   This will be the production diskette for the end system. It will contain only the .INT files for the generated system, and the necessary .COM files required to run the system.

8. APPLICATION DATA FILES

   This diskette will contain the data files created and maintained by the system you will generate with PEARL.

3.3 ALTERNATE CONFIGURATIONS

The following configurations may vary somewhat depending upon the version of PEARL Level 3 you are using and the actual storage capacity of your diskettes.

3.3.1 ALTOS System With Hard Disk Storage Capacity.

NOTE:

While this configuration was actually done using an ALTOS system, the same configuration will work for any system with mass storage capability on hard disk.

In this case, units A and B are double density floppy diskettes, and units E, F, G, H, I, and J are hard disk units with approximately 8 megabytes each. The following allocation of drives was done:

E:	COMMON UTILITY SOURCE
F:	GENERATED SOURCE
G:	WORKING SYSTEM DISK
H:	PEARL CONTROL FILES

In order to configure the system to run PEARL, files were copied to the hard disk units as follows:

a. All files from distribution diskettes 1 and 2 were placed on drive G:.

b. All files from distribution diskette 3 was placed on drive G:.

c. CRUN2.COM (renamed RUN.COM), CBAS2.COM, QSORT.COM, were copied from respective master distribution diskettes to drive G:.

To begin execution of the system, enter the command:

```
G:RUN G:PEARL3
```

When the program is loaded, the following message will appear:

```
SYSTEM CONIFURATION DATA COULD NOT BE LOCATED
Enter an ESCAPE to create a new file on DRIVE A:, or,
ENTER DEFAULT SYSTEM DISK DRIVE [A]
```

At this point, enter a "G" followed by a return. The same message will appear again with "G" in place of "A". At this time, enter an ESCAPE followed by a RETURN. This will indicate to the system that your system disk drive is to be defined as G. Then respond to each of the prompts provided by the program to complete the definition of the system configuration information. (Refer to Appendix B for a description of each of these control variables.)

3.3.2 Using Single Density 8-inch Floppy Diskettes.

This configuration will work for any system with a diskette storage capacity of 241K per diskette.

Place a CP/M operating system, and a PIP.COM file on each of the diskettes described in Section 3.2. Then distribute the PEARL Level 3 files provided on your three

MASTER diskettes as described below. Note that QSORT.COM, RUN.COM (CRUN2.COM renamed), and CBAS2.COM will also be required on some of the diskettes as noted below.

Distribution Disk	File	Working Disk
DEFINITION	PEARL3.INT	PEARL DEFINITION DISK
"	A000.INT	"
"	A100.INT	"
"	A200.INT	"
"	A300.INT	"
"	A400.INT	"
"	A400A.INT	"
"	A500.INT	"
"	A600.INT	"
"	A800.INT	"
"	AP0001.INT	"
"	AP0002.INT	"
"	AP0003.INT	"
"	AP0004.INT	"
"	AP0005.INT	"
CBASIC	RUN.COM	"
GENERATION	A900.INT	PEARL GENERATION DISK
"	A900A.INT	"
"	A900B.INT	"
"	A900C.INT	"
"	A900D.INT	"
"	A900E.INT	"
"	A900F.INT	"
"	A900G.INT	"
"	PLT031.NDX	"
	PLT032.NDX	
COMPILE	C24000E.BAS	PEARL COMPILE MASTER
"	C82000E.BAS	"
"	C83000E.BAS	"
"	C86900E.BAS	"
"	C90000E.BAS	"
"	C90400E.BAS	"
"	CENTRYE.BAS	"
"	CHAIN.BAS	"
"	CINTL.BAS	"
"	COM01E.BAS	"
"	FNFILEE.BAS	"
"	IS73000.BAS	"
"	ISCAN.BAS	"
"	ISINTL.BAS	"
"	ISUPDATE.BAS	"
CBASIC	CBAS2.COM	"
CP/M	SUBMIT.COM	"
"	ED.COM	"

COMPILE	XSORTX.INT	APPLICATION SYSTEM DISK
QSORT	QSORT.COM	"
CBASIC	RUN.COM	

The APPLICATION SYSTEM MAIN PROGRAMS diskette should be a duplicate copy of the PEARL COMPILE MASTER DISK.

NOTES:

The XSORTX.BAS, C82400A.BAS, C82500A.BAS, SR10000.BAS, and SR42000.BAS source files were not distributed from the distribution master diskette. These routines will not be required unless you wish to modify the SORT exit routine. If you wish to do this, copy these routines onto a copy of the PEARL COMPILE MASTER DISK where development work can be done.

The XSORTX.INT file is provided on the distribution diskette in source form (XSORTS.BAS). In order to set up the INT file on the system diskette for the inventory system, the source program must be compiled.

The ED.COM and SUBMIT.COM files are included on the PEARL COMPILE MASTER DISK in order to create submit files to facilitate system compilation. Their use is described later in these procedures.

3.3.3 Configuring a System With Dual Density 8-inch Floppy Drives

For systems with two or more dual density 8-inch floppy drives (440K capacity per diskette), the above configuration will also work quite well. Generation of large systems will generally need less diskette swapping. There will be less need to copy the generated source to separate diskettes to allow room for complete generation of all programs.

3.4 CONFIGURING SYSTEMS WITH MORE THAN TWO DISK DRIVES.

PEARL will run on a system with only two disk units. However, if your system has more than two drives, it is advisable to distribute your working files among as many diskettes as possible during the generation and compile steps. This will allow you to generate large systems with a minimum of diskette swapping. Files can be distributed easily as follows:

DISK UNIT DESCRIPTION	CONTENTS	WHERE DEFINED
SYSTEM DISK	PEARL.INT FILE, PROGRAM LOGIC TABLES, SYSTEM .COM FILES	SYSTEM CONFIGURATION
CONTROL DATA	PEARL.CCT, PEARL.NDX FILES	SYSTEM CONFIGURATION

GENERATED SOURCE	.BAS FILES CREATE BY PEARL	PEARL CONTROL DATA
COMMON UTILITIES SOURCE	.BAS SOURCE FILES PROVIDED OF PEARL DISTRIBUTION DISKETTES	PEARL CONTROL DATA
APPLICATION SYSTEM DISK	.INT FILES FOR GENERATED APPLICATION	SUBMIT FILE FOR COMPILE.

NOTES

1. If each of the above sets of files are defined on separate diskettes, the total space requirements on each diskette will be reduced.

2. If you had only a single disk unit on your system, all files could be placed on the same unit if the storage capacity of the drive would permit it.

3.5 SUMMARY

Format 8 diskettes and label them (see 3.2). Then put CP/M and RUN.COM on seven of these disks. Put CP/M and CBAS2.COM on the MASTER COMPILE DISK. You may put PIP.COM on all disks except the GENERATION DISK (see 3.2). (This is because on most systems, there is not enough room for PIP.COM on the generation disk.) Put the rest of the COM files on the APPLICATION SYSTEM DISK.

PIP all of the appropriate files over onto the DEFINITION, GENERATION, MASTER COMPILE and CONTROL DATA disks (see 3.3).

COPY the PEARL MASTER COMPILE disk onto another disk and label it APPLICATION SYSTEM MAIN PROGRAMS. You will use this disk for compiling. Once you have completed compilation, you will PIP RUN.COM, your .COM files and your .INT files from this disk onto your APPLICATION SYSTEM DISK for production.

Then, put the other three disks aside for the time being. Store your three PEARL DISTRIBUTION MASTER DISKS and your PEARL MASTER COMPILE disk away in a safe place.

See Chapter 19 for more detailed information on usage of diskettes and procedures.

Proceed to Chapter 4.

CHAPTER 4. GENERATING A TEST SYSTEM

This chapter presents an example of how you can use PEARL to generate an application to serve your needs.

The example is a enhanced version of the one given in the PEARL Level 1 and 2 manual. You would not necessarily have to be a programmer to generate the example given here. However, you would need to have access to someone with programming knowledge in BASIC code to be able to work on some parts of this example (such as defining reports). XYZ Company is the business desiring a computer application as follows:

THE COMPANY

XYZ Company is a service organization. They currently serve over 200 customers. One of the sections of XYZ Company is Customer Support. Customer Support is heavily involved in public relations and is responsible for ensuring that the products provided their customers are satisfactory. One way of doing this is by maintaining a "customer contact file". Therefore, every Monday they go through their files and get a list of those customers who need to be contacted during the week. They must also add new customers to the file, maintain current customers by address and telephone number, and remove inactive customers from their files, if necessary.

Customer Support also provides Marketing with a list each month of the number of customers in each business group. This gives Marketing an idea of how much business is being done and in which areas it is being done. The list can be used as one type of guideline for planning marketing strategy.

PEARL can maintain all these records for them and list them out in alphabetical order on a line printer. With PEARL Level 3, the company can also get a report of customers on the file by business group, in order by contact date, and business group totals/selected date range.

Customer Support, therefore, has decided to outline the specifications they desire for this type of application, enter these specifications into the terminal using PEARL and let PEARL generate the application for them.

NOTE:

While PEARL Level 3 can generate a system like PEARL Levels 1 and 2 (i.e., without a programmer), it can also do much more. A programmer would need to be involved in the process of generating this example application where additional applications are provided using Level 3.

"SYSTEM DESIGN"

<u>Requirements:</u>

1. Add customers to the file.

2. Maintain current customer files through editing when necessary, i.e., address changes, phone number changes.

3. List customers on a line printer by name, contact date, business group and total business group within a selected date range.

4. Delete inactive customers.

<u>Records Will Contain:</u>

1. Customer Name (40 characters long)

2. Customer Address (40 characters long)

3. City, State and Zip Code (30 characters long)

4. Phone Number (12 characters long)

5. Contact Date (6 characters long)

6. Business Group: (2 characters long)
   government
   education
   contracting
   farming
   retail sales
   wholesale distribution
   other

Each record will be 130 characters in length and contain 6 data fields within each record.

<u>Application:</u>

1. Add, edit, delete and list customer records from customer contact file.

2. There will be two reports: a) the first report will list customers currently on the file; and b) the second report will list customers by business group and by contact date. The business group report will give subtotals for each business group and the combined month-to-date total for business groups in a given month. The report will also give a year-to-date total for each business group.

CHAPTER 4. GENERATING A TEST SYSTEM

PRESENT XYZ COMPANY HARDWARE/SOFTWARE

1. CP/M Operating System.

2. Computer with 64K of memory.

3. CBASIC-2 compiler, version 2.03.

4. CRUN2.COM, version 2.06

5. Printer with automatic forms eject.

6. Hazeltine* Video Display Terminal.

7. 8-inch double density disks with 448K capacity.

It is important to be familiar with your operating system and to have followed the procedures in Chapter 3 carefully. The steps for defining the sample application follow. At this time, your working disks should be in hand and your PEARL master disks safely stored away for future use. **PLEASE DO NO USE YOUR PEARL MASTER DISKS FOR GENERATING A SYSTEM.**

*Hazeltine is a product of Hazeltine Corporation.

Step 1 - Configuration of Control Data

This is required only once. After this information (data) has been entered and the program generated, it can be changed by selecting Option 16 on the Main Selection Menu (see Step 2 for the first display of the Main Selection Menu). The system will prompt the you to enter the appropriate data.

However, since this section is provided to show you how to generate a system using PEARL Level 3 from beginning to end, this option is covered in detail.

User Action:

Place Definition Disk in Drive A.

Place Control Data Disk in Drive B.

Perform CP/M cold boot.

System Responds:

```
A>
```

User Enters:

```
RUN PEARL3
```

System Responds:

```
CRUN VERSION 2.06

SYSTEM CONFIGURATION DATA COULD NOT BE LOCATED
Enter an ESCAPE to create a new file on DRIVE A:, or,
ENTER DEFAULT SYSTEM DISK DRIVE
```

User Enters:

Depress ESCAPE and RETURN

System Responds:

```
SYSTEM CONFIGURATION CONTROL FILE IS BEING INITIALIZED
CONFIGURATION CONTROL DATA
1:   TERMINAL TYPE                  USER DEFINED CLEAR SCREEN
2:   FORMS CONTROL OPTION           USE FORM FEED
3:   DISKETTE CAPACITY IN K BYTES            70
4:   MESSAGE LEVEL                  SUPPRESS ALL MESSAGES
5:   REPORT DEPTH IN LINES                    0
6:   INSTALLATION NAME
7:   DEFAULT SYSTEM DISK DRIVE      A
8:   DEFAULT DATE DISK DRIVE        B
9:   DATE FORMAT                    MM/DD/YY FORMAT
10:  PASSWORD
ENTER TERMINAL TYPE
 0 =USER DEFINED CLEAR SCREEN
 1 =SOL
 2 =HAZELTINE
 3 =BEEHIVE
 4 =SOROC
 5 =INTERTEC
 6 =TRS 80 MODEL II
 7 =ADM-3(A)
( 0 )
```

User Enters:

```
2
```

A list of the common terminal types is provided. If your terminal (video display) is one of those listed, then enter the number to make your selection. If your terminal is not included, enter zero as your selection and then enter the character sequence which is required to clear the screen on your terminal. This information can usually be found in the manufacturer's manual for your terminal.

System Responds:

```
ENTER FORMS CONTROL OPTIONS ( 0 )
ENTER 0 TO USE FORM FEED, OR THE NUMBER OF LINES PER PAGE
```

User Enters:

```
0
```

This option is used to specify the control used to space to top of form on your system printer.

If a zero is specified, the the ASCII FF (form feed) character will be used to space to top of form. If your printer does not support forms eject protocol, then you should specify the number of lines from the top of form to the top of the next page. For standard spacing on 11-inch paper this value would be 66 (6 lines per inch). Some printers allow the option to print 8 lines per inch in which case the forms length would be 88.

System Responds:

```
ENTER DISKETTE CAPACITY IN K BYTES ( 70 )
```

User Enters:

```
448
```

System Responds:

```
ENTER MESSAGE LEVEL
0 =SUPPRESS ALL MESSAGES
1 =OPEN/CLOSE/CHAIN MESSAGES
2 =DISPLAY I/O TRACES
( 0 )
```

User Enters:

```
0
```

Three options are provided here and give the following levels of messages and diagnostics during processing:

0 SUPPRESS ALL MESSAGES

The only messages returned to the you are those resulting in the successful execution of a program; or, if an error is detected, the error code associated with the error. This option is suggested for this test run. Options 1 and 2 are programmers tools and increase the processing time.

(Options 1 and 2 are programmer's tools.)

1 OPEN/CLOSE/CHAIN MESSAGES

If this diagnostic message level is chosen, then a message will be provided to you each time a file is opened, closed, or a chain to another program is being issued. At the time a file is closed, a count of the number of records added to, read from, deleted from, and replaced on the file will be given. In addition, the amount of free space in memory will also be given.

2 DISPLAY I/O TRACE

When this option is used, a display of the records read from, and written to, a file will be displayed on the screen. This option is not available for PEARL programs themselves but is available for programs generated by PEARL.

System Responds:

```
ENTER REPORT DEPTH IN LINES ( 0 )
```

User Enters:

```
55
```

This control variable specifies the number of lines in the body of each printed page on a report. This value will always be used on your reports unless you enter a different value when defining your report.

System Responds:

```
ENTER INSTALLATION NAME
```

User Enters:

```
XYZ Company
```

System Responds:

```
ENTER DEFAULT SYSTEM DISK DRIVE ( A )
```

User Enters:

Depress RETURN (or choose another drive A-D)

System Responds:

```
ENTER DEFAULT DISK DRIVE FOR DATA FILES ( B )
```

(or choose another drive A-D)

User Enters:

Depress RETURN

System Responds:

```
ENTER DATE FORMAT
0 =MM/DD/YY FORMAT (American)
1 =DD/MM/YY FORMAT (European)
(0)
```

User Enters:

```
0
```

System Responds:

```
ENTER PASSWORD
```

User Enters:

You will enter a password of your choice if you desire. If you enter a password, be sure not to forget it or you will not be able to get onto the system. The password, as it is being entered, will be blocked out by the number symbol during keying. For example, if you enter "secret" as your password, the field as entered would display as ######. If you enter your password in lowercase, it must always be keyed that way; likewise, if you enter it in UPPERCASE, it must always be keyed in the same way.

```
#######
```

System Responds:

```
CONFIGURATION CONTROL DATA
1: TERMINAL TYPE                       HAZELTINE
2: FORMS CONTROL OPTION                USE FORM FEED
3: DISKETTE CAPACITY IN K BYTES              448
4: MESSAGE LEVEL                       SUPPRESS ALL MESSAGES
5: REPORT DEPTH IN LINES                      55
6: INSTALLATION NAME                   XYZ Company
7: DEFAULT SYSTEM DISK DRIVE           A
8: DEFAULT DATA DISK DRIVE             B
9: DATE FORMAT                         MM/DD/YY FORMAT
10: PASSWORD
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

<u>NOTE:</u>

Again, the password will not show on this display. However, you may change the password field by entering 10 and entering a new password, to make sure you entered the word you wanted or you may simply terminate the password altogether by depressing RETURN.

User Enters:

You would check the above fields to make sure they are correct and then either edit the incorrect field if needed or depress RETURN.

As discussed in Appendix A, this ability is provided you throughout the entire process. Therefore, if a error has been made during initial entry, you will be given a chance to correct it before going further.

Depress RETURN

GO TO STEP 2

Step 2 - System Initialization

System Responds:

```
                              PEARL LEVEL 3

                (C) BY COMPUTER PATHWAYS UNLIMITED, INC.
                              VERSION N.00
                             AUGUST 1, 1980

SERIAL # AAA-NNN

ENTER CURRENT DATE:
```

User Enters:

```
     MMDDYY (with or without slashes, i.e., 010180)
```

System Responds:

```
                   PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                       MAIN SELECTION MENU-MM/DD/YY
                        MINIMUM FREE SPACE=(20064)
                                (FILE=01)*
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION*
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

NOTES

1. File=NN lets you know on which file you are currently working. When you initialize a file, 1 is always used. However, if you were using multiple files, and you wanted to work on file 2, you could change the number to 02. See Step C, File Definition, on how you could do this.

2. Option 3, PHRASE SELECTION DEFINITION, is the same as DATA ELEMENT ARRAY VALIDATION MAINTENANCE on Levels 1 and 2 of PEARL. We felt PHRASE SELECTION more descriptive, and

certainly less cumbersome, so we changed it on Level 3 of PEARL.

User Enters:

```
     1
```

System Responds:

```
LOADING PROGRAM TO SYSTEM INITIALIZATION

               PEARL LEVEL 3 (A:A000) VERSION N.00
                 SYSTEM INITIALIZATION-01/01/80
                   MINIMUM FREE SPACE=(20064)
                           (FILE=01)

ENTER SYSTEM DESCRIPTION ()
```

User Enters:

```
     CUSTOMER CONTACT FILE
```

System Responds:

```
     ENTER MAIN MENU PROGRAM NAME ()
```

User Enters:

```
     CONTACT
```

System Responds:

```
     ENTER SYSTEM CODE ()
```

(See Appendix G for system codes that should <u>not</u> be used.)

User Enters:

```
     BCF  (for Business Contact File)
```

System Responds:

```
     ENTER SYSTEM DEVELOPMENT DATE
```

User Enters

```
     September, 1980
```

System Responds:

```
     ENTER SYSTEM VERSION NUMBER
```

User Enters:

```
     1.0
```

System Responds:

```
     ENTER BUFFER ALLOCATION ( 0 )
```

User Enters:

     10 (If you have 48K of memory, enter 0 instead.)

System Responds:

```
     ENTER DRIVE FOR GENERATED SOURCE (B)
```

User Enters:

     Depress RETURN (or desired drive)

System Responds:

```
     ENTER DRIVE FOR INCLUDED COMMON SUBROUTINES (A)
```

User Enters:

     Depress RETURN (or desired drive)

System Responds:

```
     ENTER DRIVE FOR INCLUDED GENERATED SUBROUTINES (B)
```

User Enters:

     Depress RETURN

System Responds:

```
PEARL SYSTEM CONTROL DATA
1: SYSTEM DESCRIPTION               CUSTOMER CONTACT FILE
2: MAIN MENU PROGRAM NAME           CONTACT
3: SYSTEM CODE                      BCF
4: SYSTEM DEVELOPMENT DATE          September, 1980
5: SYSTEM VERSION NUMBER            1.0
6: BUFFER ALLOCATION                           10.
7: DRIVE FOR GENERATED SOURCE                 B
8: DRIVE FOR INCLUDED COMMON ROUTINES         A
9: DRIVE FOR INCLUDED GENERATED ROUTINES      B
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
     SYSTEM INITIALIZATION IS COMPLETED

     ....RETURNING TO MAIN MENU....
```

GO TO STEP 3

Step 3 - File Definition Processing

Once back in the main menu (shown below) you may use Option 14 if you wish to change the data you have just entered. Otherwise, you are now ready to select Option 2.

```
               PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                  MAIN SELECTION MENU-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION CONTROL DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     2
```

System Responds:

```
LOADING PROGRAM TO FILE DEFINITION

               PEARL LEVEL 3 (A:A100) VERSION N.00
               FILE DEFINITION PROCESSING-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)

0    TO RETURN TO THE MAIN MENU
1    TO ADD A FILE TO THE SYSTEM
2    TO EDIT FILE DEFINITION CONTROL DATA
3    TO LIST FILE DEFINITION CONTROL DATA
F=nn  TO RESET FILE ID*
?
```

If you wish to change the file number in order to define a file other than 1, you may enter F=NN where NN is the file number you wish to define, i.e., F=02.

User Enters:

```
1
```

System Responds:

```
ENTER FILE DESCRIPTION ()
```

User Enters:

```
CUSTOMER CONTACT
```

System Responds:

```
ENTER FILE TYPE
0 =MASTER FILE
1 =MISC. INDEXED FILE
2 =TRANSACTION FILE
3 =HISTORICAL TRANSACTION FILE
4 =MISC. RANDOM FILE
```

User Enters:

```
1
```

For the application outlined in this section of the User's Guide, a miscellaneous indexed file is used. This is because there will be only one file used for this application and so there is no need to select MASTER FILE as opposed to TRANSACTION FILE or HISTORICAL FILE for instance.

System Responds:

```
ENTER DATA FILE NAME ()
```

User Enters:

```
CUSTOMER
```

System Responds:

```
ENTER UNIQUE FILE ID ()
```

User Enters:

```
CF
```

System Responds:

```
ENTER LOGICAL FILE UNIT ( 2 )
```

User Enters:

Depress RETURN

(User must use 2 through 20 - 1 is reserved for the index file.)

System Responds:

```
ENTER DISK DRIVE UNIT ID (B)
ENTER "Z" TO USE DEFAULT DATA DRIVE
```

User Enters:

Depress RETURN

The CUSTOMER CONTACT FILE system will produce the data file on Drive B.

System Responds:

```
ENTER I/O SUBROUTINE BASE ( 10000 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER DATA RECORD EDIT BASE ( 30000 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER CONTROL RECORD EDIT BASE ( 40000 )
```

User Enters:

Depress RETURN

System Responds:

```
FILE DEFINITION DATA
1: RELATIVE FILE ID                       1.
2: FILE DESCRIPTION             CUSTOMER CONTACT FILE
3: ACCESS METHOD                MISC. INDEXED FILE
4: DATA FILE NAME               CUSTOMER
5: UNIQUE FILE ID               CF.
6: LOGICAL FILE UNIT                     2.
7: DISK DRIVE UNIT ID           B
8: I/O SUBROUTINE BASE          10000
9: DATA RECORD EDIT BASE        30000
10: CONTROL RECORD EDIT BASE    40000
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
ADDING DATA ELEMENTS FOR FILE HEADER RECORD
CFC.SORT HAS BEEN ADDED
CFC.LAST HAS BEEN ADDED
CF.NEXT.DELETED HAS BEEN ADDED
CF.AVAIL.DELETED HAS BEEN ADDED
CF.UPDATE.FLAG HAS BEEN ADDED
CF.UPDATE.RESET HAS BEEN ADDED
CF.IS.NBUFS% HAS BEEN ADDED
```

NOTE

The following data elements for the file header record would be added if you had defined your file as MASTER, TRANSACTION or HISTORICAL only:

```
CF.S.CLOST.COUNT% HAS BEEN ADDED
CF.SH.JE% HAS BEEN ADDED
CF.S.POST.COUNT% HAS BEEN ADDED
CF.S.JE% HAS BEEN ADDED

                PEARL LEVEL 3 (A:A100) VERSION N.00
               FILE DEFINITION PROCESSING-01/01/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO THE MAIN MENU
1   TO ADD A FILE TO THE SYSTEM
2   TO EDIT FILE DEFINITION CONTROL DATA
3   TO LIST FILE DEFINITION CONTROL DATA
F=NN  TO RESET FILE ID
?
```

User Enters:

0

System Responds:

```
     ... RETURNING TO MAIN MENU ...
```

GO TO STEP 4

Step 4 - Data Element Definition

The next step is to enter the control information for each of the data elements you are going to place on your data file. For this particular application, there are 6 data file names: CUSTOMER NAME, CUSTOMER ADDRESS, LOCATION (City, State, Zip Code), PHONE NUMBER, CONTACT DATE, and BUSINESS GROUP. Each will be defined separately as follows:

System Responds:

```
     ... RETURNING TO MAIN MENU ...

              PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                  MAIN SELECTION MENU-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)
 0. RETURN TO CP/M
 1. SYSTEM INITIALIZATION
 2. FILE DEFINITION
 3. DATA ELEMENT DEFINITION
 4. PHRASE SELECTION DEFINITION
 5. MAIN MENU DEFINITION
 6. REPORT CONTROL DEFINITION
 7. REPORT DETAIL DEFINITION
 8. ENTER/EDIT POST/CLOSE COMPUTATIONS
 9. LIST DATA ELEMENT CONTROL DATA
10. LIST MENU CONTROL DATA
11. LIST REPORT CONTROL DATA
12. LIST POST/CLOSE COMPUTATION CONTROL DATA
13. VALIDATION OF CROSS FILE PROCESSES
14. EDIT SYSTEM DEFINITION DATA
15. SYSTEM GENERATION
16. RESET CURRENT SYSTEM DATE
17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     3
```

System Responds:

```
     LOADING PROGRAM TO DATA ELEMENT DEFINITION
```

```
                    PEARL LEVEL 3 (A:A200) VERSION N.00
                   FILE DATA ELEMENT DEFINITION-MM/DD/YY
                        MINIMUM FREE SPACE=(NNNNN)
                                (FILE=01)

0    RETURN TO MAIN MENU
1    TO ADD NEW DATA ELEMENTS
2    TO EDIT EXISTING DATA ELEMENTS
3    TO DELETE DATA ELEMENTS
F=NN  TO RESET FILE ID
?
```

User Enters:

```
     1
```

System Responds:

```
     Enter ESCAPE to terminate add processing, OR
     ENTER EDIT LINE SEQUENCE ( 10 )
```

(This is the line number associated with the data element being defined. You may choose a different number for your data element if you wish. See Chapter 6 for more information.)

User Enters:

Depress RETURN

The field you will be defining will be the <u>record key</u> and must be unique because it is the first data element on the file. Records will be accessible from the file in order by CUSTOMER NAME, last name first, i.e., SMITH, MARY E.

System Responds:

```
     ENTER VARIABLE DESCRIPTION ()
```

User Enters:

```
     CUSTOMER NAME
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     CUST.NAME
```

System Responds:

```
ENTER STORAGE FORMAT
0 =FLOATING POINT
1 =INTEGER
2 =STRING
3 =DATE
4 =MONEY
5 =COMPUTATIONAL
( 0 )
```

User Enters:

```
2
```

System Responds:

```
 ENTER LENGTH OF FIELD ( 0 )
```

User Enters:

```
40
```

System Responds:

```
ENTER EDIT CONTROL OPTION
0 =NO EDITING RESTRICTIONS
1 =EDIT DURING ADD ONLY
2 =DISPLAY ONLY
3 =SUPPRESS DISPLAY/EDIT
( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VALIDATION CONTROL OPTION
0 =NO SELECTION
1 =Y/N SELECTION
2 =ALPHA SINGLE CHARACTER
3 =PHRASE SELECTION
4 =CROSS FILE KEY VALIDATION
( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER KEY OPTION
      0 =NOT APPLICABLE
      1 =PRIMARY KEY
      2 =SECONDARY KEY
     ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                            10
2: VARIABLE DESCRIPTION                      CUSTOMER NAME
3: VARIABLE CODE NAME                        CUST.NAME$
4: NUMBER OF OCCURRENCES                               0.
5: STORAGE FORMAT                            STRING
6: LENGTH OF FIELD                                    40.
7: LOW RANGE FOR VARIABLE                    NONE
8: HIGH RANGE FOR VARIABLE                   NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                      NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION                NO SELECTION
12: KEY OPTION                               PRIMARY KEY
*: NUMBER OF ARRAY VALIDATION ENTRIES                 0.
*: POSITION IN RECORD                                 0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

<u>NOTE</u>

Not all of the values were entered for this data element. Lines 4:, 7:, 8: and 9: were defaulted because the other entries were such that these four lines were not required.

User Enters:

Depress RETURN

The next data element will be CUSTOMER ADDRESS.

System Responds:

```
     Enter ESCAPE to terminate add processing, OR
     ENTER EDIT LINE SEQUENCE ( 20 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VARIABLE DESCRIPTION (CUSTOMER NAME)
```

User Enters:

```
CUSTOMER ADDRESS
```

System Responds:

```
ENTER VARIABLE CODE NAME (CUST.NAME$)
```

User Enters:

```
CUST.ADDR
```

System Responds:

```
ENTER STORAGE FORMAT
0 =FLOATING POINT
1 =INTEGER
2 =STRING
3 =DATE
4 =MONEY
5 =COMPUTATIONAL
( 2 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER LENGTH OF FIELD ( 40 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER EDIT CONTROL OPTION
0 =NO EDITING RESTRICTIONS
1 =EDIT DURING ADD ONLY
2 =DISPLAY ONLY
3 =SUPPRESS DISPLAY/EDIT
( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
      ENTER VALIDATION CONTROL OPTION
      0 =NO SELECTION
      1 =Y/N SELECTION
      2 =ALPHA SINGLE CHARACTER
      3 =PHRASE SELECTION
      4 =CROSS FILE KEY VALIDATION
      ( 0 )
```

User Enters:

      Depress RETURN

System Responds:

```
      ENTER KEY OPTION
      0 =NOT APPLICABLE
      1 =PRIMARY
      2 =SECONDARY
      ( 1 )
```

User Enters:

      0

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                            20
2: VARIABLE DESCRIPTION                      CUSTOMER ADDRESS
3: VARIABLE CODE NAME                        CUST.ADDR$
4: NUMBER OF OCCURRENCES                              0.
5: STORAGE FORMAT                            STRING
6: LENGTH OF FIELD                                   40.
7: LOW RANGE FOR VARIABLE                    NONE
8: HIGH RANGE FOR VARIABLE                   NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                      NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION                NO SELECTION
12: KEY OPTION                               NOT APPLICABLE
*: NUMBER OF ARRAY VALIDATION ENTRIES                0.
*: POSITION IN RECORD                                0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

      Depress RETURN

System Responds:

```
      Enter ESCAPE to terminate add processing, OR
      ENTER EDIT LINE SEQUENCE ( 30 )
```

User Enters:

Depress RETURN

The next data element will be LOCATION (i.e., City, State, Zip)

System Responds:

Depress RETURN

System Responds:

```
ENTER VARIABLE DESCRIPTION (CUSTOMER ADDRESS)
```

User Enters:

```
LOCATION
```

System Responds:

```
ENTER VARIABLE CODE NAME (CUST.ADDR$)
```

User Enters:

```
CUST.LOC
```

System Responds:

```
ENTER STORAGE FORMAT
0 =FLOATING POINT
1 =INTEGER
2 =STRING
3 =DATE
4 =MONEY
5 =COMPUTATIONAL
( 2 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER LENGTH OF FIELD ( 40 )
```

User Enters:

```
30
```

System Responds:

```
     ENTER EDIT CONTROL OPTION
     0 =NO EDITING RESTRICTIONS
     1 =EDIT DURING ADD ONLY
     2 =DISPLAY ONLY
     3 =SUPPRESS DISPLAY/EDIT
     ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER VALIDATION CONTROL OPTION
     0 =NO SELECTION
     1 =Y/N SELECTION
     2 =ALPHA SINGLE CHARACTER
     3 =PHRASE SELECTION
     4 =CROSS FILE KEY VALIDATION
     ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER KEY OPTION
     0 =NOT APPLICABLE
     1 =PRIMARY KEY
     2 =SECONDARY KEY
     ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                                30
2: VARIABLE DESCRIPTION                          CUSTOMER LOCATION
3: VARIABLE CODE NAME                            CUST.LOC$
4: NUMBER OF OCCURRENCES                                   0.
5: STORAGE FORMAT                                STRING
6: LENGTH OF FIELD                                        30.
7: LOW RANGE FOR VARIABLE                        NONE
8: HIGH RANGE FOR VARIABLE                       NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                          NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION                    NO SELECTION
12: KEY OPTION                                   NOT APPLICABLE
*: NUMBER OF ARRAY VALIDATION ENTRIES                     0.
*: POSITION IN RECORD                                     0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

The next data element to be defined is the PHONE.

System Responds:

```
Enter ESCAPE to terminate add processing, OR
ENTER EDIT LINE SEQUENCE ( 40 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VARIABLE DESCRIPTION (CUSTOMER LOCATION)
```

User Enters:

```
PHONE
```

System Responds:

```
ENTER VARIABLE CODE NAME (CUST.LOC$)
```

User Enters:

```
CUST.PHONE
```

System Responds:

```
ENTER STORAGE FORMAT
0 =FLOATING POINT
1 =INTEGER
2 =STRING
3 =DATE
4 =MONEY
5 =COMPUTATIONAL
( 2 )
```

User Enters:

Depress RETURN

System Responds:

```
 ENTER LENGTH OF FIELD ( 30 )
```

User Enters:

```
 12
```

System Responds:

```
ENTER EDIT CONTROL OPTION
0 =NO EDITING RESTRICTIONS
1 =EDIT DURING ADD ONLY
2 =DISPLAY ONLY
3 =SUPPRESS DISPLAY/EDIT
( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VALIDATION CONTROL OPTION
0 =NO SELECTION
1 =Y/N SELECTION
2 =ALPHA SINGLE CHARACTER
3 =PHRASE SELECTION
4 =CROSS FILE KEY VALIDATION
( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER KEY OPTION
     0 =NOT APPLICABLE
     1 =PRIMARY KEY
     2 =SECONDARY KEY
     ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                               40
2: VARIABLE DESCRIPTION                         PHONE
3: VARIABLE CODE NAME                           CUST.PHONE$
4: NUMBER OF OCCURRENCES                                  0.
5: STORAGE FORMAT                               STRING
6: LENGTH OF FIELD                                       12.
7: LOW RANGE FOR VARIABLE                       NONE
8: HIGH RANGE FOR VARIABLE                      NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                         NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION                   NO SELECTION
12: KEY OPTION                                  NOT APPLICABLE
*: NUMBER OF ARRAY VALIDATION ENTRIES                    0.
*: POSITION IN RECORD                                    0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
     Enter ESCAPE to terminate add processing, OR
     ENTER EDIT LINE SEQUENCE ( 50 )
```

User Enters:

Depress RETURN

The next data element will be CONTACT DATE.

System Responds:

```
     ENTER VARIABLE DESCRIPTION (PHONE)
```

User Enters:

CONTACT DATE

System Responds:

```
     ENTER VARIABLE CODE NAME (CUST.PHONE$)
```

User Enters:

```
     CUST.DATE
```

System Responds:

```
     ENTER STORAGE FORMAT
     0 =FLOATING POINT
     1 =INTEGER
     2 =STRING
     3 =DATE
     4 =MONEY
     5 =COMPUTATIONAL
     ( 2 )
```

User Enters:

```
     3
```

(LENTH OF FIELD is not asked for because 6 digits is always assumed and is therefore the default.)

System Responds:

```
     ENTER EDIT CONTROL OPTION
     0 =NO EDITING RESTRICTIONS
     1 =EDIT DURING ADD ONLY
     2 =DISPLAY ONLY
     3 =SUPPRESS DISPLAY/EDIT
     ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER VALIDATION CONTROL OPTION
     0 =NO SELECTION
     1 =Y/N SELECTION
     2 =ALPHA SINGLE CHARACTER
     3 =ARRAY VALIDATION
     4 =CROSS FILE KEY VALIDATION
     ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER KEY OPTION
     0 =NOT APPLICABLE
     1 =PRIMARY KEY
     2 =SECONDARY KEY
     ( 0 )
```

User Enters:

```
     2
```

By choosing 2 (Secondary Key) you will be able get reports of the customers on the file by contact date. Step 10, Running Your PEARL Generated System will show you how to do this.

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                          50
2: VARIABLE DESCRIPTION                    CONTACT DATE
3: VARIABLE CODE NAME                      CUST.DATE
4: NUMBER OF OCCURRENCES                             0.
5: STORAGE FORMAT                          DATE
6: LENGTH OF FIELD                                  6.
7: LOW RANGE FOR VARIABLE                  NONE
8: HIGH RANGE FOR VARIABLE                 NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                    NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION              NO SELECTION
12: KEY OPTION                             SECONDARY KEY
*: NUMBER OF ARRAY VALIDATION ENTRIES               0.
*: POSITION IN RECORD                               0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

The next data element will be BUSINESS GROUP.

System Responds:

```
     Enter ESCAPE to terminate add processing, OR
     ENTER EDIT LINE SEQUENCE (60)
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER VARIABLE DESCRIPTION (CONTACT DATE)
```

User Enters:

```
     BUSINESS GROUP
```

System Responds:

```
     ENTER VARIABLE CODE NAME (CUST.DATE)
```

User Enters:

```
     BUS.GP
```

System Responds:

```
     ENTER STORAGE FORMAT
     0 =FLOATING POINT
     1 =INTEGER
     2 =STRING
     3 =DATE
     4 =MONEY
     5 =COMPUTATIONAL
     ( 3 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER LENGTH OF FIELD ( 6 )
```

User Enters:

```
     2
```

System Responds:

```
     ENTER LOW RANGE FOR VARIABLE (NONE)*
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER HIGH RANGE FOR VARIABLE (NONE)*
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER EDIT MASK (     ##,###     )*
```

User Enters:

```
     Depress RETURN
```

NOTE

*These fields are used for items stored as integer or floating point such as money amounts where a high and low range may be desired, and where special editing (formatting masks) on output may be desired (see CBASIC2 manual for specifications on edit mask). They are not necessary in this example.

System Responds:

```
     ENTER EDIT CONTROL OPTION
     0 =NO EDITING RESTRICTIONS
     1 =EDIT DURING ADD ONLY
     2 =DISPLAY ONLY
     3 =SUPPRESS DISPLAY/EDIT
     ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER VALIDATION CONTROL OPTION
     0 =NO SELECTION
     1 =Y/N SELECTION
     2 =ALPHA SINGLE CHARACTER
     3 =PHRASE SELECTION
     4 =CROSS FILE KEY VALIDATION
     ( 0 )
```

User Enters:

```
     3
```

System Responds:

```
     ENTER KEY OPTION
     0 =NOT APPLICABLE
     1 =PRIMARY KEY
     2 =SECONDARY KEY
     ( 2 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
DATA ELEMENT DEFINITION DATA
   CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
1: LINE EDIT NUMBER                               60
2: VARIABLE DESCRIPTION                         BUSINESS GROUP
3: VARIABLE CODE NAME                           BUS.GP%
4: NUMBER OF OCCURRENCES                                  0.
5: STORAGE FORMAT                               INTEGER
6: LENGTH OF FIELD                                       2.
7: LOW RANGE FOR VARIABLE                       NONE
8: HIGH RANGE FOR VARIABLE                      NONE
9: EDIT MASK
10: EDIT CONTROL OPTION                         NO EDITING RESTRICTIONS
11: VALIDATION CONTROL OPTION                   PHRASE SELECTION
12: REPORT HEADING                              SECONDARY KEY
*: NUMBER OF ARRAY VALIDATION ENTRIES                    0.
*: POSITION IN RECORD                                    0.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
     Enter ESCAPE to terminate add processing, OR
     ENTER EDIT LINE SEQUENCE (70)
```

User Enters:

Depress ESCAPE and RETURN

System Responds:

```
                PEARL LEVEL 3 (A:A200) VERSION N.00
              FILE DATA ELEMENT DEFINITION-01/01/80
                   MINIMUM FREE SPACE=(NNNNN)

0   RETURN TO MAIN MENU
1   TO ADD NEW DATA ELEMENTS
2   TO EDIT EXISTING DATA ELEMENTS
3   TO DELETE DATA ELEMENTS
F=nn  TO RESET FILE ID
?
```

User Enters:

0

System Responds:

```
     ... RETURNING TO MAIN MENU ...
```

GO TO STEP 5

Step 5 - Phrase Selection Definition

Since phrase selection has been chosen for BUSINESS GROUP, the variables must now be defined for each distinct group within the BUSINESS GROUP data element.

System Responds:

```
                PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                    MAIN SELECTION MENU-MM/DD/YY
                     MINIMUM FREE SPACE=(NNNNN)
                              (FILE=01)
 0. RETURN TO CP/M
 1. SYSTEM INITIALIZATION
 2. FILE DEFINITION
 3. DATA ELEMENT DEFINITION
 4. PHRASE SELECTION DEFINITION
 5. MAIN MENU DEFINITION
 6. REPORT CONTROL DEFINITION
 7. REPORT DETAIL DEFINITION
 8. ENTER/EDIT POST/CLOSE COMPUTATIONS
 9. LIST DATA ELEMENT CONTROL DATA
10. LIST MENU CONTROL DATA
11. LIST REPORT CONTROL DATA
12. LIST POST/CLOSE COMPUTATION CONTROL DATA
13. VALIDATION OF CROSS FILE PROCESSES
14. EDIT SYSTEM DEFINITION DATA
15. SYSTEM GENERATION
16. RESET CURRENT SYSTEM DATE
17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     4
```

System Responds:

```
LOADING PROGRAM TO PHRASE SELECTION DEFINITION

                PEARL LEVEL 3 (A:A300) VERSION N.00
              PHRASE SELECTION DATA ADD/EDIT-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                             (FILE=01)

0 =TO RETURN TO MAIN MENU
1 =TO ADD OR EDIT PHRASES
F=nn  TO RESET FILE ID
?
```

User Enters:

```
     1
```

System Responds:

```
     ENTER EDIT SEQUENCE NUMBER FOR DATA ELEMENT
```

User Enters:

```
     60 (Business Group was 60)
```

System Responds:

```
     CUSTOMER CONTACT FILE--CUSTOMER CONTACT
     ARRAY VALIDATION FOR: BUSINESS GROUP
     ENTER VALIDATION DATA FOR VALUE = 0
```

User Enters:

```
     GOVERNMENT
```

System Responds:

```
     ENTER VALIDATION DATA VALUE = 1
```

User Enters:

```
     EDUCATION
```

System Responds:

```
     ENTER VALIDATION DATA VALUE = 2
```

User Enters:

```
     CONTRACTING
```

System Responds:

```
     ENTER VALIDATION DATA VALUE = 3
```

User Enters:

```
     FARMING
```

System Responds:

```
     ENTER VALIDATION DATA VALUE = 4
```

User Enters:

```
     RETAIL SALES
```

System Responds:

```
     ENTER VALIDATION DATA VALUE = 5
```

User Enters:

```
WHOLESALE DIST.
```

System Responds:

```
ENTER VALIDATION DATA VALUE = 6
```

User Enters:

```
OTHER
```

System Responds:

```
ENTER VALIDATION DATA VALUE = 7
```

User Enters:

Depress RETURN

System Responds:

```
CUSTOMER CONTACT FILE--CUSTOMER CONTACT
ARRAY VALIDATION FOR: BUSINESS GROUP
 0: GOVERNMENT
 1: EDUCATION
 2: CONTRACTING
 3: FARMING
 4: RETAIL SALES
 5: WHOLESALE DIST.
 6: OTHER
ENTER LINE NUMBER TO EDIT
```

User Enters:

Depress RETURN

If you enter a number larger than the highest one shown, you will go into add mode starting with an "nn" value one higher than the last. To delete a phrase, select its line number to edit, then enter an asterisk as the new phrase and depress return. When this is done, all of the entries following the deleted entry will be resequenced to fill in the gap.

System Responds:

```
ENTER EDIT SEQUENCE NUMBER FOR DATA ELEMENT
```

User Enters:

Depress RETURN

System Responds:

```
               PEARL LEVEL 3 (A:A300) VERSION N.00
             PHRASE SELECTION DATA ADD/EDIT-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)

0 =TO RETURN TO MAIN MENU
1 =TO ADD OR EDIT PHRASES
F=nn  TO RESET FILE ID
?
```

User Enters:

```
     0
```

System Responds:

```
     ... RETURNING TO MAIN MENU ...
```

GO TO STEP 6

Step 6 - Main Menu Definition

System Responds:

```
                PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                    MAIN SELECTION MENU-MM/DD/YY
                     MINIMUM FREE SPACE=(NNNNN)
                             (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     5
```

System Responds:

```
LOADING PROGRAM TO MAIN MENU DEFINITION

                PEARL LEVEL 3 (A:A500) VERSION N.00
                MENU DEFINITION PROCESSING-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)

0  TO RETURN TO MAIN MENU
1  TO ADD MENU ENTRIES
2  TO EDIT EXISING MENU ENTRIES
3  TO DELETE A MENU ENTRY
4  TO CREATE DEFAULT MENU CONTROL
?
```

User Enters:

```
     4
```

System Responds:

PEARL will now define the default main menu. When this is done, PEARL will respond with the menu listed above:

```
                PEARL LEVEL 3 (A:A500) VERSION N.00
                MENU DEFINITION PROCESSING-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)

0  TO RETURN TO MAIN MENU
1  TO ADD MENU ENTRIES
2  TO EDIT EXISING MENU ENTRIES
3  TO DELETE A MENU ENTRY
4  TO CREATE DEFAULT MENU CONTROL
?
```

User Enters:

```
     0
```

System Responds:

```
     ... RETURNING TO MAIN MENU ...
```

NOTE:

If you wished to define you own main menu, you would choose Option 1. PEARL would then give you a series of prompts that would allow you to describe the options needed on your PEARL-generated application Main Selection Menu.

Briefly, the prompts would be:

```
     ENTER MENU SEQUENCE KEY NN
     ENTER PROGRAM DESCRIPTION
     ENTER PROGRAM
     ENTER DISK IDENTIFICATION
```

You would answer each prompt, the MENU CONTROL DATA would be displayed for editing, and then you would name the next selection option you desired.

GO TO STEP 7

Step 7 - Report Control Definition

To get a printed report of the records on your file once your system it is up and running, you must now define the parameters (tab positions, headings, etc) for the placement of the text as you wish it to look when your report prints out. To do this, you must figure out the width of your page and the longest line in each column so that the text will fit across the page. You must know the PEARL Generated Variable Codes for report definition (Refer to Appendix G for the variable names you must use) as well as the variable code names you assigned during Data Element Definition, Step 4.

There will be two reports. The following layouts were derived before the data was entered into PEARL.

1. The page will hold 80 characters of text across it. Therefore, you have 80-characters of space to work with horizontally.

2. Variable names to be used on Report 1 are: CC.REPORT.ID$, CC.INSTALL$, CC.PAGE%, CC.DATE$, CUST.NAME$, CUST.ADDR$, CUST.LOC$, CUST.PHONE$, FN.DATE$(CUST.DATE), and BUS.GP.VAL$(BUS.GP%).

3. Variable names to be used on Report 2 are: CC.REPORT.ID$, CC.INSTALL$, CC.PAGE%, CC.DATE$, BUS.GROUP.MTDT, BUS.GP%, BUS.GP.X%, BUS.GP.VAL$(BUS.GP.X%), CONTROL.DATE, STOT.BUS.GROUP.MTDT, STOT.BUS.GROUP.YTDT, TOT.BUS.GROUP.MTDT, TOT.BUS.GROUP.YTDT.

4. You will need to know the tab positions for the layout of the text on each report.

5. You will want to know how many spaces should be left after each line.

Report 1 will look like this:

```
ıbtitle                          Company Name                          Page ###
                             Customer Contact List
                                  Current Date

                                                         Contact      Business
ıstomer Name                               Phone           Date         Group
-------------------------------------------------------------------------------
ıme                                     NNN-NNN-NNNN    MM/DD/YY  nnnnnnnnnnnnnnn
ldress
.ty and State

ıme                                     NNN-NNN-NNNN    MM/DD/YY  nnnnnnnnnnnnnnn
ldress
.ty and State

:c.
```

1. The string of dashes and the text above the dashes is HEADER information (SEQUENCE KEY 1000-1999).

2. The text below the dashes is DETAIL information (SEQUENCE KEY 5000-5999).

3. The longest word in column 1 will be 40.

4. The longest word in column 2 will be 12.

5. The longest word in column 3 will be 8.

6. The longest word in column 4 will be 15.

7. Column 1 = 1 through 40 character spaces.
   Column 2 = 42 through 54 character spaces.
   Column 3 = 56 through 64 character spaces.
   Column 4 = 66 through 80 character spaces.

To get your report to look like this, the following variables will be entered in the order shown:

Variable	Tab Position	Lines To Skip
CC.REPORT.ID$	1	0
CC.INSTALL$	0 (Center)	0
CC.PAGE% (Page ###)	73	1 (After)
Customer Contact List	0 (Center)	1 (After)
CC.DATE$	0 (Center)	2 (After)
Contact	56	0
Business	69	1 (After)
Customer Name	1	0
Phone	42	0
Date	57	0
Group	70	1 (After)
Line of Dashes (1 - 40, etc.)	1	0
Line of Dashes (41 - 80, etc.)	40	1 (After)
CUST.NAME$	1	0
CUST.PHONE$	42	0
FN.DATE$(CUST.DATE)	56	0
BUS.GP.VAL$(BUS.GP%)	66	1 (After)
CUST.ADDR$	1	0
CUST.LOC$	1	2 (After)

Report 2 will have subtotals, totals and grand totals:

```
Subtitle
                        Company Name                        Page ##
                    List by Business Group
                        Current Date

Business Group          Date Range                    Contact Totals
-----------------------------------------------------------------------------
0    Government          MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

1    Education           MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

2    Contracting         MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

3    Farming             MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

4    Retail Sales        MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

5    Wholesale Dist.     MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

6    Other               MM/DD/YY         Month-to-Date:
                                           Year-to-Date:

TOTAL MONTH-TO-DATE:

 TOTAL YEAR-TO-DATE:
```

1. Initialization computations will be entered for this report (SEQUENCE KEY 0001-0009).

2. The string of dashes and the text above the dashes is HEADER information (SEQUENCE KEY 1000-1999).

3. You will need to enter ACCUMULATION information (SEQUENCE KEYS 2000-2999.

4. You will need to enter CONTROL BREAKS information (SEQUENCE KEYS 3000-3999.

5. You will need to enter DETAIL COMPUTATION information (SEQUENCE KEYS 4000-4499).

6. You will need to enter SUBTOTAL and TOTAL information (SEQUENCE KEYS 6000-6999 and 7000-7999).

NOTE:

There is no DETAIL on this report, so sequence keys 5000-5999 are not used.

1. The longest word in column 1 will be 2.

2. The longest word in column 2 will be 15.

3. The longest word in column 2 will be 14.

4. The longest word in column 3 will be 18.

To get your report to look like this, the following variables will be entered in the order shown:

COMPUTATION CODES ARE:

```
REM
REM ENTER DATE RANGE
REM
GO SUP 90000
PRINT "ENTER THE STARTING DATE FOR MONTH-TO-DATE TOTALS"
GO SUB 82174     REM GET DATE
CONTROL.DATE=V
REM SET THE INITIAL COUNTER VALUES
BUS.GROUP.MTDT=1:BUS.GROUP.YTDT=1
REM
```

Variables/Computations	Tab Position	Lines To Skip
CC.REPORT.ID$	1	1 (After)
CC.INSTALL$	0 (Center)	0
CC.PAGE% (Page ###)	70	1 (After)
Contact List by Business Group	0 (Center)	1 (After)
CC.DATE$	0 (Center)	2 (After)
Business Group	5	0
Date Range	29	0
Contact Totals	60	1 (After)
Line of Dashes (5 - 40 etc.)	5	0
Line of Dashes (41 - 80, etc.)	40	1 (After)

```
BUS.GROUP.MTDT  (ACCUMULATION)
BUS.GP% (CONTROL BREAK)
BUS.GP.X%=BUS.GP% (DETAIL COMPUTATION)
IF CUST.DATE < CONTROL.DATE THEN\
BUS.GROUP.MTDT=0 ELSE BUS.GROUP.MTDT=1 (SUBTOTAL COMP.)
```

BUS.GP.X%	5	1 (Before)
BUS.GP.VAL$(BUS.GP.X%)	10	0
CONTROL.DATE	30	0
STOT.BUS.GROUP.MTDT	45	1 (After)
STOT.BUS.GROUP.YTDT	46	2 (After)
TOT.BUS.GROUP.MTDT	5	2 (After)
TOT.BUS.GROUP.YTDT	6	0 (After)

Step 7 - Report Control Definition

System Responds:

```
                PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                    MAIN SELECTION MENU-MM/DD/YY
                     MINIMUM FREE SPACE=(20064)
                             (FILE=01)
   0. RETURN TO CP/M
   1. SYSTEM INITIALIZATION
   2. FILE DEFINITION
   3. DATA ELEMENT DEFINITION
   4. PHRASE SELECTION DEFINITION
   5. MAIN MENU DEFINITION
   6. REPORT CONTROL DEFINITION
   7. REPORT DETAIL DEFINITION
   8. ENTER/EDIT POST/CLOSE COMPUTATIONS
   9. LIST DATA ELEMENT CONTROL DATA
  10. LIST MENU CONTROL DATA
  11. LIST REPORT CONTROL DATA
  12. LIST POST/CLOSE COMPUTATION DATA
  13. VALIDATION OF CROSS FILE PROCESSES
  14. EDIT SYSTEM DEFINITION DATA
  15. SYSTEM GENERATION
  16. RESET CURRENT SYSTEM DATE
  17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     6
```

System Responds:

```
LOADING PROGRAM TO REPORT CONTROL DEFINITION

                PEARL LEVEL 3 (A:A400) VERSION N.00
                REPORT CONTROL DEFINITION-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)

0  TO RETURN TO MAIN MENU
1  TO DEFINE A NEW REPORT
2  TO EDIT REPORT CONTROL DATA
3  TO CHANGE REPORT ID (CURRENT REPORT =01)
4  TO DELETE REPORT CONTROL FOR SPECIFIED REPORT
5  TO ENTER/EDIT REPORT DETAIL
?
```

User Enters:

```
     1
```

System Responds:

ENTER YOUR REPORT IDENTIFICATION (01)

User Enters:

Depress RETURN

System Responds:

ENTER PRIMARY FILE NUMBER (0)

User Enters:

1

System Responds:

ENTER SECONDARY FILE NUMBER (0)

User Enters:

Depress RETURN

This option is may be used if you have multiple files. For example, if you have a MASTER FILE and a HISTORICAL TRANSACTION FILE.

System Responds:

ENTER REPORT WIDTH (0)

User Enters:

80

System Responds:

ENTER REPORT DEPTH (0)

User Enters:

Depress RETURN (Default is 55 as defined in the System Configuration Control Data.)

System Responds:

ENTER PROCESS SORTED FILE (Y/N) (N)

User Enters:

Depress RETURN

System Responds

```
     ENTER REPORT SUBROUTINE LABEL ( 50000 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER REPORT DESCRIPTION
```

User Enters:

     Customer Contact List

System Responds:

```
REPORT CONTROL DATA
1: REPORT ID                    01
2: PRIMARY FILE NUMBER           1
3: SECONDARY FILE NUMBER         0
4: REPORT WIDTH                 80
5: REPORT DEPTH                  0
6: PROCESS SORTED FILE (Y/N)    N
7: REPORT SUBROUTINE LABEL      50000
8: REPORT DESCRIPTION           Customer Contact List
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
               PEARL LEVEL 3 (A:A400) VERSION N.00
                REPORT CONROL DEFINITION-MM/DD/YY
                  MINIMUM FREE SPACE=(NNNNN)
                          (FILE=01)

0  RETURN TO MAIN MENU
1  TO DEFINE A NEW REPORT
2  TO EDIT REPORT CONTROL DATA
3  TO CHANGE REPORT ID (CURRENT REPORT =01)
4  TO DELETE REPORT CONTROL FOR SPECIFIED REPORT
5  TO ENTER/EDIT REPORT DETAIL
?
```

User Enters:

     5

     (You may also select Option 7 on the Main Menu to enter this data.)

Step 7 - Report Control Definition

System Responds:

```
LOADING PROGRAM TO REPORT DETAIL DEFINITION

                   PEARL LEVEL 3 (A:A400A) VERSION N.00
                     REPORT DETAIL DEFINITION-MM/DD/YY
                        MINIMUM FREE SPACE=(NNNNN)
                                (FILE=01)

0  RETURN TO MAIN MENU
1  TO ADD ITEMS TO THE REPORT
2  TO EDIT REPORT ITEMS
3  TO DELETE ITEMS FROM THE REPORT
4  TO CHANGE THE REPORT ID (CURRENT REPORT =01)
5  TO PROCESS REPORT CONTROL DATA
?
```

User Enters:

```
     1
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY ( 11 )
```

User Enters:

```
     1010
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     CC.REPORT.ID$
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1010
2: VARIABLE CODE NAME           CC.REPORT.ID$
3: TAB POSITION                    1
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
```

ENTER SEQUENCE KEY (1020)

User Enters:

    Depress RETURN

System Responds:

    ENTER VARIABLE CODE NAME (CC.REPORT.ID$)

User Enters:

    CC.INSTALL$

System Reponds:

    ENTER TAB POSITION (1)

User Enters:

    0

System Responds:

    ENTER FORMATTING CONTROL MASK ()

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (0)

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (0)

User Enters:

    Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1020
2: VARIABLE CODE NAME           CC.INSTALL$
3: TAB POSITION                   0
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1030)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.INSTALL$)
```

User Enters:

```
     CC.PAGE%
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     73
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

```
     Page ###
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    1030
2: VARIABLE CODE NAME             CC.PAGE%
3: TAB POSITION                     73
4: FORMATTING CONTROL MASK        Page ###
5: LINES TO SKIP BEFORE PRINT     0
6: LINES TO SKIP AFTER PRINT      1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1040)
```

User Enters:

```
     1110
```

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.PAGE%)
```

User Enters:

     Depress SPACE and RETURN

System Reponds:

```
     ENTER TAB POSITION ( 73 )
```

User Enters:

     0

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Page ###)
```

User Enters:

     Customer Master List

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

     Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1110
2: VARIABLE CODE NAME
3: TAB POSITION                  0
4: FORMATTING CONTROL MASK     Customer Master List
5: LINES TO SKIP BEFORE PRINT  0
6: LINES TO SKIP AFTER PRINT   1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1120)
```

User Enters:

```
     1210
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     CC.DATE$
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Customer Master List)
```

User Enters:

Enter SPACE and RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     2
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   1210
2: VARIABLE CODE NAME            CC.DATE$
3: TAB POSITION                     0
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     2
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1220)
```

User Enters:

```
     1310
```

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.DATE$)
```

User Enters:

Depress SPACE and RETURN

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     56
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

```
     Contact
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 2 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   1310
2: VARIABLE CODE NAME
3: TAB POSITION                   56
4: FORMATTING CONTROL MASK       Contact
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1320)
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     Depress RETURN
```

System Reponds:

```
     ENTER TAB POSITION ( 56 )
```

User Enters:

```
     69
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Contact)
```

User Enters:

```
     Business
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1320
2: VARIABLE CODE NAME
3: TAB POSITION                  69
4: FORMATTING CONTROL MASK      Business
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1330)
```

User Enters:

```
     1410
```

System Responds:

```
     ENTER VARIABLE CODE NAME
```

User Enters:

```
     Depress RETURN
```

System Reponds:

```
     ENTER TAB POSITION ( 69 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Business)
```

User Enters:

```
     Customer Name
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1410
2: VARIABLE CODE NAME
3: TAB POSITION                  1
4: FORMATTING CONTROL MASK      Customer Name
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1420)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

     Depress RETURN

System Reponds:

```
     ENTER TAB POSITION ( 1 )
```

User Enters:

```
     42
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Customer Name)
```

User Enters:

```
     Phone
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1420
2: VARIABLE CODE NAME
3: TAB POSITION                  42
4: FORMATTING CONTROL MASK      Phone
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
```

ENTER SEQUENCE KEY (1430)

User Enters:

    Depress RETURN

System Responds:

    ENTER VARIABLE CODE NAME ()

User Enters:

    Depress RETURN

System Reponds:

    ENTER TAB POSITION (42)

User Enters:

    57

System Responds:

    ENTER FORMATTING CONTROL MASK (Phone)

User Enters:

    Date

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (0)

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (0)

User Enters:

    Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1430
2: VARIABLE CODE NAME
3: TAB POSITION                   57
4: FORMATTING CONTROL MASK      Date
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1440)
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

Depress RETURN

System Reponds:

```
     ENTER TAB POSITION ( 57 )
```

User Enters:

```
     70
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Date)
```

User Enters:

```
     Group
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1440
2: VARIABLE CODE NAME
3: TAB POSITION                  70
4: FORMATTING CONTROL MASK      Group
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1450)
```

User Enters:

```
     1510
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

     Depress RETURN

System Reponds:

```
     ENTER TAB POSITION ( 70 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Group)
```

User Enters:

```
     ----------------------------------------
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    1510
2: VARIABLE CODE NAME
3: TAB POSITION                    1
4: FORMATTING CONTROL MASK       ----------------------------------------
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1520)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

    Depress RETURN

System Reponds:

```
     ENTER TAB POSITION ( 1 )
```

User Enters:

```
     41
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (----------------------------------------)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1520
2: VARIABLE CODE NAME
3: TAB POSITION                  41
4: FORMATTING CONTROL MASK      ----------------------------------------
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1530)
```

You will now begin entry of DETAIL information.

User Enters:

```
     5010
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     CUST.NAME$
```

System Reponds:

```
     ENTER TAB POSITION ( 41 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (----------------------------------------)
```

User Enters:

     Depress SPACE and RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  5010
2: VARIABLE CODE NAME          CUST.NAME$
3: TAB POSITION                  1
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
```

ENTER SEQUENCE KEY (5020)

User Enters:

    Depress RETURN

System Responds:

    ENTER VARIABLE CODE NAME (CUST.NAME$)

User Enters:

    CUST.PHONE$

System Reponds:

    ENTER TAB POSITION (1)

User Enters:

    42

System Responds:

    ENTER FORMATTING CONTROL MASK ()

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (0)

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (0)

User Enters:

    Depress RETURN

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   5020
2: VARIABLE CODE NAME           CUST.PHONE$
3: TAB POSITION                   42
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (5030)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME (CUST.PHONE$)
```

User Enters:

```
     FN.DATE$(CUST.DATE)
```

System Reponds:

```
     ENTER TAB POSITION ( 42 )
```

User Enters:

```
     56
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

Depress RETURN

System Responds:

```
ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    5030
2: VARIABLE CODE NAME            FN.DATE$(CUST.DATE)
3: TAB POSITION                    56
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (5040)
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME (FN.DATE$(CUST.DATE))
```

User Enters:

```
     BUS.GP.VAL$(BUS.GP%)
```

System Reponds:

```
     ENTER TAB POSITION ( 56 )
```

User Enters:

```
     66
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  5040
2: VARIABLE CODE NAME          BUS.GP.VAL$(BUS.GP%)
3: TAB POSITION                  66
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (5050)
```

User Enters:

```
    5110
```

System Responds:

```
    ENTER VARIABLE CODE NAME (BUS.GP.VAL$(BUS.GP%))
```

User Enters:

```
    CUST.ADDR$
```

System Reponds:

```
    ENTER TAB POSITION ( 66 )
```

User Enters:

```
    1
```

System Responds:

```
    ENTER FORMATTING CONTROL MASK ()
```

User Enters:

    Depress RETURN

System Responds:

```
    ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
    ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

Depress RETURN

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  5110
2: VARIABLE CODE NAME          CUST.ADDR$
3: TAB POSITION                   1
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (5110)
```

User Enters:

5210

System Responds:

```
ENTER VARIABLE CODE NAME (CUST.ADDR$)
```

User Enters:

CUST.LOC$

System Reponds:

```
ENTER TAB POSITION ( 1 )
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     2
```

System Responds:

```
DETAIL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   5210
2: VARIABLE CODE NAME           CUST.LOC$
3: TAB POSITION                    1
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     2
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

You are now done defining the report parameters. If you made an error while entering the report definition and you did not correct it when the fields were displayed for editing, you must go to Option 3 of the REPORT DETAIL DEFINITION menu, TO DELETE ITEMS FROM THE REPORT, and delete these entries by their sequence number. You will then return to Option 1, TO ADD ITEMS TO THE REPORT, and enter the items correctly.

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (5220)
```

User Enters:

Depress ESCAPE and RETURN

System Responds:

```
               PEARL LEVEL 3 (A:A400A) VERSION N.00
                 REPORT DETAIL DEFINITION-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                             (FILE=01)

0  RETURN TO MAIN MENU
1  TO ADD ITEMS TO THE REPORT
2  TO EDIT REPORT ITEMS
3  TO DELETE ITEMS FROM THE REPORT
4  TO CHANGE THE REPORT ID (CURRENT REPORT =01)
5  TO PROCESS REPORT CONTROL DATA
?
```

You will now define Report 2.

User Enters:

5

Step 7 - Report Control Definition

System Responds:

```
LOADING PROGRAM TO REPORT CONTROL DEFINITION

                 PEARL LEVEL 3 (A:A400) VERSION N.00
                 REPORT CONTROL DEFINITION-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                           (FILE=01)

0  TO RETURN TO MAIN MENU
1  TO DEFINE A NEW REPORT
2  TO EDIT REPORT CONTROL DATA
3  TO CHANGE REPORT ID (CURRENT REPORT =01)
4  TO DELETE REPORT CONTROL FOR SPECIFIED REPORT
5  TO ENTER/EDIT REPORT DETAIL
?
```

User Enters:

```
    1
```

System Responds:

```
    ENTER YOUR REPORT IDENTIFICATION (01)
```

User Enters:

```
    2
```

System Responds:

```
    ENTER PRIMARY FILE NUMBER ( 0 )
```

User Enters:

```
    1
```

System Responds:

```
    ENTER SECONDARY FILE NUMBER ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
    ENTER REPORT WIDTH ( 0 )
```

User Enters:

```
    80
```

System Responds:

```
     ENTER REPORT DEPTH ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER PROCESS SORTED FILE (Y/N) (N)
```

User Enters:

```
     Depress RETURN
```

System Responds

```
     ENTER REPORT SUBROUTINE LABEL ( 50000 )
```

User Enters:

```
     51000
```

System Responds:

```
     ENTER REPORT DESCRIPTION
```

User Enters:

```
     List by Business Group (2K2)
```

System Responds:

```
REPORT CONTROL DATA
1: REPORT ID                 02
2: PRIMARY FILE NUMBER         1
3: SECONDARY FILE NUMBER       0
4: REPORT WIDTH               80
5: REPORT DEPTH                0
6: PROCESS SORTED FILE (Y/N) N
7: REPORT SUBROUTINE LABEL   51000
8: REPORT DESCRIPTION        List By Business Group
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
                    PEARL LEVEL 3 (A:A400) VERSION N.00
                     REPORT CONROL DEFINITION-MM/DD/YY
                        MINIMUM FREE SPACE=(NNNNN)
                                 (FILE=01)

0   RETURN TO MAIN MENU
1   TO DEFINE A NEW REPORT
2   TO EDIT REPORT CONTROL DATA
3   TO CHANGE REPORT ID (CURRENT REPORT =01)
4   TO DELETE REPORT CONTROL FOR SPECIFIED REPORT
5   TO ENTER/EDIT REPORT DETAIL
?
```

User Enters:

```
     5
```

Step 7 - Report Control Definition

System Responds:

```
LOADING PROGRAM FOR REPORT DETAIL DEFINITION

                 PEARL LEVEL 3 (A:A400A) VERSION N.00
                   REPORT DETAIL DEFINITION-MM/DD/YY
                      MINIMUM FREE SPACE=(NNNNN)
                               (FILE=01)

0  RETURN TO MAIN MENU
1  TO ADD ITEMS TO THE REPORT
2  TO EDIT REPORT ITEMS
3  TO DELETE ITEMS FROM THE REPORT
4  TO CHANGE THE REPORT ID (CURRENT REPORT =01)
5  TO PROCESS REPORT CONTROL DATA
?
```

User Enters:

```
     4
```

System Responds:

```
ENTER YOUR REPORT IDENTIFICATION (01)
```

User Enters:

```
     2
```

System Responds:

```
                 PEARL LEVEL 3 (A:A400A) VERSION N.00
                   REPORT DETAIL DEFINITION-MM/DD/YY
                      MINIMUM FREE SPACE=(NNNNN)
                               (FILE=01)

0  RETURN TO MAIN MENU
1  TO ADD ITEMS TO THE REPORT
2  TO EDIT REPORT ITEMS
3  TO DELETE ITEMS FROM THE REPORT
4  TO CHANGE THE REPORT ID (CURRENT REPORT =02)
5  TO PROCESS REPORT CONTROL DATA
?
```

User Enters:

```
     1
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY ( 11 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     REM
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        1
2: COMPUTATION CODE    REM
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY ( 11 )
```

User Enters:

```
     10
```

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     REM ENTER DATE RANGE
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY       10
2: COMPUTATION CODE   REM ENTER DATE RANGE
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0020)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     REM
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY       20
2: COMPUTATION CODE   REM
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0030)
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     GOSUB 90000
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        30
2: COMPUTATION CODE    GOSUB 90000
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0040)
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     PRINT "ENTER THE STARTING DATE FOR MONTH-TO-DATE TOTALS"
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        40
2: COMPUTATION CODE    PRINT "ENTER THE STARTING DATE FOR MONTH-TO-DATE TOTA
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
C000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0050)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     GOSUB 82174     REM GET DATE
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        50
2: COMPUTATION CODE    GOSUB 82174     REM GET DATE
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0060)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     CONTROL.DATE=V
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        60
2: COMPUTATION CODE    CONTROL.DATE=V
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0070)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     REM SET THE INITIAL COUNTER VALUES
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY         70
2: COMPUTATION CODE     REM SET THE INITIAL COUNTER VALUES
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0080)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     BUS.GROUP.MTDT=1:BUS.GROUP.YTDT=1
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY         80
2: COMPUTATION CODE     BUS.GROUP.MTDT=1:BUS.GROUP.YTDT=1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (0090)
```

User Enters:

    Depress RETURN

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     REM
```

System Responds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        90
2: COMPUTATION CODE    REM
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY ( 100)
```

User Enters:

```
     1010
```

System Responds:

```
     ENTER VARIABLE CODE NAME ( )
```

User Enters:

```
     CC.REPORT.ID$
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     5
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (REM)
```

User Enters:

     Depress SPACE and RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1010
2: VARIABLE CODE NAME          CC.REPORT.ID$
3: TAB POSITION                   5
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   1
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1020)
```

User Enters:

```
     1110
```

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.REPORT.ID$)
```

User Enters:

```
     CC.INSTALL$
```

System Reponds:

```
     ENTER TAB POSITION ( 5 )
```

User Enters:

```
     0
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   1110
2: VARIABLE CODE NAME            CC.INSTALL$
3: TAB POSITION                     0
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT     0
6: LINES TO SKIP AFTER PRINT      0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1120)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.INSTALL$)
```

User Enters:

```
     CC.PAGE%
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     70
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

```
     Page ###
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1120
2: VARIABLE CODE NAME           CC.PAGE%
3: TAB POSITION                   70
4: FORMATTING CONTROL MASK      Page ###
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1130)
```

User Enters:

    1210

System Responds:

    ENTER VARIABLE CODE NAME (CC.PAGE%)

User Enters:

    Depress SPACE and RETURN

System Reponds:

    ENTER TAB POSITION (70)

User Enters:

    0

System Responds:

    ENTER FORMATTING CONTROL MASK (Page ###)

User Enters:

    Contact List by Business Group

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (0)

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (1)

User Enters:

    Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1210
2: VARIABLE CODE NAME
3: TAB POSITION                  0
4: FORMATTING CONTROL MASK     Contact List by Business Group
5: LINES TO SKIP BEFORE PRINT  0
6: LINES TO SKIP AFTER PRINT   1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1220)
```

User Enters:

```
     1310
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

```
     CC.DATE$
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Contact List by Business Group)
```

User Enters:

```
     Enter SPACE and RETURN
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     2
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   1310
2: VARIABLE CODE NAME            CC.DATE$
3: TAB POSITION                     0
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     2
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1320)
```

User Enters:

```
     1410
```

System Responds:

```
     ENTER VARIABLE CODE NAME (CC.DATE$)
```

User Enters:

     Depress SPACE and RETURN

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     5
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK ()
```

User Enters:

```
     Business Group
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 2 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  1410
2: VARIABLE CODE NAME
3: TAB POSITION                    5
4: FORMATTING CONTROL MASK      Business Group
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
```

ENTER SEQUENCE KEY (1420)

User Enters:

    Depress RETURN

System Responds:

    ENTER VARIABLE CODE NAME ()

User Enters:

    Depress RETURN

System Reponds:

    ENTER TAB POSITION (5)

User Enters:

    29

System Responds:

    ENTER FORMATTING CONTROL MASK (Business Group)

User Enters:

    Date Range

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (0)

User Enters:

    Depress RETURN

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (0)

User Enters:

    Depress RETURN

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1420
2: VARIABLE CODE NAME
3: TAB POSITION                  29
4: FORMATTING CONTROL MASK      Date Range
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1430)
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VARIABLE CODE NAME
```

User Enters:

Depress RETURN

System Reponds:

```
ENTER TAB POSITION ( 29 )
```

User Enters:

60

System Responds:

```
ENTER FORMATTING CONTROL MASK ( Date Range )
```

User Enters:

```
     Contact Totals
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1430
2: VARIABLE CODE NAME
3: TAB POSITION                 60
4: FORMATTING CONTROL MASK     Contact Totals
5: LINES TO SKIP BEFORE PRINT  0
6: LINES TO SKIP AFTER PRINT   1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1440)
```

User Enters:

```
     1510
```

System Responds:

```
     ENTER VARIABLE CODE NAME ()
```

User Enters:

     Depress RETURN

System Reponds:

```
     ENTER TAB POSITION ( 60 )
```

User Enters:

```
     5
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Contact Totals)
```

User Enters:

```
     ------------------------------------
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
     0
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1510
2: VARIABLE CODE NAME
3: TAB POSITION                 5
4: FORMATTING CONTROL MASK     ------------------------------------
5: LINES TO SKIP BEFORE PRINT  0
6: LINES TO SKIP AFTER PRINT   0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1520)
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VARIABLE CODE NAME ()
```

User Enters:

Depress RETURN

System Reponds:

```
ENTER TAB POSITION ( 5 )
```

User Enters:

```
41
```

System Responds:

```
ENTER FORMATTING CONTROL MASK (-------------------------------------)
```

User Enters:

```
----------------------------------------
```

System Responds:

```
ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
HEADING/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                 1520
2: VARIABLE CODE NAME
3: TAB POSITION                 41
4: FORMATTING CONTROL MASK     ----------------------------------------
5: LINES TO SKIP BEFORE PRINT  0
6: LINES TO SKIP AFTER PRINT   1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (1530)
```

User Enters:

```
     2000
```

System Responds:

```
     ENTER VARIABLE TO BE TOTALED ( )
```

User Enters:

```
     BUS.GROUP.MTDT
```

System Reponds:

```
     ENTER ACCUMULATE SUBTOTAL (Y/N) (N)
```

User Enters:

```
     Y
```

System Responds:

```
     ENTER ACCUMULATE GRAND TOTAL (Y)
```

User Enters:

     Depress RETURN

System Responds:

```
TOTAL ACCUMULATION CONTROL DATA
1: SEQUENCE CONTROL                2000
2: VARIABLE TO BE TOTALED         BUS.GROUP.MTDT
3: ACCUMULATE SUBTOTAL (Y/N)       Y
4: ACCUMULATE GRAND TOTAL (Y/N)    Y
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (2010)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER VARIABLE TO BE TOTALED (BUS.GROUP.MTDT)
```

User Enters:

```
     BUS.GROUP.YTDT
```

System Reponds:

```
     ENTER ACCUMULATE SUBTOTAL (Y/N) (Y)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER ACCUMULATE GRAND TOTAL (Y)
```

User Enters:

     Depress RETURN

System Responds:

```
TOTAL ACCUMULATION CONTROL DATA
1: SEQUENCE CONTROL                2010
2: VARIABLE TO BE TOTALED         BUS.GROUP.YTDT
3: ACCUMULATE SUBTOTAL (Y/N)       Y
4: ACCUMULATE GRAND TOTAL (Y/N)    Y
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (2020)
```

User Enters:

```
     3000
```

System Responds:

```
     ENTER VARIABLE NAME FOR CONTROL BREAK (BUS.GROUP.YTDT)
```

User Enters:

```
     BUS.GP%
```

System Responds:

```
     ENTER RECORD POSITION FOR CONTROL BREAK
```

User Enters:

```
     0
```

System Reponds:

```
     ENTER LENGTH OF CONTROL FIELD ( 41 )
```

User Enters:

```
     0
```

System Responds:

```
CONTROL BREAK CONTROL DATA
1: SEQUENCE                             3000
2: VARIABLE NAME FOR CONTROL BREAK      BUS.GP%
3: RECORD POSITION FOR CONTROL BREAK    0
4: LENGTH OF CONTROL FIELD              0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (3010)
```

User Enters:

```
     4000
```

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     BUS.GP.X%=BUS.GP%
```

System Reponds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY         4000
2: COMPUTATION CODE     BUS.GP.X%=BUS.GP%
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (4010)
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     IF CUST.DATE < CONTROL.DATE THEN\
```

System Reponds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY     4010
2: COMPUTATION CODE  IF CUST.DATE < CONTROL.DATE THEN\
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (4020)
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER COMPUTATION CODE ?
```

User Enters:

```
     BUS.GROUP.MTDT=0 ELSE BUS.GROUP.MTDT=1
```

System Reponds:

```
COMPUTATION CONTROL DATA
1: SEQUENCE KEY        4020
2: COMPUTATION CODE    BUS.GROUP.MTDT=0 ELSE BUS.GROUP.MTDT=1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (4030)
```

User Enters:

```
     6000
```

System Responds:

```
    ENTER VARIABLE CODE NAME (BUS.GP%)
```

User Enters:

```
     BUS.GP.X%
```

System Reponds:

```
     ENTER TAB POSITION ( 0 )
```

User Enters:

```
     5
```

System Responds:

```
     (BUS.GROUP.MTDT=0 ELSE BUS.GROUP.MTDT=1)
```

User Enters:

```
     ##
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

     Depress RETURN

```
SUBTOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                     6000
2: VARIABLE CODE NAME             BUS.GP.X%
3: TAB POSITION                        5
4: FORMATTING CONTROL MASK        ##
5: LINES TO SKIP BEFORE PRINT      1
6: LINES TO SKIP AFTER PRINT       0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (6010)
```

User Enters:

     Depress RETURN

System Responds:

    ENTER VARIABLE CODE NAME (BUS.GP.X%)

User Enters:

    BUS.GP.VAL$(BUS.GP.X%)

System Reponds:

    ENTER TAB POSITION (5)

User Enters:

    10

System Responds:

    ENTER FORMATTING CONTROL MASK (##)

User Enters:

    Depress SPACE and RETURN

System Responds:

    ENTER LINES TO SKIP BEFORE PRINT (1)

User Enters:

    0

System Responds:

    ENTER LINES TO SKIP AFTER PRINT (0)

User Enters:

    Depress RETURN

System Responds:

```
SUBTOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  6010
2: VARIABLE CODE NAME          BUS.GP.VAL$(BUS.GP.X%)
3: TAB POSITION                  10
4: FORMATTING CONTROL MASK
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (6020)
```

User Enters:

    Depress RETURN

System Responds:

```
ENTER VARIABLE CODE NAME (BUS.GP.VAL$(BUS.GP.X%))
```

User Enters:

```
CONTROL.DATE
```

System Reponds:

```
ENTER TAB POSITION ( 10 )
```

User Enters:

```
30
```

System Responds:

```
ENTER FORMATTING CONTROL MASK ()
```

User Enters:

```
##/##/##
```

System Responds:

```
ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

Depress RETURN

System Responds:

```
SUBTOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    6020
2: VARIABLE CODE NAME             CONTACT.DATE
3: TAB POSITION                    30
4: FORMATTING CONTROL MASK        ##/##/##
5: LINES TO SKIP BEFORE PRINT     0
6: LINES TO SKIP AFTER PRINT      0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (6030)
```

User Enters:

Depress RETURN

System Responds:

```
ENTER VARIABLE CODE NAME (CONTROL.DATE)
```

User Enters:

```
STOT.BUS.GROUP.MTDT
```

System Reponds:

```
ENTER TAB POSITION ( 30 )
```

User Enters:

```
45
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (##/##/##)
```

User Enters:

```
     Month-to-Date: ####
```

System Responds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 0 )
```

User Enters:

```
     1
```

System Responds:

```
SUBTOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    6030
2: VARIABLE CODE NAME             STOT.BUS.GROUP.MTDT
3: TAB POSITION                     1
4: FORMATTING CONTROL MASK        Month-to-Date: ####
5: LINES TO SKIP BEFORE PRINT    0
6: LINES TO SKIP AFTER PRINT     1
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

```
     Depress RETURN
```

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (6040)
```

User Enters:

```
    6110
```

System Responds:

```
    ENTER VARIABLE CODE NAME (STOT.BUS.GROUP.MTDT)
```

User Enters:

```
    STOT.BUS.GROUP.YTDT
```

System Reponds:

```
    ENTER TAB POSITION ( 45 )
```

User Enters:

```
    46
```

System Responds:

```
    ENTER FORMATTING CONTROL MASK (Month-to-Date: ####)
```

User Enters:

```
    Year-to-Date:#####
```

System Responds:

```
    ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

    Depress RETURN

System Responds:

```
    ENTER LINES TO SKIP AFTER PRINT ( 1 )
```

User Enters:

```
    2
```

System Responds:

```
SUBTOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                    6110
2: VARIABLE CODE NAME            STOT.BUS.GROUP.YTDT
3: TAB POSITION                    46
4: FORMATTING CONTROL MASK       Year-to-Date:#####
5: LINES TO SKIP BEFORE PRINT     0
6: LINES TO SKIP AFTER PRINT      2
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

    Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (6120)
```

User Enters:

```
     7000
```

System Responds:

```
     ENTER VARIABLE CODE NAME (STOT.BUS.GROUP.YTDT)
```

User Enters:

```
     TOT.BUS.GROUP.MTDT
```

System Responds:

```
     ENTER TAB POSITION ( 46 )
```

User Enters:

```
     5
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (Year-to-Date:#####)
```

User Enters:

```
     TOTAL MONTH-TO-DATE: ####
```

System Reponds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     0
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 2 )
```

User Enters:

     Depress RETURN

System Responds:

```
TOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                  7000
2: VARIABLE CODE NAME           TOT.BUS.GROUP.MTDT
3: TAB POSITION                    5
4: FORMATTING CONTROL MASK      TOTAL MONTH-TO-DATE: ####
5: LINES TO SKIP BEFORE PRINT     0
6: LINES TO SKIP AFTER PRINT      2
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (7010)
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER VARIABLE CODE NAME (TOT.BUS.GROUP.MTDT)
```

User Enters:

```
     TOT.BUS.GROUP.YTDT
```

System Responds:

```
     ENTER TAB POSITION ( 5 )
```

User Enters:

```
     6
```

System Responds:

```
     ENTER FORMATTING CONTROL MASK (TOTAL MONTH-TO-DATE: ####)
```

User Enters:

```
     TOTAL YEAR-TO-DATE:#####
```

System Reponds:

```
     ENTER LINES TO SKIP BEFORE PRINT ( 0 )
```

User Enters:

```
     0
```

System Responds:

```
     ENTER LINES TO SKIP AFTER PRINT ( 2 )
```

User Enters:

Depress RETURN

System Responds:

```
TOTAL/PRINT LINE CONTROL DATA
1: SEQUENCE KEY                   7010
2: VARIABLE CODE NAME           TOT.BUS.GROUP.YTDT
3: TAB POSITION                    6
4: FORMATTING CONTROL MASK      TOTAL YEAR-TO-DATE:#####
5: LINES TO SKIP BEFORE PRINT   0
6: LINES TO SKIP AFTER PRINT    0
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

You are now done defining the report parameters for report 2. If you made any errors while entering the report definitions and you did not correct them when the fields were displayed for editing, you must go to Option 3 of the REPORT DETAIL DEFINITION menu, to DELETE ITEMS FROM THE REPORT, and delete these entries by their sequence number. You will then return to Option 1, TO ADD ITEMS TO THE REPORT, and enter the items correctly.

System Responds:

```
0001-0999 INITIALIZATION COMPUTATIONS (rb000-rb020)
1000-1999 HEADER
2000-2999 ACCUMULATION
3000-3999 CONTROL BREAKS
4000-4499 DETAIL COMPUTATION (rb300-rb349)
4500-4799 SUBTOTAL COMPUTATION (rb350-rb379)
4800-4999 TOTAL COMPUTATION (rb380-rb399)
5000-5999 DETAIL
6000-6999 SUBTOTAL
7000-7999 TOTAL
ENTER SEQUENCE KEY (7020)
```

User Enters:

Depress ESCAPE RETURN

System Responds:

```
               PEARL LEVEL 3 (A:1400A) VERSION N.00
                 REPORT DETAIL DEFINITION-MM/DD/YY
                    MINIMUM FREE SPACE=(NNNNN)
                             (FILE=01)

0   RETURN TO MAIN MENU
1   TO ADD ITEMS TO THE REPORT
2   TO EDIT REPORT ITEMS
3   TO DELETE ITEMS FROM THE REPORT
4   TO CHANGE THE REPORT ID (CURRENT REPORT =01)
5   TO PROCESS REPORT CONTROL DATA
?
```

User Enters:

Depress RETURN

System Responds:

```
     ... RETURNING TO MAIN MENU ...
```

GO TO STEP 8

Step 8 - Listing Control Information

Before generating, compiling and running your application, it is a good idea to list the control data you have defined -- Options 9, 10, and 11 (and 12 if necessary). Once these listings are printed, you can then check them for possible errors. If you find any errors, you can correct them now instead of after you have tried to generate, compile and/or run your application. This will save time in the long run because, for one thing, generation and compilation can take quite awhile.

NOTE:

You must have a printer that will support 11 by 14 wide-form paper as some of these listings will print over 120 characters per line. If you have a printer that can reduce the line length, then this is also acceptable.

System Responds:

```
                PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                    MAIN SELECTION MENU-MM/DD/YY
                     MINIMUM FREE SPACE=(NNNNN)
                              (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     9
```

System Responds:

```
LOADING PROGRAM TO LIST DATA ELEMENT CONTROL DATA
```

```
                 PEARL LEVEL 3 (A:AP0001) VERSION N.00
              DATA ELEMENT DEFINITION REPORTING-MM/DD/YY
                      MINIMUM FREE SPACE=(NNNNN)
                               (FILE=01)
0  TO RETURN TO MAIN MENU
1  DETAIL LIST OF DATA ELEMENTS BY RECORD/EDIT POSITION
2  SUMMARY LIST OF DATA ELEMENTS BY VARIABLE NAME
3  GENERATE REPORT FOR ALL FILES
F=NN  TO RESET FILE ID
?
```

User Enters:

```
     3
```

System Responds:

The followings listings will be printed:

```
                PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)  PAGE    1
                       RECORD DATA ELEMENT DEFINITION
                CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
                                  09/30/80

SEQUENCE EDIT LINE DESCRIPTION

  10       1      CUSTOMER NAME                                   STRING
OCCURS:   0  CODE NAME: CUST.NAME$               EDIT: NO EDITING RESTRICTI
   POS:   3  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:  40 VALIDATION: NO SELECTION             HIGH:  NONE
                                           KEY OPTION:PRIMARY KEY


  20       2      CUSTOMER ADDRESS                                STRING
OCCURS:   0  CODE NAME: CUST.ADDR$               EDIT: NO EDITING RESTRICTI
   POS:  43  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:  40 VALIDATION: NO SELECTION             HIGH:  NONE

  30       3      LOCATION                                        STRING
OCCURS:   0  CODE NAME: CUST.LOC$                EDIT: NO EDITING RESTRICTI
   POS:  83  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:  30 VALIDATION: NO SELECTION             HIGH:  NONE

  40       4      PHONE                                           STRING
OCCURS:   0  CODE NAME: CUST.PHONE$              EDIT: NO EDITING RESTRICTI
   POS: 113  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:  12 VALIDATION: NO SELECTION             HIGH:  NONE

  50       5      CONTACT DATE                                    DATE
OCCURS:   0  CODE NAME: CUST.DATE                EDIT: NO EDITING RESTRICTI
   POS: 125  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:   6 VALIDATION: NO SELECTION             HIGH:  NONE
                                           KEY OPTION:SECONDARY KEY

  60       6      BUSINESS GROUP                                  INTEGER
OCCURS:   0  CODE NAME: BUS.GP%                  EDIT: NO EDITING RESTRICTI
   POS: 131  EDIT MASK:                    RANGES-LOW:  NONE
LENGTH:   2 VALIDATION: PHRASE SELECTION         HIGH:  NONE
                                           KEY OPTION:SECONDARY KEY
                  VALIDATE ON  0 = GOVERNMENT
                  VALIDATE ON  1 = EDUCATION
                  VALIDATE ON  2 = CONTRACTING
                  VALIDATE ON  3 = FARMING
                  VALIDATE ON  4 = RETAIL SALES
                  VALIDATE ON  5 = WHOLESALE DIST.
                  VALIDATE ON  6 = OTHER
                PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)  PAGE     2
                       RECORD DATA ELEMENT DEFINITION
                      RECORD DESCRIPTION: FILE CONTROL
        CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01) CONTROL RECORD 1
                                  09/30/80

SEQUENCE EDIT LINE DESCRIPTION

1001        1      SORT FLAG                                          FLOATING P
OCCURS:   0  CODE NAME: CFC.SORT                      EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   3 VALIDATION: NO SELECTION                   HIGH:  NONE

1003        1      LAST RECORD                                        FLOATING P
OCCURS:   0  CODE NAME: CFC.LAST                      EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   5 VALIDATION: NO SELECTION                   HIGH:  NONE

1004        1      LAST RECORD DELETED                                FLOATING P
OCCURS:   0  CODE NAME: CF.NEXT.DELETED               EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   5 VALIDATION: NO SELECTION                   HIGH:  NONE

1005        1      DELETED REC. COUNT                                 FLOATING P
OCCURS:   0  CODE NAME: CF.AVAIL.DELETED              EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   3 VALIDATION: NO SELECTION                   HIGH:  NONE

1006        1      UPDATE IN PROCESS FLAG                             FLOATING P
OCCURS:   0  CODE NAME: CF.UPDATE.FLAG                EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   1 VALIDATION: NO SELECTION                   HIGH:  NONE

1007        1      OPEN ERROR RESET COUNTER                           FLOATING P
OCCURS:   0  CODE NAME: CF.UPDATE.RESET               EDIT: DISPLAY ONLY
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   2 VALIDATION: NO SELECTION                   HIGH:  NONE

1008        1      INDEX BUFFERS                                      INTEGER
OCCURS:   0  CODE NAME: CF.IS.NBUFS%                  EDIT: NO EDITING RESTRICTI
   POS:   0  EDIT MASK:                          RANGES-LOW:  NONE
LENGTH:   2 VALIDATION: NO SELECTION                   HIGH:  NONE
```

```
                 PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)  PAGE    3
                        RECORD DATA ELEMENT DEFINITION
                 CUSTOMER CONTACT FILE--CUSTOMER CONTACT (01)
                                  09/30/80

SEQUENCE EDIT LINE DESCRIPTION

FILE DEFINITION DATA
 1: RELATIVE FILE ID                    1.
 2: FILE DESCRIPTION              CUSTOMER CONTACT
 3: ACCESS METHOD                 INDEXED
 4: FILE TYPE                     MISC. INDEXED FILE
 5: DATA FILE NAME                CUSTOMER
 6: UNIQUE FILE ID                CF.
 7: LOGICAL FILE UNIT                        2.
 8: DISK DRIVE UNIT ID            B
 9: I/O SUBROUTINE BASE           10000
10: DATA RECORD EDIT BASE         30000
11: CONTROL RECORD EDIT BASE      40000
*: KEY FIELD DESCRIPTION          CUSTOMER NAME
*: VARIABLE NAME FOR KEY FIELD    CUST.NAME$
*: KEY FIELD LENGTH                         40.
*: RECORD LENGTH                           136.
*: CONTROL RECORD COUNT                      1.
*: TOTAL DATA ELEMENT COUNT                 14.

CONTROL RECORD 1 LENGTH =   49

   0 ERRORS DETECTED
```

```
                              PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)                        PAGE   1
                                        DATA ELEMENT LIST BY FILE
                                          CUSTOMER CONTACT FILE
                                               09/30/80

VARIABLE NAME            DESCRIPTION                          FILE   TYPE       LEN SEQUENCE  EDIT CONTROL

BUS.GP%                  BUSINESS GROUP                        01  INTEGER       2   0060     NO EDITING RESTRICTIONS
CF.AVAIL.DELETED         DELETED REC. COUNT                    01  FLOATING      3   1005     DISPLAY ONLY
CF.IS.NBUFS%             INDEX BUFFERS                         01  INTEGER       2   1008     NO EDITING RESTRICTIONS
CF.NEXT.DELETED          LAST RECORD DELETED                   01  FLOATING      5   1004     DISPLAY ONLY
CF.UPDATE.FLAG           UPDATE IN PROCESS FLAG                01  FLOATING      1   1006     DISPLAY ONLY
CF.UPDATE.RESET          OPEN ERROR RESET COUNTER              01  FLOATING      2   1007     DISPLAY ONLY
CFC.LAST                 LAST RECORD                           01  FLOATING      5   1003     DISPLAY ONLY
CFC.SORT                 SORT FLAG                             01  FLOATING      3   1001     DISPLAY ONLY
CUST.ADDR$               CUSTOMER ADDRESS                      01  STRING       40   0020     NO EDITING RESTRICTIONS
CUST.DATE                CONTACT DATE                          01  DATE          6   0050     NO EDITING RESTRICTIONS
CUST.LOC$                LOCATION                              01  STRING       30   0030     NO EDITING RESTRICTIONS
CUST.NAME$               CUSTOMER NAME                         01  STRING       40   0010     NO EDITING RESTRICTIONS
CUST.PHONE$              PHONE                                 01  STRING       12   0040     NO EDITING RESTRICTIONS
```

Once the listings are done printing, the submenu will again be displayed, and you may choose another option from this menu, or return to the Main Selection Menu:

System Responds:

```
                  PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                      MAIN SELECTION MENU-MM/DD/YY
                       MINIMUM FREE SPACE=(NNNNN)
                               (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     10
```

System Responds:

```
LOADING PROGRAM FOR LIST MENU CONTROL DATA
                  PEARL LEVEL 3 (A:AP0003) VERSION N.00
                     MENU CONTROL REPORTING-MM/DD/YY
                       MINIMUM FREE SPACE=(NNNNN)
                               (FILE-01)

0   TO RETURN TO THE MAIN MENU
1   LIST MENU CONTROL INFORMATION
2   CREATE DEFAULT MENU CONTROL INFORMATION
?
```

User Enters:

```
     1
```

System Responds:

The following listing will be printed on your printer:

```
              PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)  PAGE    1
                           MENU CONTROL DATA
                              MAIN MENU
                              09/30/80

   SEQ.  PROMPT DESCRIPTION                                         PROGRAM   DISKETTE

1. 0-110 INITIALIZE CUSTOMER CONTACT FILE                           CF000        2

2. 0-120 EDIT CUSTOMER CONTACT CONTROL DATA                         CF000A       2

3. 0-130 UPDATE CUSTOMER CONTACT FILE                               CF100        1

4. 0-140 GENERATE CUSTOMER CONTACT REPORTS                          CFP0001      4

5. 0-640 COMPRESS CUSTOMER CONTACT FILE                             CF000R       2

6. 0-980 CHANGE SYSTEM DATE                                         DATE         0

7. 0-995 EDIT SYSTEM CONFIGURATION DATA                             CON          0
```

Once the listing is done printing, the submenu will again be displayed, choose another option from this menu, or return to the Main Selection Menu:

System Responds:

```
                  PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                      MAIN SELECTION MENU-MM/DD/YY
                       MINIMUM FREE SPACE=(NNNNN)
                                (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     11
```

System Responds:

```
                  PEARL LEVEL 3 (A:AP0002) VERSION N.00
            REPORT FORMAT CONTROL REPORT PROCESSING-MM/DD/YY
                       MINIMUM FREE SPACE=(NNNNN)
                                (FILE=01)

0    TO RETURN TO THE MAIN MENU
1    TO LIST REPORT CONTROL DATA FOR CURRENT REPORT
2    SELECT NEW REPORT ID (CURRENT REPORT =00)
3    TO LIST CONTROL FOR ALL REPORTS
F=nn  TO RESET FILE ID
?
```

User Enters:

```
     3
```

System Responds:

The following reports will be printed on your printer:

```
                              PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)                            PAGE    1
                                     CONTROL FOR REPORT NUMBER 01
                                              09/30/80

SEQ. VARIABLE NAME                TAB FORMATTING MASK                                   PRE SKIP   POST SKIP

1010 CC.REPORT.ID$                  1                                                      0          0
1020 CC.INSTALL$                    0                                                      0          0
1030 CC.PAGE%                      73 Page ###                                             0          1
1110                                0 Customer Master List                                 0          1
1210 CC.DATE$                       0                                                      0          2
1310                               56 Contact                                              0          0
1320                               69 Business                                             0          1
1410                                1 Customer Name                                        0          0
1420                               42 Phone                                                0          0
1430                               57 Date                                                 0          0
1440                               70 Group                                                0          1
1510                                1 ----------------------------------------             0          0
1520                               41 ----------------------------------------             0          1

SEQ. VARIABLE NAME                TAB FORMATTING MASK                                   PRE SKIP   POST SKIP

5010 CUST.NAME$                     1                                                      0          0
5020 CUST.PHONE$                   42                                                      0          0
5030 FN.DATE$(CUST.DATE)           56                                                      0          0
5040 BUS.GP.VAL$(BUS.GP%)          66                                                      0          1
5110 CUST.ADDR$                     1                                                      0          1
5210 CUST.LOC$                      1                                                      0          2
```

```
                                   PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)                         PAGE    1
                                         CONTROL FOR REPORT NUMBER 02
                                                   09/30/80

SEQ.          COMPUTATION                                                       TYPE

0001          REM                                                               INITIALIZATION
0010          REM ENTER DATE RANGE                                              INITIALIZATION
0020          REM                                                               INITIALIZATION
0030          GOSUB 90000                                                       INITIALIZATION
0040          PRINT "ENTER THE STARTING DATE FOR MONTH-TO-DATE TOTALS"          INITIALIZATION
0050          GOSUB 82174     REM GET DATE                                      INITIALIZATION
0060          CONTROL.DATE=V                                                    INITIALIZATION
0070          REM SET THE INITIAL COUNTER VALUES                                INITIALIZATION
0080          BUS.GROUP.MTDT=1:BUS.GROUP.YTDT=1                                 INITIALIZATION
0090          REM                                                               INITIALIZATION

SEQ. VARIABLE NAME              TAB FORMATTING MASK                                      PRE SKIP   POST SKIP

1010 CC.REPORT.ID$                5                                                         1          1
1110 CC.INSTALL$                  0                                                         0          0
1120 CC.PAGE%                    70 Page ###                                                0          1
1210                              0 Contact List by Business Group                          0          1
1310 CC.DATE$                     0                                                         0          2
1410                              5 Business Group                                          0          0
1420                             29 Date Range                                              0          0
1430                             60 Contact Totals                                          0          1
1510                              5 ------------------------------------                    0          0
1520                             41 ----------------------------------------                0          1

SEQ.          VARIABLE TO TOTAL              SUB     GRAND

2000          BUS.GROUP.MTDT                  Y        Y
2010          BUS.GROUP.YTDT                  Y        Y

SEQ.     VARIABLE FOR BREAK           POS     LEN

3000     BUS.GP%                       0       0

SEQ.          COMPUTATION                                                       TYPE

4000          BUS.GP.X%=BUS.GP%                                                 DETAIL
4010          IF CUST.DATE < CONTROL.DATE THEN\                                 DETAIL
4020          BUS.GROUP.MTDT=0 ELSE BUS.GROUP.MTDT=1                            DETAIL

SEQ. VARIABLE NAME              TAB FORMATTING MASK                                      PRE SKIP   POST SKIP

6000 BUS.GP.X%                    5 ##                                                      1          0
6010 BUS.GP.VAL$(BUS.GP.X%)      10                                                         0          0
6020 CONTROL.DATE                30 ##/##/##                                                0          0
6030 STOT.BUS.GROUP.MTDT         45 Month-to-Date: ####                                     0          1
6110 STOT.BUS.GROUP.YTDT         46 Year-to-Date:#####                                      0          2

SEQ. VARIABLE NAME              TAB FORMATTING MASK                                      PRE SKIP   POST SKIP

[illegible]00 TOT.BUS.GROUP.MTDT           5 TOTAL MONTH-TO-DATE: ####                               0          2
```

```
                    PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)
                           CONTROL FOR REPORT NUMBER 01
                                    09/30/80

REPORT CONTROL DATA
1: REPORT ID                 01
2: PRIMARY FILE NUMBER        1
3: SECONDARY FILE NUMBER      0
4: REPORT WIDTH              80
5: REPORT DEPTH               0
6: PROCESS SORTED FILE (Y/N) N
7: REPORT SUBROUTINE LABEL   50000
8: REPORT DESCRIPTION        Customer Contact File
```

```
                    PEARL LEVEL 3  (VERSION 3.00.25, SN ABC12345)
                           CONTROL FOR REPORT NUMBER 02
                                     09/30/80

SEQ. VARIABLE NAME           TAB FORMATTING MASK                                  PRE SKIP   POST SKIP

7010 TOT.BUS.GROUP.YTDT        6 TOTAL YEAR-TO-DATE:#####                             0          0
```

```
                    CONTROL FOR REPORT NUMBER 02
                              09/30/80

REPORT CONTROL DATA
1: REPORT ID                 02
2: PRIMARY FILE NUMBER        1
3: SECONDARY FILE NUMBER      0
4: REPORT WIDTH              80
5: REPORT DEPTH               0
6: PROCESS SORTED FILE (Y/N)  N
7: REPORT SUBROUTINE LABEL    51000
8: REPORT DESCRIPTION         List by Business Group (2K2)
```

Once the listings are done printing, the submenu will again be displayed. Choose another option from this menu, or return to the Main Selection Menu:

System Responds:

```
              PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                  MAIN SELECTION MENU-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                           (FILE=01)
  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

You should review your listings at this time to make sure there are no entry errors. If all the listings are as you wish, you may now generate and compile your application.

GO TO STEP 9

Step 9 - Generate and Compile

You are now ready to begin generation and compilation of the application you have defined.

System Responds:

```
                    PEARL LEVEL 3 (A:PEARL3) VERSION N.OO
                        MAIN SELECTION MENU-MM/DD/YY
                         MINIMUM FREE SPACE=(NNNNN)
                                 (FILE=01)

  0. RETURN TO CP/M
  1. SYSTEM INITIALIZATION
  2. FILE DEFINITION
  3. DATA ELEMENT DEFINITION
  4. PHRASE SELECTION DEFINITION
  5. MAIN MENU DEFINITION
  6. REPORT CONTROL DEFINITION
  7. REPORT DETAIL DEFINITION
  8. ENTER/EDIT POST/CLOSE COMPUTATIONS
  9. LIST DATA ELEMENT CONTROL DATA
 10. LIST MENU CONTROL DATA
 11. LIST REPORT CONTROL DATA
 12. LIST POST/CLOSE COMPUTATION DATA
 13. VALIDATION OF CROSS FILE PROCESSES
 14. EDIT SYSTEM DEFINITION DATA
 15. SYSTEM GENERATION
 16. RESET CURRENT SYSTEM DATE
 17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

Take out Definition Disk from Drive A.

Insert Generation Disk in Drive A.

Enter: 15

If you do not put the Generation Disk in Drive A before depressing 15, you will get the following message telling you to do so:

```
A:A900 CANNOT BE LOCATED
Place SYSTEM GENERATION DISKETTE on DRIVE A and
Press RETURN to continue, or
Press ESCAPE to return to main menu, or
Enter NAME of PROGRAM to be loaded
```

You would then place the Generation Disk in Drive A and depress RETURN.

System Responds:

```
LOADING PROGRAM FOR SYSTEM GENERATION

               PEARL LEVEL 3 (A:A900) VERSION N.00
               SYSTEM GENERATION CONTROL-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)
 0  TO RETURN TO MAIN MENU
 1  GENERATE ENTIRE SYSTEM (ALL FILES)
 2  GENERATE ENTIRE SYSTEM (CURRENT FILE ONLY)
 3  GENERATE I/O ROUTINES
 4  GENERATE DISPLAY/EDIT SUBROUTINES
 5  GENERATE REPORT ROUTINES
 6  GENERATE SYSTEM INITIALIZATION CONTROL UPDATE PROGRAM
 7  GENERATE FILE UPDATE/EDIT AND REPORT MAINLINE PROGRAMS
 8  GENERATE MENU SELECTION PROGRAM
 9  GENERATE POSTING & CLOSING ROUTINES
10  TO SET UP COMPILE SUBMIT FILE
F=NN  TO RESET FILE ID
?
```

User Enters:

```
     1
```

System Responds:

The following responses will be displayed on the video screen one line at a time. The process will take some time, so do not be concerned if your response is not immediate.

```
SETTING CONTROL FOR FILE 01

B:CF10000.BAS HAS BEEN PROCESSED
B:CFCREC1.BAS HAS BEEN PROCESSED
B:CF10000S.BAS HAS BEEN PROCESSED
B:CF30100.BAS HAS BEEN PROCESSED
B:CF40100.BAS HAS BEEN PROCESSED
B:CF30000.BAS HAS BEEN PROCESSED
B:CF40000.BAS HAS BEEN PROCESSED
B:CF50000.BAS HAS BEEN PROCESSED
B:CF51000.BAS HAS BEEN PROCESSED
B:CFINTL.BAS HAS BEEN PROCESSED
B:COMBCF.BAS HAS BEEN PROCESSED
B:CF10900.BAS HAS BEEN PROCESSED
B:CF000.BAS HAS BEEN PROCESSED
B:CF000A.BAS HAS BEEN PROCESSED
B:CF100.BAS HAS BEEN PROCESSED
B:CF000R.BAS HAS BEEN PROCESSED
B:CF29000.BAS HAS BEEN PROCESSED
B:CFP0001.BAS HAS BEEN PROCESSED
B:CONTACT.BAS HAS BEEN PROCESSED
2730 LINES OF CBASIC SOURCE CODE HAVE BEEN GENERATED.
```

Once generation has successfully completed, you may now compile your PEARL-generated system:

```
               PEARL LEVEL 3 (A:A900) VERSION N.00
               SYSTEM GENERATION CONTROL-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)
 0  TO RETURN TO MAIN MENU
 1  GENERATE ENTIRE SYSTEM (ALL FILES)
 2  GENERATE ENTIRE SYSTEM (CURRENT FILE ONLY)
 3  GENERATE I/O ROUTINES
 4  GENERATE DISPLAY/EDIT SUBROUTINES
 5  GENERATE REPORT ROUTINES
 6  GENERATE SYSTEM INITIALIZATION CONTROL UPDATE PROGRAM
 7  GENERATE FILE UPDATE/EDIT AND REPORT MAINLINE PROGRAMS
 8  GENERATE MENU SELECTION PROGRAM
 9  GENERATE POSTING & CLOSING ROUTINES
10  TO SET UP COMPILE SUBMIT FILE
F=NN  TO RESET FILE ID
?
```

User Enters:

```
     10
```

System Responds:

```
     COMPILE/SUBMIT SETUP CONTROL
     1 MAIN PROGRAM SOURCE DISK          B
     2 INTERMEDIATE FILE OUTPUT DISK     B
     3 CBASIC COMPILE DIRECTIVES         $$B
     4 SUBMIT FILE TYPE                  SUB
     5 FILE COPY COMMAND FILE NAME       PIP
     6 FILE COPY DIRECTIVE               [V]
     7 FILE COPY DIRECTIVE POSITION      PIP X.BAS=Y.BAS[V]
     ENTER LINE NUMBER TO EDIT OR RETURN TO CREATE SUBMIT FILE
```

User Enters:

```
     2
```

System Responds:

```
     ENTER INTERMEDIATE FILE OUTPUT DISK
```

User Enters:

```
     A
```

System Responds:

```
     COMPILE/SUBMIT SETUP CONTROL
     1 MAIN PROGRAM SOURCE DISK            B
     2 INTERMEDIATE FILE OUTPUT DISK       A
     3 CBASIC COMPILE DIRECTIVES           $$B
     4 SUBMIT FILE TYPE                    SUB
     5 FILE COPY COMMAND FILE NAME         PIP
     6 FILE COPY DIRECTIVE                 [V]
     7 FILE COPY DIRECTIVE POSITION        PIP X.BAS=Y.BAS[V]
     ENTER LINE NUMBER TO EDIT OR RETURN TO CREATE SUBMIT FILE
```

User Enters:

     Depress RETURN

System Responds:

```
     B:COMPILE.SUB HAS BEEN CREATED
     EXIT TO CPM. THEN ENTER
                   SUBMIT B:COMPILE
     TO COMPILE YOUR GENERATED SYSTEM

     Press RETURN to continue
```

User Enters:

     Depress RETURN

System Responds:

```
               PEARL LEVEL 3 (A:A900) VERSION N.00
               SYSTEM GENERATION CONTROL-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                            (FILE=01)
 0  TO RETURN TO MAIN MENU
 1  GENERATE ENTIRE SYSTEM (ALL FILES)
 2  GENERATE ENTIRE SYSTEM (CURRENT FILE ONLY)
 3  GENERATE I/O ROUTINES
 4  GENERATE DISPLAY/EDIT SUBROUTINES
 5  GENERATE REPORT ROUTINES
 6  GENERATE SYSTEM INITIALIZATION CONTROL UPDATE PROGRAM
 7  GENERATE FILE UPDATE/EDIT AND REPORT MAINLINE PROGRAMS
 8  GENERATE MENU SELECTION PROGRAM
 9  GENERATE POSTING & CLOSING ROUTINES
10  TO SET UP COMPILE SUBMIT FILE
F=NN  TO RESET FILE ID
?
```

User Enters:

```
     Take Generation Disk from Drive A.

     Insert Definition Disk in Drive A.

     Enter: 0
```

System Responds:

```
     ... RETURNING TO MAIN MENU ...

              PEARL LEVEL 3 (A:PEARL3) VERSION N.00
                  MAIN SELECTION MENU-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)
                           (FILE=01)
   0. RETURN TO CP/M
   1. SYSTEM INITIALIZATION
   2. FILE DEFINITION
   3. DATA ELEMENT DEFINITION
   4. PHRASE SELECTION DEFINITION
   5. MAIN MENU DEFINITION
   6. REPORT CONTROL DEFINITION
   7. REPORT DETAIL DEFINITION
   8. ENTER/EDIT POST/CLOSE COMPUTATIONS
   9. LIST DATA ELEMENT CONTROL DATA
  10. LIST MENU CONTROL DATA
  11. LIST REPORT CONTROL DATA
  12. LIST POST/CLOSE COMPUTATION DATA
  13. VALIDATION OF CROSS FILE PROCESSES
  14. EDIT SYSTEM DEFINITION DATA
  15. SYSTEM GENERATION
  16. RESET CURRENT SYSTEM DATE
  17. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     0
```

System Responds:

```
     PEARL LEVEL 3  PROCESSING COMPLETED

     A>
```

User Enters:

```
     Take out Definition Disk from Drive A.

     Insert Application System Main Programs Disk in Drive A.

     SUBMIT B:COMPILE
```

System Responds:

> This process also takes some time. Each program generated will now be compiled. If you notice any errors, be sure to note the program and the line number at which the error occurred. Once compilation is completed, you may go then programs which are in error, find the line number and determine what is wrong. You may be able to correct the error in the generated source and simply recompile the program, or you may have to redefine, regenerate and recompile your application.

```
A>CBAS2 B:CONTACT A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:        150
CODE SIZE:          20000
DATA STMT AREA:         0
VARIABLE AREA:       3000

A>CBAS2 B:CF000 A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:         56
CODE SIZE:           4971
DATA STMT AREA:         0
VARIABLE AREA:       1152

A>CBAS2 B:CF000A A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:         40
CODE SIZE:           4474
DATA STMT AREA:         0
VARIABLE AREA:        896

A>CBAS2 B:CF100 A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:         80
CODE SIZE:          12233
DATA STMT AREA:         0
VARIABLE AREA:       1728

A>CBAS2 B:CFP0001 A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:         96
CODE SIZE:          13568
DATA STMT AREA:         0
VARIABLE AREA:       1800
```

```
A>CBAS2 B:CF000R A: $B
CBASIC COMPILER VER 2.06
NO ERRORS DETECTED
CONSTANT AREA:         96
CODE SIZE:          10959
DATA STMT AREA:         0
VARIABLE AREA        1760
A>
```

The compilation process is now complete.

GO TO STEP 10

Setting Up Disks

You are now ready to run your PEARL-generated system. To do this, you should put the Application System Main Program Disk you used during compilation in Drive A and PIP your .INT files onto the Application System Disk in Drive B. Also, you should have RUN.COM and other .COM files on your Application System Disk. (You no longer need your Control Data Disk). Then you will put your Application System Disk in Drive A, and you will use this disk as your system disk. Put your Application Data Disk in Drive B. Your data files will go on Drive B.

NOTE:

For larger systems with multiple files, you may want to use different procedures in order to keep SOURCE and the .INT files separate. See Chapter 19 for some suggested procedures.

You may now enter data for your application.

User Enters:

Insert Application System Main Programs Disk in Drive A.

Insert Application Data Disk in Drive B.

Boot up your computer.

Configuration

System Responds:

```
A>
```

User Enters:

```
RUN CONTACT
```

System Responds:

```
CRUN VER 2.06

SYSTEM CONFIGURATION DATA COULD NOT BE LOCATED
Enter an ESCAPE to create a new file on DRIVE A:, or,
ENTER DEFAULT SYSTEM DISK DRIVE A
```

User Enters:

Depress ESCAPE and RETURN

System Responds:

```
SYSTEM CONFIGURATION CONTROL FILE IS BEING INITIALIZED
CONFIGURATION CONTROL DATA
1:  TERMINAL TYPE                   USER DEFINED CLEAR SCREEN
2:  FORMS CONTROL OPTION            USE FORM FEED
3:  DISKETTE CAPACITY IN K BYTES             70
4:  MESSAGE LEVEL                   SUPPRESS ALL MESSAGES
5:  REPORT DEPTH IN LINES                     0
6:  INSTALLATION NAME
7:  DEFAULT SYSTEM DISK DRIVE       A
8:  DEFAULT DATE DISK DRIVE         B
9:  DATE FORMAT                     MM/DD/YY FORMAT
10: PASSWORD
ENTER TERMINAL TYPE
 0 =USER DEFINED CLEAR SCREEN
 1 =SOL
 2 =HAZELTINE
 3 =BEEHIVE
 4 =SOROC
 5 =INTERTEC
 6 =TRS 80 MODEL II
 7 =ADM-3(A)
( 0 )
```

User Enters:

```
     2
```

System Responds:

```
     ENTER FORMS CONTROL OPTIONS ( 0 )
     ENTER 0 TO USE FORM FEED, OR THE NUMBER OF LINES PER PAGE
```

User Enters:

```
     0
```

System Responds:

```
     ENTER DISKETTE CAPACITY IN K BYTES ( 70 )
```

User Enters:

```
     448
```

System Responds:

```
     ENTER MESSAGE LEVEL
     0 =SUPPRESS ALL MESSAGES
     1 =OPEN/CLOSE/CHAIN MESSAGES
     2 =DISPLAY I/O TRACES
     ( 0 )
```

User Enters:

```
     0
```

System Responds:

```
     ENTER REPORT DEPTH IN LINES ( 0 )
```

User Enters:

```
     55
```

System Responds:

```
     ENTER INSTALLATION NAME
```

User Enters:

```
     XYZ Company
```

System Responds:

```
      ENTER DEFAULT SYSTEM DISK DRIVE A
```

User Enters:

     Depress RETURN (or choose another drive A-D)

System Responds:

```
     ENTER DEFAULT DISK DRIVE FOR DATA FILES B
     (or choose another drive A-D)
```

User Enters:

     Depress RETURN

System Responds:

```
      ENTER DATE FORMAT
      0 =MM/DD/YY FORMAT (American)
      1 =DD/MM/YY FORMAT (European)
      (0)
```

User Enters:

```
     0
```

System Responds:

```
     ENTER PASSWORD
```

User Enters:

```
     #######
```

System Responds:

```
CONFIGURATION CONTROL DATA
1: TERMINAL TYPE                     HAZELTINE
2: FORMS CONTROL OPTION              USE FORM FEED
3: DISKETTE CAPACITY IN K BYTES              448
4: MESSAGE LEVEL                     SUPPRESS ALL MESSAGES
5: REPORT DEPTH IN LINES                      55
6: INSTALLATION NAME                 XYZ Company
7: DEFAULT SYSTEM DISK DRIVE         A
8: DEFAULT DATA DISK DRIVE           B
9: DATE FORMAT                       MM/DD/YY FORMAT
10: PASSWORD
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

Initialize Customer Contact File

NOTE

When you boot up the system once the configuration data has been entered, you will have to enter the PASSWORD before you can go on if one was defined.

System Responds:

```
                              CUSTOMER CONTACT FILE


                             GENERATED BY PEARL (TM)
                                       FOR
                                   XYZ Company
                                   VERSION 1.0
                                 September, 1980


ENTER CURRENT DATE:  091580
          CUSTOMER CONTACT FILE(A:CONTACT) VERSION 1.0
                  MAIN SELECTION MENU-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)
0. RETURN TO CP/M
1. INITIALIZE CUSTOMER CONTACT FILE
2. EDIT CUSTOMER CONTACT CONTROL DATA
3. UPDATE CUSTOMER CONTACT FILE
4. GENERATE CUSTOMER CONTACT REPORTS
5. COMPRESS CUSTOMER CONTACT FILE
6. CHANGE SYSTEM DATE
7. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

```
     1
```

System Responds:

```
LOADING PROGRAM TO INITIALIZE CUSTOMER CONTACT FILE

           CUSTOMER CONTACT FILE(A:CF000) VERSION 1.0
            CUSTOMER CONTACT INITIALIZATION-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

PROCESSING FILE CONTROL DATA
ENTER INDEX BUFFERS   (0)
```

User Enters:

```
     1
```

System Responds:

```
     FILE CONTROL DATA
     *: SORT FLAG                    0
     *: LAST RECORD                  1
     *: LAST RECORD DELETED          0
     *:`DELETED REC. COUNT           0
     *: UPDATE IN PROCESS FLAG       0
     *: OPEN ERROR RESET COUNTER     0
     1: INDEX BUFFERS                1
     *: CLOSING COUNT                0
     *: LAST JE CLOSED               0
     *: POSTING COUNT                0
     *: LAST JE POSTED               0
     ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

Depress RETURN

System Responds

```
     CUSTOMER CONTACT INTIALIZATION COMPLETED


     ....GOOD BYE....
```

System Responds:

```
          CUSTOMER CONTACT FILE(A:CONTACT) VERSION 1.0
                  MAIN SELECTION MENU-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)
0. RETURN TO CP/M
1. INITIALIZE CUSTOMER CONTACT FILE
2. EDIT CUSTOMER CONTACT CONTROL DATA
3. UPDATE CUSTOMER CONTACT FILE
4. GENERATE CUSTOMER CONTACT REPORTS
5. COMPRESS CUSTOMER CONTACT FILE
6. CHANGE SYSTEM DATE
7. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

Update Customer Contact File

User Enters:

3

System Responds:

```
LOADING PROGRAM TO UPDATE CUSTOMER CONTACT FILE
           CUSTOMER CONTACT FILE(A:CF100) VERSION 1.0
          CUSTOMER CONTACT UPDATE MAINTENANCE-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   TO ADD RECORDS TO THE FILE
2   TO EDIT OR DISPLAY EXISTING RECORDS
3   TO DELETE RECORDS
?
```

User Enters:

```
     1
```

System Responds:

```
     ENTER CUSTOMER NAME ( )
```

User Enters:

```
     DOE, JOHN
```

System Responds:

```
     ENTER CUSTOMER ADDRESS ( )
```

User Enters:

```
     1099 THIRD STREET
```

System Responds:

```
     LOCATION ( )
```

User Enters:

```
     SALEM, OR 97301
```

System Responds:

```
     PHONE ( )
```

User Enters:

```
     392-6299
```

System Responds:

```
     ENTER CONTACT DATE ( )
```

User Enters:

```
     040580
```

System Responds:

```
     ENTER BUSINESS GROUP
      0 =Government
      1 =Education
      2 =Contracting
      3 =Farming
      4 =Retail Sales
      5 =Wholesale Dist.
      6 =Other
     (6)
```

User Enters:

```
     1
```

System Responds

```
CUSTOMER CONTACT DATA
1: CUSTOMER NAME        DOE, JOHN
2: CUSTOMER ADDRESS     1099 THIRD STREET
3: LOCATION             SALEM, OR 97301
4: PHONE                392-6299
5: CONTACT DATE         04/05/80
6: BUSINESS GROUP       Education
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User Enters:

     Depress RETURN

System Responds:

```
     ENTER CUSTOMER NAME (DOE, JOHN)
```

User Enters:

Continue to enter the customer names, etc. You may enter them in any order and they will come out on your report in alphabetical order. This is true for INDEXED files only. This is why you enter the last name first and then first Name and Middle initial. However, if you had specified the RANDOM access method, you would have to SORT your file in order to get it to print out in alphabetical order.

NOTE

If you make a mistake during entry of the key field (CUSTOMER NAME) and you do not correct it when your answers are first displayed, you must select Option 3 (TO DELETE RECORDS) in order to correct the name. This is true only for the key field. You may choose Option 2 (TO EDIT OR DISPLAY EXISTING RECORDS) to correct errors in any of the other fields.

When you are done entering customer names, you may exit from this program as follows:

System Responds:

```
     ENTER CUSTOMER NAME (ADAMS, MARIE)
```

User Enters:

Depress ESCAPE and RETURN

System Responds:

```
           CUSTOMER CONTACT FILE(A:CF100) VERSION 1.0
          CUSTOMER CONTACT UPDATE MAINTENANCE-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   TO ADD RECORDS TO THE FILE
2   TO EDIT OR DISPLAY EXISTING RECORDS
3   TO DELETE RECORDS
?
```

User Enters:

```
     0
```

System Responds:

```
     ....GOOD BYE....
```

Generate Customer Contact Reports

<u>To Generate A Report by Primary Key</u>

System Responds:

```
LOADING PROGRAM TO GENERATE CUSTOMER CONTACT REPORTS
          CUSTOMER CONTACT FILE(A:CONTACT) VERSION 1.0
                  MAIN SELECTION MENU-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)
0. RETURN TO CP/M
1. INITIALIZE CUSTOMER CONTACT FILE
2. EDIT CUSTOMER CONTACT CONTROL DATA
3. UPDATE CUSTOMER CONTACT FILE
4. GENERATE CUSTOMER CONTACT REPORTS
5. COMPRESS CUSTOMER CONTACT FILE
6. CHANGE SYSTEM DATE
7. EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

    4

Make sure you have 8 1/2 by 11-inch paper in your printer and that your printer is on before you proceed.

System Responds:

```
          CUSTOMER CONTACT FILE(A:CFP0001) VERSION 1.0
           CUSTOMER CONTACT REPORT PROCESSING-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   GENERATE Customer Contact List REPORT
2   GENERATE List by Business Group (2K2) REPORT
?
```

User Enters:

    1

System Responds:

```
     ENTER REPORT.ID,
     (P=#,) FOR PAGE NUMBER
     (PAUSE) TO PAUSE AT EACH PAGE
```

User Enters:

Depress RETURN

You could enter a Report Id, page number or pause here if you wanted. For example: By Name,P=10,(PAUSE). This would mean you wanted your report to print with a subtitle of "By Name, to start printing on Page 10, and to Pause at each new page.

System Responds:

The following is an example of what the report would look like:

```
                                          XYZ Company                                        Page
                                      Customer Master List
                                            09/30/10

                                                                      Contact       Business
  stomer Name                                          Phone           Date          Group
-------------------------------------------------------------------------------------------------
ADAMS, MARIE                                           392-8793       04/07/80  OTHER
7899 EAST STATE
SALEM, OR    97303

DOE, JOHN                                              392-6299       04/05/80  EDUCATION
1099 THIRD STREET
SALEM, OR    97301

JOHNSON, RICHARD                                       988-2780       04/07/80  WHOLESALE DIST
6540 EAST PEARL STREET
SALEM, OR    97304

JONES, SAMUEL                                          399-8923       04/28/80  RETAIL SALES
3369 OVERTON DR.
SALEM, OR    97302

MILLER, DANIEL                                         992-2034       04/14/80  WHOLESALE DIST
2333 CONOVER DRIVE, NE
PORTLAND, OR    97222

SMITH, MARY                                            322-6390       04/14/80  EDUCATION
4329 WESTHAVEN RD.
SALEM, OR    97302

  HOMAS, MICHAEL                                       872-2300       04/21/80  FARMING
3250 97TH AVE., SE
ALBANY, OR    97322
```

To Generate a Report by Contact Date - Secondary Key

System Responds:

```
                CUSTOMER CONTACT FILE(A:CFP0001) VERSION 1.0
                 CUSTOMER CONTACT REPORT PROCESSING-09/15/80
                        MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   GENERATE Customer Contact List REPORT
2   GENERATE List by Business Group (2K2) REPORT
?
```

User Enters:

```
     1K*
```

System Responds:

```
          0 - PRIMARY KEY
          1 - CONTACT DATE
          2 - BUSINESS GROUP

ENTER REPORT SEQUENCE KEY
```

User Enters:

```
     1
```

*You could also enter 1K1 if you already knew the sequence.

NOTE

The report option differs if you have defined RANDOM access files. See Chapter 10, Section 10.4.2 and Appendix I for a more detailed study of Report Processing for both Indexed and Random Access files.

System Responds:

```
ENTER REPORT.ID
(P=#,) FOR PAGE NUMBER
(PAUSE) TO PAUSE AT EACH PAGE
```

User Enters:

```
     Contact Date
```

System Responds:

The following report will print out on your printer:

```
Contact Date                      XYZ Company                                Page
                              Customer Master List
                                    09/30/80

                                                             Contact      Business
Customer Name                                   Phone         Date         Group
----------------------------------------------------------------------------------
DOE, JOHN                                       392-6299     04/05/80   EDUCATION
1099 THIRD STREET
SALEM, OR   97301

ADAMS, MARIE                                    392-8793     04/07/80   OTHER
7899 EAST STATE
SALEM, OR   97303

JOHNSON, RICHARD                                988-2780     04/07/80   WHOLESALE DIS
6540 EAST PEARL STREET
SALEM, OR   97304

SMITH, MARY                                     322-6390     04/14/80   EDUCATION
4329 WESTHAVEN RD.
SALEM, OR   97302

MILLER, DANIEL                                  992-2034     04/14/80   WHOLESALE DIS
2333 CONOVER DRIVE, NE
PORTLAND, OR   97222

THOMAS, MICHAEL                                 872-2300     04/21/80   FARMING
3250 97TH AVE., SE
 LBANY, OR   97322

JONES, SAMUEL                                   399-8923     04/28/80   RETAIL SALES
3369 OVERTON DR.
SALEM, OR   97302
```

To Generate a Report by Business Group - Secondary Key

System Responds:

```
          CUSTOMER CONTACT FILE(A:CFP0001) VERSION 1.0
           CUSTOMER CONTACT REPORT PROCESSING-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   GENERATE Customer Contact List REPORT
2   GENERATE List by Business Group (2K2) REPORT
?
```

User Enters:

```
     1K*
```

System Responds:

```
          0 - PRIMARY KEY
          1 - CONTACT DATE
          2 - BUSINESS GROUP

ENTER REPORT SEQUENCE KEY
```

User Enters:

```
     2
```

*You could also enter 1K2 if you already knew the sequence.

System Responds:

```
ENTER REPORT.ID
(P=#,) FOR PAGE NUMBER
(PAUSE) TO PAUSE AT EACH PAGE
```

User Enters:

```
     By Business Group
```

System Responds:

The following report will print out on your printer:

```
By Business Group                    XYZ Company                                    Page
                                Customer Master List
                                      09/30/80

                                                              Contact       Business
Customer Name                                 Phone           Date           Group
---------------------------------------------------------------------------------------
SMITH, MARY                                   322-6390       04/14/80   EDUCATION
4329 WESTHAVEN RD.
SALEM, OR   97302

DOE, JOHN                                     392-6299       04/05/80   EDUCATION
1099 THIRD STREET
SALEM, OR   97301

THOMAS, MICHAEL                               872-2300       04/21/80   FARMING
3250 97TH AVE., SE
ALBANY, OR   97322

JONES, SAMUEL                                 399-8923       04/28/80   RETAIL SALES
3369 OVERTON DR.
SALEM, OR   97302

MILLER, DANIEL                                992-2034       04/14/80   WHOLESALE DIS
2333 CONOVER DRIVE, NE
PORTLAND, OR   97222

JOHNSON, RICHARD                              988-2780       04/07/80   WHOLESALE DIS
6540 EAST PEARL STREET
SALEM, OR   97304

ADAMS, MARIE                                  392-8793       04/07/80   OTHER
7899 EAST STATE
SALEM, OR   97303
```

To List by Business Group (2K2)

System Responds:

```
          CUSTOMER CONTACT FILE(A:CFP0001) VERSION 1.0
           CUSTOMER CONTACT REPORT PROCESSING-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   GENERATE Customer Contact List REPORT
2   GENERATE List by Business Group (2K2) REPORT
?
```

User Enters:

```
     2K2
```

System Responds:

```
ENTER REPORT.ID
(P=#,) FOR PAGE NUMBER
(PAUSE) TO PAUSE AT EACH PAGE
```

User Enters:

Depress RETURN

System Responds:

```
     ENTER THE STARTING DATE FOR MONTH-TO-DATE TOTALS
```

User Enters:

```
     41780
```

System Responds:

The following report will print out on the printer:

```
                              XYZ Company                           Page    1
                    Contact List by Business Group
                               09/30/80

Business Group          Date Range                     Contact Totals
------------------------------------------------------------------------------

 1   EDUCATION            04/17/80      Month-to-Date:     0
                                         Year-to-Date:     2


 3   FARMING              04/17/80      Month-to-Date:     1
                                         Year-to-Date:     1


 4   RETAIL SALES         04/17/80      Month-to-Date:     1
                                         Year-to-Date:     1


 5   WHOLESALE DIST.      04/17/80      Month-to-Date:     0
                                         Year-to-Date:     2


 6   OTHER                04/17/80      Month-to-Date:     0
                                         Year-to-Date:     1

TOTAL MONTH-TO-DATE:     2

 TOTAL YEAR-TO-DATE:     7
```

System Responds:

```
          CUSTOMER CONTACT FILE(A:CFP0001) VERSION 1.0
           CUSTOMER CONTACT REPORT PROCESSING-09/15/80
                   MINIMUM FREE SPACE=(NNNNN)

0   TO RETURN TO MAIN MENU
1   GENERATE Customer Contact List REPORT
2   GENERATE List by Business Group (2K2) REPORT
?
```

User Enters:

```
     0
```

System Responds:

```
     ...RETURNING TO MAIN MENU...
```

YOU HAVE COMPLETED THE EXAMPLE APPLICATION.

After testing your system, you would PIP over your .INT files (and your .COM files if not done) from your Application System Main Program disk to your Application System disk for production.

For procedures on file reorganization, Option 5 of Main Menu, see Appendix L of this manual.

CHAPTER 5. PEARL MAIN SELECTION MENU

When PEARL Level 3 is executed, the selection menu for all PEARL processing options will be displayed. The processing associated with each option is described in detail in the remainder of the manual. The following is a quick cross reference to a discussion of each menu option.

0. RETURN TO CP/M	
1. SYSTEM INITIALIZATION	(Chapter 6)
2. FILE DEFINITION	(Chapter 7)
3. DATA ELEMENT DEFINITION	(Chapter 7)
4. PHRASE SELECTION DEFINITION	(Chapter 7)
5. MAIN MENU DEFINITION	(Chapter 8)
6. REPORT CONTROL DEFINITION	(Chapter 9)
7. REPORT DETAIL DEFINITION	(Chapter 10)
8. ENTER/EDIT POST/CLOSE COMPUTATIONS	(Chapter 11)
9. LIST DATA ELEMENT CONTROL DATA	(Chapter 13)
10. LIST MENU CONTROL DATA	(Chapter 14)
11. LIST REPORT CONTROL DATA	(Chapter 15)
12. LIST POST/CLOSE COMPUTATION CONTROL DATA	(Chapter 16)
13. VALIDATION OF CROSS FILE PROCESSES	(Chapter 11)
14. EDIT SYSTEM DEFINITION DATA	(Chapter 6)
15. SYSTEM GENERATION	(Chapter 17)
16. RESET CURRENT SYSTEM DATE	
17. EDIT SYSTEM CONFIGURATION DATA	(Appendix B)

CHAPTER 6. PEARL SYSTEM INITIALIZATION

The A000.INT program is used to initialize system definition control files (PEARL.CCT and PEARL.NDX). These files are placed on the default data drive as defined in the SYSTEM CONFIGURATION CONTROL data. This program first tests for the presence of PEARL.CCT and PEARL.NDX on the default data drive disk. If these files already exist, an error message is issued and control is returned to the Main Menu. PEARL.CCT will contain all of the definition data entered by you when defining a system, including system description data, and file and data element definitions. PEARL.NDX is an index into PEARL.CCT. In addition to creating PEARL.CCT, this step also prompts for the system description data and writes it to PEARL.CCT. These prompts are:

6.1 SYSTEM DESCRIPTION:

The string that is entered here will appear on all menus in the generated system. It will also appear as part of the title on every page of the data element detail and summary listings produced under Main Menu Selection 6 below.

6.2 MAIN MENU PROGRAM NAME:

The 1-8 character CP/M primary file name you enter here will be the name of the generated source file containing the main menu program and hence, after compilation, the name of the .INT file used to begin execution of the generated system. You should only enter the file name. A file type of .BAS will be appended automatically.

6.3 SYSTEM CODE:

This 1-3 character code becomes part of the name of the generated COMsss.BAS (sss=system code) file. This file will contain variable initialization statements for items relevant to the whole system, such as SYSTEM DESCRIPTION (as entered above), SYSTEM DEVELOPMENT DATE and SYSTEM VERSION NUMBER.

6.4 SYSTEM DEVELOPMENT DATE:

The string entered here will appear as a string constant in COMsss.BAS as the value of VER.DATE$, and will be displayed when the generated main menu program is initially loaded.

6.5 SYSTEM VERSION NUMBER:

The string entered here, prefixed by "VERSION N.N", will appear as a string constant in COMsss.BAS as the value of VER.NO$, will be displayed when the generated main menu program is initially loaded. It will appear together with SYSTEM DESCRIPTION at the top of all menus in the generated system.

PROGRAMMING NOTE:

The date and the version number are placed in an included file with the name COMuuu (where uuu is the SYSTEM CODE entered above) with the variable names VER.NO$ and VER.DATE$. Because these variables are not included in the COMMON area, the version number and date displayed in each program reflect the variable values the last time each program was compiled rather than at the time the main menu program was last compiled.

6.6 BUFFER ALLOCATION:

This determines the number of disk sectors on a given file to be maintained in memory at one time. Example:

```
CREATE filename AS filenum BUFF nn RECS 128
```

where nn is the buffer allocation you specify. Larger values for nn result in faster generation, but require more memory space. On a system with 48K of memory there is only room for nn = 1; with 50K or more, use 10.

This value is also the number of index buffers which will be allocated for processing the PEARL.NDX file. Refer to Appendix D for a description of the IS.NBUFS% variable.

NOTE:

Tuning of this value may be done by watching the MINIMUM FREE SPACE=(xxxxx) displayed at the top of each PEARL display screen. If the value indicated by xxxxx is less than 1000, you should decrease the number of buffers to minimize the chance of running out of working memory during the execution of PEARL. (If this occurs, an ERROR OM will result during execution and may cause the files to become unusable.)

6.7 DRIVE FOR GENERATED SOURCE:

This is a single character indicating the drive on which the BASIC source files generated by PEARL should be placed. This drive is independent of the DEFAULT DATA DISK DRIVE on which the PEARL control files are placed. Both specifications, however, may be the same drive. In this case, the PEARL control files and the generated source code will be placed on the same unit.

6.8 DRIVE FOR COMMON ROUTINES:

This is a single character indicating the drive to contain the common routines (COMPILE DISKETTE) during the compile processing.

6.9 DRIVE FOR GENERATED ROUTINES:

This is a single character indicating the drive to contain the generated subroutines during the compile process. This will probably be the same drive specified above in "DRIVE FOR GENERATED SOURCE", but does not necessarily have to be the same.

NOTES:

1. Once the PEARL.CCT and PEARL.NDX files have been created on a PEARL system data disk using the SYSTEM INITIALIZATION menu selection, another system cannot be initialized on the same disk. To re-use a system data disk, these two files must first be erased.

2. Once a system has been defined, selection 1 cannot be used to change any of the definitions listed above. In order to edit this information, select main menu option 14. You are presented with a menu allowing you to edit the definitions, or list them out on your printer.

CHAPTER 7. FILE DEFINITION

Before any data elements can be defined for a file, the file must first be defined. The file definition information, like all the system definitions, is maintained by PEARL in the PEARL.CCT file. When you enter this step, you will be presented with a sub-menu providing the options of adding a file definition or editing or listing an existing definition. Here we will describe the prompts issued by PEARL for adding a new file definition, what the possible answers are, and the effects of those answers on the generated program code.

A word on <u>control records</u> is in order here. Up to nine (9) control records may be defined for a file. These control records occupy the first 1 to 9 record positions in the data file. The rest of the file contains only data records. The first data record immediately follows the last control record. For example: If there were 3 control records, the first data record would be record 4. PEARL automatically defines several control data elements in control record 1 which are used by the generated I/O subroutines for file maintenance; the user may add more for his own use. (Refer to Appendix H.)

The replies you enter in response to the prompts control data and generated code file names, variable names, and subroutine labels. Some replies go into assignment statements that initialize file accessing variables.

The following information is required when a file is defined:

7.1 RELATIVE FILE ID:

When the file definition processing menu is loaded, you may define or edit the file control data for any number of files prior to returning to the main menu. In order to alter the current file number, enter F=n where n is the file number to be processed. The number may be any number between 1 and 99. The current file being processed will be indicated on the bottom line of the screen header in each PEARL program currently being used.

Once a file number has been selected through the submenu, subsequent file processing will process the file which has been selected.

7.2 FILE DESCRIPTION:

A phrase description of the data file, such as CUSTOMER MASTER LIST, GENERAL JOURNAL, or RECIPE FILE. This phrase appears as part of the title on the file and data element definition listings (produced by PEARL Main Menu selection 6, Report Processing), and as part of menu options on the generated program. If you entered RECIPE FILE here, for example, the generated main menu would contain options to UPDATE RECIPE FILE and PRINT REPORTS FOR RECIPE FILE.

7.3 ACCESS METHOD:

Two file access methods are provided by PEARL: RANDOM and INDEXED. (The access method to be used will be determined by the **FILE TYPE** selection as described below in Section 7.4.) In either case, the file is a CBASIC random-access file with fixed-length records. Each record is accessed by its CBASIC record number, starting at 1 for the first record on the file. The control record(s) are at the beginning of the file, with the data records immediately following. The control records are accessed directly by number (1-9). The data records are accessed by a key (explained below), and must be the first data element defined in the data record. The two different methods are explained below.

RANDOM access

The data records in the file will be accessed by relative record number. The relative record number to assign to the first data record is asked for at the time the file is created and initialized by the generated file initialization program. It is stored as one of the PEARL-defined data elements in control record 1. Each record added to the file is written at the end of the file, and given a relative record number one higher than the previous record. Since assignment of a relative record number is automatic, the generated file update program does not ask the operator for the relative record number when adding records to the file. However, it appears on the display which is shown to the operator for his approval before adding. To edit a data record, you must supply its relative record number. This number can usually be obtained from a printout of the records on the file. To read or write a data record using the generated I/O subroutines, you supply the relative record number.

The relative record number is also stored into the first data element of each data record, so you must define the first data element as an integer or floating-point number. Floating-point is advised because any number may be defined as the beginning relative record number later on when the file is initialized, and might exceed the maximum size for an integer (32767).

This access method has the advantage that the generated I/O routines are simpler than for indexed files. The INDEX routines are not needed, saving several K of memory space.

For example, RANDOM ACCESS would be ideal for an application where a stack of tickets or invoices is to be entered into a transaction file. Since the generated report program will list out the records in the same order as the tickets were entered, proofreading would be easier. The corrections would be made right on the listing, and thus, the relative record numbers would be immediately available. Also, if the tickets were sequentially numbered, you could have the relative record numbers equal the ticket numbers by setting the first relative record number equal to the first ticket number when you create the data file using main menu selection 1 of the generated system.

The disadvantages of RANDOM access are that the relative record number must be known in order to edit a record, and that deleted records cannot be re-used (without copying all the records over onto a new file).

INDEXED access

Access to the data records will be by a character string key. The key for each record will be the value of the first data element in the record. It may have a data element type of number, date, or string. The only restriction is that the key for each record must be unique. A second file with a file name of NDX will be maintained, containing all the keys with the numbers of the associated data records in a tree-structured index. When records are added to the data file, they are written at the end, just as with the RANDOM access method, and the CBASIC record number is inserted into the index along with the record key. To retrieve a record for display or editing, the key is looked up in the index to find out the CBASIC record number of the record on the data file. Because of the tree structure of the index, any key can be found very quickly, without the need to read the whole index file.

This method has the disadvantages of requiring 1) an additional file on the disk to contain the index; and 2) the inclusion of the INDEX subroutines into the processing programs, which takes up memory space.

The advantages are: 1) the ability to retrieve a customer master record, for example, by customer name directly. You do not have to consult a printout to find the record number of the desired record; 2) the generated report program will list the records in order by key rather than the order they were originally entered in; and 3) deleted data records are re-used immediately when new records are added.

Indexed files also support up to 9 secondary keys that are not unique. These keys are stored on the index file in the same manner as are primary keys. The advantage of secondary keys is the ability to access data file records in order by a variable or record other then the primary key. Secondary keys are used by report writer routines only. The disadvantage of secondary keys is that they require 1K or more memory for the processing routine and increase the size of the index file on disk.

7.4 FILE TYPE:

The following options are available:

MASTER FILE

This file is always an indexed file, and may be associated with a TRANSACTION FILE and HISTORICAL TRANSACTION file.

MISC. INDEXED FILE

An indexed file which does not have transaction files associated with it. (Logical connections may occur between this file and other files in the system, but PEARL will not generate the code for such connections.)

TRANSACTION FILE

A transaction file is a random access file used to capture data which will in turn be posted to update a MASTER FILE. After this file is defined, the relationships between the transaction file and the master file must also be defined by the user.

One of the attributes of the TRANSACTION FILE is that the data placed on this file may also be transferred to a Cumulative Historical Transaction File.

A master file must be defined.

HISTORICAL TRANSACTION FILE

If a file is defined as an HISTORICAL TRANSACTION FILE, then you may select to have the detail transaction data which is captured on the Transaction File moved to this file during the period-end closing process.

A transaction file must be defined.

MISC. RANDOM FILE

When this option is chosen, the programs to create, update, and generate reports will be created by PEARL. No logical connections exist between this file and other files generated by PEARL.

7.5 DATA FILE NAME:

A valid CP/M unambiguous primary file name. A secondary file name of DAT is automatically supplied by PEARL. This is the name of the file being defined.

7.6 UNIQUE FILE IDENTIFIER:

A one- or two-alpha-character string. This identification code will be used to prefix all generated source files to process the file being defined.

PROGRAMMING NOTE:

It is suggested that during the definition of the data elements for each file, this same code be used to prefix all variable names. (This is done automatically for data elements defined by PEARL in the file header control records for each file.) This is a programming standard which will make the programs easier to modify and will give clarity for programmers who maintain the system in the future.

7.7 LOGICAL FILE UNIT:

The CBASIC runtime file identification number to be used for accesses to this file. Must be in the range 1-20.

PROGRAMMING NOTE:

A value of 1 should be reserved for an indexed file. Also note that the numbers specified should be unique for the MASTER FILE, TRANSACTION FILE, and HISTORICAL TRANSACTION FILE. If these numbers are not unique, a run-time error will occur when an attempt is made by the program to open two files with the same file id number.

7.8 DISK DRIVE UNIT:

A single alpha character is used here to identify the disk unit on which the file will be placed. If a value of Z is specified, the generated programs will replace the drive unit specification with the value of the DEFAULT DATA DISK DRIVE prior to the time that the file is processed.

7.9 I/O SUBROUTINE BASE:

The first label of the generated I/O module for the file. Defaults to 10000. This module has entry points for opening and closing a file and adding, replacing or deleting records. The statement labels in the module will range from base to base+999.

PROGRAMMING NOTE:

You should insure that the I/O subroutine base is unique for the MASTER FILE, TRANSACTION FILE, and the HISTORICAL TRANSACTION FILE within the system. For example, values of 10000, 11000, and 12000 could be used for each file, respectively. If these values are not unique, duplicate label errors will occur during the compile of those programs which process data in more than one of these files.

7.10 DATA RECORD EDIT BASE:

The first label of the generated display/edit subroutine module for the file. Defaults to 30000. This module has entry points for displaying a data record on the screen, and inputting new data during editing. The statement labels in this module will range from base to base+199.

7.11 CONTROL RECORD EDIT BASE:

The first label of the generated control record display/edit module. Defaults to 40000. This module has entry points for displaying and editing the contents of control records. If the file has only one control record, the labels in this module will range from base to base+199. If there is a second control record, its routines will range from base+1000 to base+1199, a third from base+2000 to base+2199, etc.

Display-only data fields. In addition to the above fields for which you are prompted, the file definition record contains the following fields which will be displayed but cannot be edited:

KEY FIELD DESCRIPTION. The prompt for inputting the key field of each data record (which is always the first data element in the record).

VARIABLE NAME FOR KEY. The CBASIC variable name of the key field.

KEY FIELD LENGTH. Length in characters of the key field.

RECORD LENGTH. The total number of characters each data record occupies. It will be the sum of all the data element field lengths plus 6; this total cannot exceed 255.

CONTROL RECORD COUNT. The number of control records which have been defined.

TOTAL DATA ELEMENT COUNT. The total number of data elements defined for this file, including those in the data record and those in all of the control records.

NOTES

1. Often more than one file must be processed by a program. Interference among the generated subroutines for multiple files is prevented by the unique file identifier, which prefixes all generated code file and variable names, and by entering different file names, logical file units, and subroutine bases for each file. While multiple files can be

defined on a single PEARL control file, however, only one MASTER FILE, one TRANSACTION FILE, and one HISTORICAL TRANSACTION FILE may be defined in a single system.

2. At the time a file definition record is written to PEARL.CCT, several data elements for the file are defined automatically by PEARL. These data elements will be placed in the first control record of the file being defined and are always required in order for the generated I/O routines to function properly. (Thus, there will always be at least one control record on the file even if the user does not define any control record data elements himself.) Edit line sequence numbers 1001 to 1008 (see Chapter 8 on defining data elements) are reserved for these control data elements. These elements will be displayed on the data element definition listing produced by main menu selection 11, List Report Control Data.

   Refer to Appendix H for details on the file control variables generated for each file type defined.

CHAPTER 8. DATA ELEMENT DEFINITION AND PHRASE SELECTION

8.1 DATA ELEMENT DEFINITION.

Once a file has been defined, the data elements to be stored on the file can be defined. These data elements are of two types: 1) the elements which make up a data record on the file, which we will call simply data elements, and 2) the data elements which make up the control record(s) at the beginning of the file, which we will call control data elements. Note that any given control data element occurs only once on a file since the control record of which it is a part occurs only once. It is typically a datum applying to the entire file, such as number of records on the file, a total of the overtime hours field of all the data records, a flag indicating whether the file has been posted to some master file, the date the file was last edited, etc. On the other hand, the data elements making up the data records are repeated as many times as there are data records. For example, in an employee master file, there will be as many employee names on the file as there are data records.

Although the power of PEARL is most obvious in defining data records, the programmer is encouraged to try defining control records, too, as they are generally quite useful in developing an application.

Each data element has a number of attributes defined for it which affect the generation of the I/O, display and editing routines for the file. These are prompted for as described below.

EDIT LINE SEQUENCE:

Data and control data elements are entered with an edit line sequence number, which works like the line numbers used by most BASIC interpreters. Regardless of the order in which you enter the data element definitions, they will be ordered within the record, on edit displays and on definition listings by numerical order of edit line sequence numbers. On initial entry of definitions, a default sequence number is provided equal to the last one entered plus 10. This is convenient most of the time since you will usually enter the elements in ascending order anyway. Should you wish to insert one between two existing ones, simply give it a sequence number in between the other two. A maximum of four digits is allowed.

The sequence number also determines whether an element is part of a data record or a control record. Numbers in the range of 1-999 are for elements comprising the data record. Numbers in the range 1001-1999 are for elements to be placed in control record 1; 2001-2999 for control record 2; and so on up to 9001-9999 for control record 9. The number of control records there will be is determined by the highest sequence number entered.

The detail report printed by Main Menu selection 6. (Report Control Definition) will have a separate listing beginning on a new page for the data record and for each control record. If you enter a data element definition with a sequence number of 1000, no actual element will be defined, but the VARIABLE DESCRIPTION field of the definition will be used as a subtitle at the top of the first page of definition listing for control record 1. Sequence numbers 2000, 3000, ..., 9000 work similarly for control records 2 through 9.

Care should be taken not to define any control data elements in control record 2 before defining at least one in control record 1, nor in 3 before having at least one in control records 1 and 2, etc. Doing so will cause a program malfunction. However, you may go back to earlier records and define more elements in any order as long as you follow the above rule.

Sequence numbers 1001 to 1009 are reserved for control data elements defined automatically by PEARL.

The data element with the lowest sequence number will be the <u>key</u> for the data record. For RANDOM access files, it must be a numeric (preferably floating-point) type. When adding records to the file, the generated update program will not ask for the key to be input, but will use a number one higher than the key for the previous record. For INDEXED access files, the key may be a data element of any type. When adding records to the file, the generated update program will use the value entered by the operator for the first data element as the key to insert into the index for that record.

VARIABLE DESCRIPTION:

The phrase entered here will be used as a prompt to the operator by the generated programs when inputting a new value for the data element, and will be used to describe the data on the display/edit screen. Control data element values are asked for by the generated file initialization program when it first creates the file; these values may be edited by the generated file control information edit program. Data record element values are asked for by the generated file update program.

VARIABLE CODE NAME:

This will be the actual CBASIC variable name appearing in the generated file I/O routines which is used to contain the value of the data element. PEARL restricts the length of this name to 24 characters; the standard CBASIC variable naming conventions must be followed. The last character of the name should be % for CBASIC integer variables, and $ for string variables. If you wish to store the data element's value in an element of a dimensioned array, rather than in an unsubscripted variable, simply append the subscript (enclosed in parentheses) to the variable name. For example, you may wish to store a salesman's total sales for each quarter in four data elements of the data record, and keep all four totals in an array rather than four differently-named variables for programming convenience. You could simply define the first one as QUARTERLY.SALES(1), the second as QUARTERLY.SALES(2), etc.

NUMBER OF OCCURRENCES:

If you have chosen a subscripted variable name, PEARL can automatically generate the CBASIC DIMENSION statement for you. This prompt is issued only when the VARIABLE CODE NAME ends with a right parenthesis. In response to this prompt, enter the number to put into the DIMENSION statement. If you enter 0, no DIMENSION statement will be generated this time. Enter a non-zero value only once for a given subscripted variable, and zero for succeeding occurrences; otherwise there will be several DIMENSION statements generated for the variable.

STORAGE FORMAT:

This determines how the variable will be stored on the file. There are five choices:

FLOATING POINT. A CBASIC real number. Up to 12 significant decimal digits will be kept.

INTEGER. A CBASIC integer value. Must range from -32768 to +32767.

STRING. A string of up to 60 characters.

DATE. Kept as a 6-digit real number in the form yymmdd. The generated file update routines will accept only valid dates in the form mm/dd/yy or mmddyy or mm dd yy.

MONEY. Kept as a real number, but is rounded off to two digits after the decimal point before writing to the file.

COMPUTATIONAL An expression is used in place of a data element name. This expression will be evaluated and displayed with the description on the display/edit screen. If this option is used, no space will be reserved for the data element on the file, and the item may not be edited by the operator. The variable code name may be an expression and must be suffixed with the character "=".

For example:

If elsewhere in the record the variables IM.PRICE and IM.UNITS are defined, and you wish a display line of the extension of these two values to be provided to the operator, the COMPUTATIONAL mode may be used, and the variable name may be specified as **IM.PRICE*IM.UNITS=**. At the time the code is generated for the program, the "=" will be removed from the expression.

LENGTH OF FIELD:

The number of characters to allocate for the data element on the record. The generated file update routines will not allow input of strings longer than this length. When writing to the file using the generated I/O routines, strings longer than this length will be truncated; floating point, integer and money values whose string representations (including decimal point and a minus sign if necessary) is longer than this length will be set to zero, and a message will be issued to the console. Dates are automatically allocated a length of 6.

LOW RANGE FOR VARIABLE:

For floating point, integer and money quantities, a range validation can be provided. The generated data input and editing programs will require input values to be within a range. If no low range (i.e., lower limit) is desired, hit the spacebar and RETURN.

HIGH RANGE FOR VARIABLE:

Upper limit, if desired. See above.

EDIT MASK:

This will be the format string for the data element used for display on the edit display screen and in the reports generated by the system. This field is optional. If it is used, it must be a valid CBASIC format string; otherwise, runtime errors may occur in the generated programs. Dates are converted to an 8-character string with imbedded slashes (mm/dd/yy) before printing, so a string formatting specification should not be used for dates. If no edit mask is entered, an appropriate default will be supplied.

EDIT CONTROL OPTION:

You may select:

NO EDITING RESTRICTIONS. Self-explanatory.

EDIT DURING ADD ONLY. When data element values for a record are initially input from the console, the operator is allowed to edit the entered values before the record is written to the file. The edit during add only option allows the edit at this time, but once the record is written to the file this data element cannot be changed.

NOTE:

Indexed file primary keys must be editable. Transaction and historical transaction file cross file validation keysmust also be editable.

DISPLAY ONLY. This data element will not be asked for during input, nor can it be edited. On edit displays, this element will have an asterisk by its line number.

SUPPRESS DISPLAY/EDIT. For data elements which are not to be input from, nor displayed, to the user. User-supplied code will read and write these values.

VALIDATION CONTROL OPTION:

Specifies restrictions on the input allowed.

NO SELECTION. No restrictions other than field length and any low and/or high range checks which may have been entered above.

Y/N SELECTION. For string data elements. The generated file update routines will accept only a "Y" or "N" from the console.

ALPHA SINGLE CHARACTER. For string data elements. Only a single alpha character will be accepted from the console.

PHRASE SELECTION. For integer data elements. Using the phrase menu selection (see Section 8.3 below) you will enter a list of selection strings. The strings are numbered starting at 0, and will be displayed on the screen when editing or displaying the element. Whenever the value of the item is displayed on the screen or by the generated report program, the string corresponding to the current value of the integer data element will be displayed. This allows the data element to occupy only one or two positions in the record, while still interacting with the user in meaningful phrases.

CROSS FILE VALIDATION. This option may be used on transaction and historical transaction files. The first variable with a cross validation option will be used to access a master file record via the master file primary key.

Cross file validation keys must be of the same variable type and length as the master file primary key and all such keys must be editable.

NOTE:

This option may be used in conjunction with the COMPUTATIONAL option described above to display on the update screen data obtained from the master file record. For example, if data elements are defined as follows:

IM.ITEM$	MASTER FILE PRIMARY KEY
IM.DES$	DESCRIPTION OF THE ITEM ON THE MASTER FILE
IT.IM.ITEM$	CROSS VALIDATION DATA ITEM DEFINED IN THE TRANSACTION FILE

then the description of the item may be displayed on the transaction file display/edit screen by entering a COMPUTATIONAL item in the list as IM.DES$= for the variable name.

KEY OPTION

The KEY OPTION is available only for indexed files. The available options are as follows:

0 NOT APPLICABLE

Used for all data items not defined as keys.

1 PRIMARY KEY

Used for the first item in an Indexed File Record. This variable will always be used to access records during update processing. Each record on the indexed file must have a unique primary key.

2 SECONDARY KEY

Any variable other than the primary key on an indexed file may also be defined as a secondary key. When this option is used, an index entry will be placed in the index for all occurrences of the key on the file. Duplicate occurrences of secondary keys are allowed on the file.

If secondary keys are defined on a file, the PEARL-generated reports will allow the end user to generate reports using any one of the key sequences. Control breaks and summary subtotals may be defined using all or a portion of the secondary key.

DESIGN NOTE:

The use of secondary keys on a file may expand the size of the index file considerably. In order to estimate the amount of index file space that will be required, the following formula may be used:

ESTIMATED SPACE = (111 + 6) * 1.37

Where 111 is the average length of the key field (not including trailing blanks).

DISPLAY-ONLY FIELDS

The data element definition record contains the following fields which will be displayed but cannot be edited:

NUMBER OF PHRASE SELECTION ENTRIES. This number will be incremented by the phrase selection table processor at the time phrase selection entries are added to the PEARL.CCT definition file using Main Menu selection 4.

POSITION IN RECORD. This value is computed just before report processing, by going through all the defined variables in sequence number order. For data elements in the data records, it will indicate the beginning position of the data element in each record (see Appendix H for record formats). For control records, this will always be zero as control records are written in variable rather than fixed format.

8.2 CONSIDERATIONS WHEN DEFINING DATA ELEMENTS

MAXIMUM RECORD LENGTH.

The total length of a record is limited to 253 characters, so that the entire record can be read into a string variable. The sum of the field lengths of all the data elements in a data record must be less than or equal to 249 (six characters are used internally).

FILE HEADER RECORD (CONTROL RECORD) FORMAT.

Control data elements are stored in variable format, separated by commas. The total length of a control record must not exceed the length of the data record. The detail report produced by Main Menu selection 6. below, Report Processing, provides an estimated maximum length for each of the control records. If the estimated length exceeds the length of the data records, either data elements in the offending control record should be eliminated or moved to other control records, or data record data element lengths should be increased or more data elements added, to make the file record length big enough to accommodate the control records. If a control record comes out longer than the record length declared when the file was opened, a runtime error will occur.

SORTING DATA FILES.

Because the data elements in the data records are placed in fixed positions, the data file may be sorted using programs like Structured Systems Group's QSORT program. The first sort key should be on character position 2 of the record in order to keep the control records at the beginning of the file, and make deleted data records come out at the end. The data element definition listing conveniently supplies the position and length of each field for specifying key fields for sorting. The sort utility routine (XSORTX) automatically takes care of the above mentioned first sort key position (See Appendix I).

NUMBER OF DATA ELEMENTS PER RECORD.

While the number of data elements in a record is limited only by the resulting total record length, you will normally want to limit it to 22 elements, which is the number of lines available on a 24-line CRT screen after the header and trailer lines are displayed, so that the entire edit display fits on the screen.

8.3 DATA ELEMENT PHRASE SELECTION ENTRY AND EDITING

Option 4 on the Main Menu will load the program to add and edit phase selection information. Upon entry to this program the operator may select the file for which phrase selection entry and editing is to be performed by using the F=nn option on the menu.

The data element for which phrases are to be entered must have been defined in the file which is being processed, and must have PHRASE SELECTION specified as the VALIDATION OPTION.

The generated file update routines will prompt for the value of the data item by displaying the selection phrases with their corresponding integer value; the user enters the integer. On display/edit screens displayed by the generated file update routines, the phrase corresponding to the current integer value of the data element will be displayed rather than the integer value itself.

You will be prompted for the edit line sequence of the data element for which selection phrases are to be entered or edited. Enter the number, or RETURN to exit back to the Main Menu. If the data element is not defined to have array validation, you will get an error message and be prompted to enter another line sequence number.

When editing previously defined phrases for an element, the phrases are displayed and you are prompted to ENTER LINE NUMBER TO EDIT. If you enter a number larger than the highest one shown, you will go into add mode starting with an "nn" value one higher than the last. To delete a phrase, select its line number to edit, then enter an asterisk as the new phrase and depress return. When this is done, all of the entries below the deleted entry will be resequenced to fill in the gap.

When the file update routines are generated, code is inserted into the generated prINTL.BAS file for the validation phrases for each data element.

Validation phrases are set up in a subscripted array on the generated prINTL file which is included in all generated main programs.

Example:

A variable (v.name%) was defined as a data element with a description of (THE PHRASE) and a validation option for phrase selection. The following phrases were defined for v.name%:

```
0 = first phrase
1 = second phrase
2 = third phrase
```

The generated code to process the defined phrases will be as follows:

Source File	Code
prINTL.BAS	V.NAME.COUNT = 2 (number of phrases defined -1) DIM V.NAME.VAL$(V.NAME.Count) V.NAME.VAL$(0)="first phrase" V.NAME.VAL$(1)="second phrase" V.NAME.VAL$(2)="third phrase"
pr30100.BAS (a file update routine)	PRINT "Enter the Phrase " FOR J%=0 TO V.NAME.COUNT PRINT J%;"_";V.NAME.VAL$(J%) NEXT J%

CHAPTER 9. MAIN MENU DEFINITION

The menu defintion program is invoked by selecting option 5 on the main PEARL control menu. The following menu is then displayed:

```
0    TO RETURN TO MAIN MENU
1    TO ADD NEW MENU ENTRIES
2    TO EDIT EXISTING MENU ENTRIES
3    TO DELETE A MENU ENTRY
4    TO CREATE DEFAULT MENU CONTROL
```

9.1 REQUIRED CONTROL DATA

All systems generated by PEARL Level 3 support a main menu program that has the following functions:

9.1.1 The first and last program to be executed for any system.

9.1.2 The display of a menu containing all the major functions of the system.

9.1.3 Chain to any defined program specified on the menu.

The following is a description of how menus are defined:

MENU SEQUENCE KEY

Generated menus may consist of a simple menu, or may consist of one primary menu and a seris of submenus Submenus are useful when a given menu becomes too large and cluttered. In such cases, a submenu may be used to block a group of related menu options into a separate menu which is displayed when called from the main or primary menu.

The sequence key is used to define which menu, and the sequence within each menu that an entry will appear. The following ranges may be used:

0001-0999	Entries in Main Selection Menu
1000	1st Sub Menu Description
1001-1999	Entries in 1st Sub Menu
2000	2nd Sub Menu Description
2001-2999	Entries in 2nd Sub Menu
.	
.	
9000	9th Sub Menu Description
9001-9999	Entries in 9th Sub Menu

PROGRAM DESCRIPTION

You may enter here the description of the program to be chained to, the description of a submenu, or any other phrase which corresponds to the function to be invoked as described below.

PROGRAM

The name of the program to be chained to from the menu program, or one of the following reserved words to invoke a special function:

SUBx	Where x is a value 1 through 9 and identifies one of the submenus. When this entry is selected, the submenu will be displayed.
LIST	When this option is used, it must be the first entry in a submenu. When selected by the operator, it will cause a list of programs to be executed consecutively, and the programs placed in the list will be the programs appearing in the remainder of the submenu. This will be accomplished in the main menu program by placing the menu entries in the common variable CC.CHAIN.LIST$, and then chaining to the first entry in the list. When execution of each program is completed, a chain to the next program in the list will occur.
DATE	When this entry is selected, the operator will be requested to enter a new system date. The date will be placed in the CC.DATE and CC.DATE$ variables in common, and will then be available to all programs in system.
CON	When this entry is selected, the system configuration data may be edited by the operator.

DISK IDENTIFICATION

You should enter a value between 0 and 5. This value will be used by the menu program to prompt the operator to mount a diskette when a program is not loaded. The number which is used here will depend upon the hardware configuration of the end user of a generated system.

When a program is not located, the number which is specified will be used as a subscript to the variable SYSTEM.DISK$(n) which is defined in the menu program as follows:

1	MAIN SYSTEM DISKETTE
2	FILE MAINTENANCE DISKETTE
3	PERIOD END CLOSING DISKETTE
4	REPORT PROCESSING DISKETTE
5	SORT WORK DISKETTE

For example, if the program IM000 is not located on the disk drive which has been described as the default system disk unit, and "2" has been specified as the disk identification for this program, you will instructed to placed the FILE MAINTENANCE DISKETTE on XX where XX is the system drive.

In the event that one of the special functions has been used as the PROGRAM name, a zero should be used as the DISK IDENTIFICATION.

Example:

1:	MENU SEQUENCE KEY	10
2:	PROGRAM DESCRIPTION	FILE INITIALIZATION
3:	PROGRAM	SUB1
4:	DISK IDENTIFICATION	0

When this entry is selected, submenu 1 will be displayed and another entry may be selected.

9.2 CREATING DEFAULT MENU CONTROL DATA

If you do not wish to define each entry in the main menu program, the option may be selected to create the default menu control. If this option is selected, the program will first check to determine if any menu control information has been previously defined. If it has, you will be given the option to replace it or to exit.

When default menu control data is created, menu entries will be created to invoke all programs to process all of the files which have been defined. If a MASTER FILE, TRANSACTION FILE, and HISTORICAL TRANSACTION FILE have been defined, entries will also be created to invoke the posting and closing programs.

If only one file has been defined, no submenus will be defined. However, if more than one file has been defined, two submenus will be created. The first submenu will contain the entries required to invoke the file initialization subroutines. The second submenu will invoke the programs used to edit the system control data.

In some cases, the programmer may want to create the default menu control data, and then modify it to meet the specific needs of the system being developed.

PEARL Level 3 provides for both a user-defined menu and a default system generated menu. If you defined a file and a default menu generated, then the default menu will contain references to report programs whether report programs have been defined or not. In the evvent that a report program is not needed for a file, then the default menu record for the report program should be deleted.

CHAPTER 10. REPORT DEFINITION

PEARL Level 3 requires you to define all reports needed for the application to be generated. Generated reports support the following features :

1. The inclusion of more than one report for any given file.

2. The processing of multiple files (transaction, and historical transaction files only).

3. The processing of a hierarchy of subtotal control breaks and printed statements.

4. The insertion of user-defined code into the body of the report subroutines at four key positions.

5. The formatting of the report printed layout through simplified instructions.

6. Sorting random access files for report processing.

7. Sequential accessing of indexed data files by primary and secondary keys.

This chapter will cover the following points:

Two programs are provided to create report control data. The first of these two is used to identify a report (option 6 on the main menu), and the second is used to define detail report control (option 7 on the main menu).

10.1 REPORT IDENTIFICATION.

When the program to define or edit report control data (option 6 on the main menu) is loaded, the following menu is provided:

```
0     RETURN TO MAIN MENU
1     TO DEFINE A NEW REPORT
2     TO EDIT REPORT CONTROL DATA
3     TO CHANGE REPORT ID (CURRENT REPORT = nn)
4     TO DELETE CONTROL FOR SPECIFIED REPORT
5     TO ENTER/EDIT REPORT DETAIL
```

(OPTION)

(1),(2) These two options are used to define and identify general requirements for each report. These include the following which will be discussed in detail:

a- Report id number.
b- Primary file number.
c- Secondary file number.
d- Report width.
e- Report depth.
f- Processing sorted files.
g- The report subroutine label.
h- The report description.

(3) A report id is any number between 1 and 99. Each report is related directly to a file (see discussion of Primary and Secondary files below). Any combination of the report id and the file id numbers may be used. A report must be defined before the report control data may be edited.

(4) If Option 4 is selected, all of the control data for a specified report (by report number) will be deleted from the control file.

(5) If Option 5 on the menu is selected, you may go directly to the program to define the report detail data.

You will be prompted to enter the following information when a report is defined:

REPORT ID NUMBER

This is a number in the range between 01 and 99. Each report defined within PEARL must have a unique number. The report id is not used in the generated code. A report id must be established prior to editing report control data, or prior to adding or editing report definition data.

PRIMARY FILE NUMBER

This is the number identifying the file for which the report will be generated. This file will be read sequentially during report generation. (This may be sorted prior to report processing as indicated below.)

This number may not be altered once a report has been defined, and one or more data elements to appear on the report have been entered as extracted from this file.

SECONDARY FILE NUMBER

This is the number identifying a secondary file. If this number is zero, it is assumed that no secondary file will be processed.

Secondary files may only be used by a transaction or an historical transaction file report to access the master file records.

The use of a secondary file provides for data which comes from other than the primary file to be inserted in the report. Take the following example:

> A report routine is being defined to create a transaction file report. This transaction file is related to a master file through a customer id number. We want to place the customer name on the report which is being generated. We have defined the primary file as file 2 (the transaction file) and we define the secondary file as file 1 (the master file).
>
> We have further defined in our data base structure that the TT.MM.CUS.ID% variable in our transaction file is logically connected to the MM.CUS.ID% variable which is the primary key for our master file record. Therefore, each transaction record accessed during report generation will cause a customer record from the master file to be accessed, and all of the data items on the master record will be available for processing on the report.

REPORT WIDTH

The width of the report in characters. The value entered here will be used during the report processing to determine the center of the report when the option to center a variable on a print line is requested.

REPORT DEPTH

The number of lines from the top of one page of the report to the top of the next page. If this variable is used, it will override the form length which is indicated in the generated system configuration control data. This control may also be used when a detail or heading line is defined to print a specified number of lines up from the bottom of a page.

PROCESS SORTED FILE (Y/N)

If a "Y" is used here, the file may be sorted prior to report processing. An extended discussion of the entry of sort control information is provided in Section 10.4.

REPORT SUBROUTINE LABEL

This value indicates the label to be assigned to the first subroutine line created to generate the report. In addition to this number, the **UNIQUE FILE ID** assigned to the primary file will be used as a prefix in addition to this number to identify the BASIC source file to be included in the mainline report writers.

PROGRAMMING NOTE:

If more than one report is defined for a file, you should insure that each report has a unique REPORT SUBROUTINE LABEL. Each label should be an even multiple of 1000, and a range of 1000 should be allowed. For example, if 50000 is used, the labels between 50000 and 50999 should be reserved for the report subroutines.

REPORT DESCRIPTION

This is a phrase describing the report. It will not be included on the report itself, but will appear in the main menu of the report control program to allow the end user to select a specific report at the time reports are created.

10.2 DESIGNING PEARL GENERATED REPORTS.

In order to use the REPORT GENERATION capability provided by PEARL Level 3, the programmer should 1) carefully design and layout the report which is desired, and 2) be aware of the structure and logic flow of the generated report writing routines.

The most effective use of the entry routines for report control information can be made by using a standard layout form to define the report, determining ahead of time all of the calculations which will have to be made, and the sequence of computation logic which will be required during report generation processing.

The generated report writer routines are segmented in such a manner to support most of the conditions which are encountered during report generation. The following is a list of the main segments involved in report control.

Control break checks.

Printing of report headings.

Printing the detail lines.

Printing multiple levels of subtotal print statements when related control breaks are encountered.

Process user-defined detail, subtotal, grand total, and report initialization in CBASIC code.

Spacing to the top of the form.

CHAPTER 10. REPORT DEFINITION

Anyone who has designed and written a complex report writing program knows that sequence of execution is critical in order for the report to handle various combinations of data which may occur in the end user environment.

Before a you attempt to define a report which includes multiple levels of subtotaling and computational code, it would probably save time to first define a simple report, generate the report control subroutine, and study the listing of the program. Once a you become familiar with the points at which code is inserted in the program, you will be able to make the best use of the capability to insert computations in the programs as they are being generated by PEARL.

The following is a summary of the sequence in which execution of subroutines will occur as controlled by a header routine in the report generation subroutine.

1. GET RECORD TO BE PROCESSED
2. CHECK FOR CONTROL BREAK
3. SUBTOTAL COMPUTATIONS
4. SUBTOTAL PRINT
5. GRAND TOTAL COMPUTATIONS
6. GRAND TOTAL PRINT
7. DETAIL COMPUTATIONS
8. DETAIL PRINT
9. SUBTOTAL ACCUMULATION
10. GRAND TOTAL ACCUMULATION

The SUBTOTAL COMPUTATIONS will be executed only when subtotal control break has occurred, or if an end of file has been encountered.

The GRAND TOTAL COMPUTATIONS will be executed only when a an end of file has been encountered.

CODE GENERATION.

The following is a general discussion of how programmer-defined code is inserted in the report generation routine.

The users of PEARL Level 3 are provided selective entry points for the insertion of CBASIC code or code instruction. The insertion of such code allows the definition of report layouts and cross file data element relationships.

There are two types of user-defined code. One is called **"Free-Form CBASIC2 Code"** and the other is called **"Code Instruction"**.

FREE-FORM CBASIC2 CODE (COMPUTATIONAL DATA).

Free-form code is stored on the PEARL definition file as records. The length of each line of code may not exceed 60 characters, and must represent CBASIC2 source code.

Each line of code is assigned a record sequence number in the same manner as Data Elements are assigned a sequence number. Record sequence numbers are grouped into functional units with a starting and ending sequence number that corresponds to a starting and ending address in the generated code.

Example:

A report was defined with a base address at 50000. When defining a report that requires detail computations, you may enter such detail computation code using sequence keys:

4000 to 4499

The code on the definition records with these sequence keys will be inserted into the Report Writer subroutine between the line addresses of 50300 and 50399.

Each code instruction is stored on the PEARL Level 3 definition file as a single record with a sequence number. Sequence numbers are grouped into functional units. Each unit constructs CBASIC code to perform a specific task. Unlike free-form code, code instruction does not provide for the input of line numbers.

CODE INSTRUCTION (REPORT CONTROL DATA).

Code instruction is mainly used to define the report formats and totals control breaks. The purpose of code structure is to allow you to input a minimum of information and allow PEARL Level 3 to generate the CBASIC2 code.

CODE INSTRUCTION FOR PRINTING.

Code instruction provides the input of the following information:

The variable,

The mask,

Tab position,

Number of lines to skip before or after a print statement.

Example:

A report was defined with an address base of 50000. The report is to print the number of items sold. The input would be:

```
sequence Number = 5000
the Variable = ITEM%
the Mask = NUMBER OF ITEMS ####
the Tab = 10

number of lines to skip before printing = 2
number of lines to skip after printing = 2
```

PEARL Level 3 will generate the following code:

```
PRINT : PRINT
PRINT USING "NUMBER OF ITEMS ####";TAB(10);ITEM%;
PRINT : PRINT
```

The following is a reference for user-inserted code sequence numbers and the generated code address into which the code is inserted. Note that insertion of code will be entered **between** the address ranges outlined in Section 10.2.

CODE INSTRUCTION FOR CONTROL BREAKS.

Control breaks are used to check for the changes in the data being read from the file. These changes may occur in data record detail fields or within these fields.

The information items required to process control breaks are:

A variable name, and

The position and length of the field on the data record.

Example:

A report was defined to print out the dollar volume for each department in a clothing store. Each transaction record contains the following variables:

1- DEPARTMENT.ID%
2- DOLLAR.SALES

The following should be added as part of the detail report definitions as a control break definition:

1- Sequence number = 3000
2- The variable = DEPARTMENT.ID%

The following code will be generated and processed each time a record from the file is read:

```
IF T.DEPARTMENT.ID% <> DEPARTMENT.ID% THEN \
T.DEPARTMENT.ID% = DEPARTMENT.ID% :\
RCODE = -10
```

Explanation:

A temporary variable (T.DEPARTMENT.ID%) is set up to store the value of the control break control variable (DEPARTMENT.ID%). When a record is read, the value for (DEPARTMENT.ID%) is compared with the temporary variable. If they are not the same, then the temporary variable is set to the value, and a return code (RCODE) is returned with a value indicating the change in the department number. This return code is the communicator of control breaks to all routines that depend on such control breaks.

INSERTED CODE ORGANIZATION

PEARL Definition Sequence Keys Range		Generated Program Address Range
0001 - 0999	Initialization Computation	xx001 - xx019
1000 - 1999	Heading Control	xx100 - xx199
2000 - 2999	Total Control	none
3000 - 3999	Control Break Control	xx200 - xx299
4000 - 4499	Detailed Computation	xx300 - xx349
4500 - 4799	Subtotal Computation Control	xx350 - xx379
4800 - 4999	Grant Total Computation Control	xx380 - xx399
5000 - 5999	Detail Line Print Control	xx800 - xx900
6000 - 6999	Subtotal Print Control	xx400 - xx450
7000 - 7999	Grand Total Print Control	xx500 - xx550

10.3 DETAIL REPORT CONTROL DATA.

This section describes the method that is used to define detail report control information. The following topics will be covered:

The menu.
The sequence keys.
Report initialization code.
Report headings.
Totaling control variables.

Control breaks.
Computational code.
Detail line definition.
Subtotal line definitions.
Grand total line definitions.

THE MENU

When the program to define report detail is loaded, the following menu will be displayed:

```
0     RETURN TO MAIN MENU
1     TO ADD ITEMS TO THE REPORT
2     TO EDIT REPORT ITEMS
3     TO DELETE ITEMS FROM THE REPORT
4     TO CHANGE THE REPORT ID (CURRENT REPORT =nn)
5     TO PROCESS REPORT CONTROL DATA
```

THE SEQUENCE KEYS

When you select to add, edit, or to delete items from the report control data, a request will be made by the program to enter the sequence key. The ranges for the sequence control keys are as follows:

Sequence Key	Control Group
000-999	Report initialization
1000-1999	Heading control
2000-2999	Total control
3000-3999	Control break control
4000-4499	Detail computation control
4500-4799	Subtotal computation control
4800-4999	Grand total computation control

Sequence Key	Control Group
5000-5999	Detail line print control
6000-6999	Subtotal print control
7000-7999	Grand total print control

Depending upon which sequence key is entered, you will be requested to enter control information pertaining to the control group.

When adding, editing, or deleting report detail control information, the sequence key will refer to the current report id only. You operator must change the report id to the appropriate number prior to selecting options 1, 2 or 3.

If option 5 is selected, the program will chain directly to the program to define reports, or to edit the report control data as described above.

The following is a description of the report detail control information.

REPORT INITIALIZATION (FREE-FORM CBASIC2 CODE).

When the sequence key is in the range 0001 through 0999, computational code may be entered to be executed prior to processing the file to generate the report. See Section 10.2 for additional explanations on how this code is entered and used in the generated report processing program.

HEADING CONTROL (CODE INSTRUCTION).

The following discussion on print statements is applicable to the format and control of:

Report Headings
Report Detail Lines (Section 10.2)
Report Subtotal Lines (Section 10.2)
Report Grand Total Lines (Section 10.2)

Print statements defined for any of the above generally consist of the following:

1 - A variable
2 - A TAB position
3 - A formatting control mask
4 - And the number of lines to skip before or after a statement is printed

VARIABLE NAME

The variable code name is optional. If a blank is used in place of a variable name, the program assumes that a literal is to be printed as indicated in the print mask.

The specified variable code names may identify variables from a number of sources as indicated by a control character or prefix on the variable as follows:

Special variables:	CC.DATE$	current system date
	CC.PAGE%	current page in the report
	CC.LINE%	current line in report body
	CC.INSTALL$	Installation name

File variables: When a variable is to be processed as accessed from the primary file, the variable name should be entered as defined on the file.

Totaled variables: When a total or a subtotal is to be printed, the prefix "TOT." or "STOT." should be added to the standard variable name. (Note that totaling for the variable must also be indicated in the totaling control data discussed below.) For example, if the variable MM.COST were defined on the file, and the control to total and subtotal this variable were defined, then the variables TOT.MM.COST and STOT.MM.COST could be defined to be printed.

The code to compute totals is included for each variable to be totaled. In addition, these variables are cleared (reset to zero) each time subtotals and totals are printed.

Computational variables: The user may define the printing of any valid CBASIC variable in a report. If variables are defined through the entry of computational code (described below), they may be included in the report control in the same manner as variables defined on the system files.

TAB POSITION

The tab position indicates where the item should be printed on the print line. If a value of zero is entered, the tab position will be computed at run time so the variable being printed is centered in the middle of the page.

FORMATTING CONTROL MASK

This defines a string of characters which is valid in a CBASIC PRINT USING statement. If a variable name is not specified, a string may be entered. For example, a title for a report may be indicated as follows: (CUSTOMER MASTER LIST). Literal values and variables may be combined. For example, (CURRENT DATE = &) may be used when the variable CC.DATE$ is specified as the variable, or (PAGE: ###) may be used with the variable CC.PAGE.

LINES TO SKIP

PEARL Level 3 formats print statements that are contiguous. In order to start printing on a new line, a line must be skipped. Lines may be skipped before a variable is printed so it is printed on a new line, or a line may be skipped after a variable is printed so consecutive printed data is printed on a new line.

LINES TO SKIP BEFORE PRINTING

This control may be used to force the insertion of blank lines prior to printing the current item. It should be used only for the first variable being defined on the current line.

LINES TO SKIP AFTER PRINT

This indicates the number of lines to be skipped after the specified variable is printed. If several more variables are to be printed on the same line, a value of zero should be used. If the item being printed is the last item on the line, and there are to be no blank lines between the current line and the next line, a value of 1 should be used. If there are to be blank lines inserted between the current line and the next line to be printed, the number should be indicated by the value. For example: 2 will cause one blank line to be inserted, 3 will cause 2 blank lines to be inserted, etc.

TOTAL CONTROL DATA (CODE INSTRUCTION)

This control information is used to indicate which variables should be totaled during report generation.

A key in the range between 2000 and 2999 is required.

VARIABLE TO BE TOTALED

This is the name of the variable to be totaled. It may be a key which is defined on the primary file, or which is initialized during the detail computation defined by the control data. When a variable is identified as a variable to be totaled or subtotaled, the variable will be automatically incremented for each record processed, and will automatically be set to zero each time it is printed in a report.

ACCUMULATE SUBTOTAL (Y/N)

If yes is specified, all subtotals will be accumulated. The variable will be output in the subtotal print line as indicated in the control data for printing the subtotal data. The point at which subtotals will be printed is defined in the control break control data.

<u>ACCUMULATE GRAND TOTAL (Y/N)</u>

If yes is specified, the amount of the variable will be accumulated in a grand total which will be printed according to the grand total print control. This data will be printed at the time an end of file is encountered.

<u>PROGRAMMING NOTE:</u>

MM.XYZ is identified as a variable for which both a subtotal and grand total will be maintained. In this case, TOT.MM.XYZ and STOT.MM.XYZ will be incremented by the value in MM.XYZ each time a record is processed. MM.XYZ will not be modified. The user must declare in a subtotal or grand total print control statement to print STOT.MM.XYZ and TOT.MM.XYZ. Immediately after these variables are printed, they will be reset to zero.

CONTROL BREAK DEFINITION (CODE INSTRUCTION)

The control break information is used to indicate the point at which subtotal lines should be printed.

When control breaks are defined, the order of their definition should start with the lowest level control break variable and end with the highest level. The subtotal print control information must also be entered in the same exact sequence.

<u>Example:</u>

A report is to be written for a welding supply outfit. A report to be generated is to list the monthly activity of each of its three departments by customer id. The control break and printed data should be entered in the following sequence:

1 - Customer ID
2 - DEPARTMENT
3 - MONTH

The result is:

When Customer ID changes, then the customer break print statement(s) will be processed;

When DEPARTMENT ID changes, then the customer and the department break point statements will be processed;

When the MONTH changes, then the customer, the department, and the month break point statements will be processed.

Multilevel control breaks can be used for sorted files. For example, if a file is sorted in order by Department, job number, and employee, three control breaks may be specified to cause subtotals to be printed at each level. When this is done, the control breaks control and the subtotal print control should be specified in the reverse order as used for sort control.

SEQUENCE CONTROL KEY

This is a key in the range between 3000 and 3999.

VARIABLE NAME FOR CONTROL BREAK

A variable may be specified here to control the point at which subtotals will be printed. If a variable is specified, a subtotal will be printed each time the specified variable field changes.

If you wish to indicate control breaks through the use of an absolute record position, enter a blank.

RECORD POSITION FOR CONTROL BREAK

If the user wishes to specify the control break by an absolute record position, the position should be entered here. If the control break is controlled through the use of a variable, enter a zero here.

LENGTH OF CONTROL FIELD

When a record position has been specified, the length of the control field should be specified here.

COMPUTATIONAL CONTROL DATE (FREE-FORM CBASIC CODE)

In the event that the PEARL Level 3 report generation options do not cover your needs, you may enter actual CBASIC2 code to be executed prior to the printing of detail, subtotal, and grand total lines.

SEQUENCE CONTROL KEY

This is a key in the range of 000 and 999, and 4000 to 4999. If the key value is in the range between 000 and 999, then the computational code will be inserted at the highest level in the report writer subroutine and will be executed only once for each routine. If the value is in the range between 4000-4499, the computations specified will occur prior to printing detail information. If the value is in the range between 4500-4799, the computations specified will occur prior to printing subtotal information after a subtotal control break has been detected. If the value is in the range between 4800-4999, the computations will occur prior to printing the grand totals for the report.

COMPUTATION CODE

You may enter up to 60 characters of CBASIC code. Each line of code will be inserted into the generated program as defined, and in the sequence provided by the sequence key.

Example:

An inventory file has been defined which contains in the master file a count (II.COUNT) and the cost per item (II.COST). The user wishes to determine the value of the number of items in stock. The following code is used:

```
COM.VALUE = II.COUNT * II.COST
```

PROGRAMMING NOTES:

1. Subtotal and grand total computation code will be executed only if a subtotal control break or an end of file, respectively, have occurred. Detail computations will occur prior to printing the detail print line for each record processed.

2. Conditional branches to labels included in the computational code may be used in the computational code defined. The labels must be within the following ranges:

xx301-xx349	(detail computations)
xx351-xx379	(subtotal computations)
xx381-xx399	(grand total computations)

The xx portion of the label is the thousands portion of the REPORT SUBROUTINE LABEL (see Section 10.1) which was specified for the report. The labels xx301, xx351, and xx381 are entry labels for each computational control block. xx349, xx379, and xx399 are the last labels used for each block and may be used for exit points.

DETAIL LINE PRINT CONTROL (CODE INSTRUCTION)

Print control for detail lines are defined in the same manner as described in Section 10.2. The range of the sequence keys is between 5000 and 5999.

SUMMARY SUBTOTAL PRINT CONTROL (CODE INSTRUCTION)

Print control for summary subtotal lines are defined in the same manner as described in Section 10.2. The range of the sequence keys is between 6000 and 6999.

Every control break is associated with a subtotal print control block. A subtotal print block consists of two or more subtotal print control statements. The first statement must contain three asterisks (***) in the print mask. This is the only method by which the generation routines can separate print statements from one subtotal break level to another. Each subtotal print control break block may contain any number of print statements up to the maximum allowed by the range of sequence keys.

NOTES:

1. In order to use summary subtotal lines, a control break control entry must also be made. If this definition is not done, then no summary line will be printed.

2. After each summary subtotal is printed, the STOT variables into which the subtotals have been accumulated will automatically be reset to zero.

GRAND TOTAL PRINT CONTROL (CODE INSTRUCTION)

Print control for summary grand total lines is defined in the same manner as described in Section 10.2. The range of the sequence keys is between 7000 and 7999.

The grand totals will be printed when an end of file is encountered on the primary file. All TOT variables will be set to zero when the report is complete.

NOTE:

The TOT variables are accumulated as each detail record is processed. This allows you to use the TOT and STOT variables in detail and summary lines alike, so that an accumulative total may be printed during report processing.

10.4 OPERATING THE GENERATED REPORT PROGRAMS.

PEARL Level 3 reports support the following functions:

10.4.1 List indexed files by the primary key or any of the nine (9) secondary keys defined at Data Element Definition time.

When you select report processing, you may get a report using the secondary key as well as the primary key. To get a report on the secondary key, enter: nK# where n equals the report menu option, and # equals the secondary key number, i.e., 2K1 = report 2, secondary key 1. If the secondary key number is not known, then enter the report option followed by "K". The will cause a list of secondary keys to be displayed for the operator. (See Chapter 4, Step 10)

10.4.2 Sort non-indexed files using "QSORT.COM" and "XSORTX.INT".

Random aaccess files maintained by PEARL generated systems usually contain records that are not ordered by any specific file except the record number. Sorting data files before reports are generated can provide for more efficient processing time and a simplified report design.

The PEARL-generated report programs for random access files will interface with the QSORT program (a program product of Structured Systems). Sort control is entered directly by the end user and is saved in a sort control file accessed by the PEARL-generated programs prior to sorting a file.

The following are some considerations related to sorting files maintained by PEARL-generated programs:

QSORT provides sorting capabilities for up to five sort fields. However, only four of these fields are available to the user for sorting files. The fifth sort field is used internally by the sort routine (XSORTX) to maintain the integrity of the sorted files and to insure their proper processing by the generated subroutines.

In order to determine the position and length of each field to be used as a sort control key, refer to the detail data element reports which are generated by PEARL. These reports specify the position of each variable on the file, and the length of each field.

You may or may not want to have the sort fields correspond to the control breaks defined for a report. This option is open to the designer of a PEARL-generated system and the needs of the end user of the system.

See Appendix I for details on setting up and executing sorts on generated system files.

10.4.3 To clearly distinguish one report from another, a report identification phrase may be entered. This phrase will be requested from the operator prior to printing the report. In order for this phrase to appear on the report, the report must contain the print statement variable CC.REPORT.ID$. (See Chapter 4, Steps 10 and 7, respectively.)

10.4.4 Each report will start at page number 1. If any other page number is desired, then you should enter the page number as "P=#" (without the quotes) where # equal number other than 1, i.e, P=3. The variable CC.PAGE% must be present in the report definition. (See Chapter 4, Step 10.)

10.4.5 If single sheets of paper (as opposed to form feed paper) are to be used for printing of reports, then enter "PAUSE" (without the quotes) when prompted. This will cause the printing to stop at the top of each new page for insertion of the next sheet of paper. (See Chapter 4, Step 10.)

NOTE:

When requested from the operator, Steps 3, 4 and 5 may be entered simultaneously, separated by commas in the order presented on the video screen. For example: Report 1 of 10,P=3,PAUSE.

10.5 GENERATED REPORT CONTROL VARIABLES.

The following are control variables used to govern the processing of the reports:

CONTROL VARIABLES :-

RCODE=0	SEQUENTIAL PROCESSING AND PRINTING
RCODE=-1	END OF FILE. SIGNALS GRAND TOTAL PRINTING
RCODE=-10,-20..	SUBTOTAL BREAK. PRINT SUBTOTAL
REPORT.WIDTH%	# OF CHARACTERS IN ONE LINE OF REPORT
REPORT.DEPTH%	# LINES ON A REPORT PAGE
PCODE=0	PRINT PROCEEDING TOTAL OR SUBTOTAL
PCODE<>0	TEMPORARY SUPPRESS OF TOTAL OR SUBTOTAL PRINTING
UCODE=0	UPDATE TOTAL OR SUBTOTAL VARIABLE
UCDOE<>0	TEMPORARY SUPPRESS OF UPDATING OF TOTALS
HCODE=0	SPACE TO TOP OF FORM AND PRINT HEADING
HCODE<>0	SUPPRESS THE SPACING TO TOP OF FORM AND THE PRINTING OF HEADINGS FOR THE DURATION OF THE REPORT
CC.REPORT.ID	ENTERED BY USER AT TIME OF REPORT PRINTED WITH HEADING
CC.PAUSE% 0	PAUSE AT TOP OF EACH PAGE
CC.START%=0	START OF THE REPORT
KCODE=-1	IF MASTER FILE RECORD NOT FOUND

The following is a detailed description on how the control variables work during the report process.

THE VARIABLE	DESCRIPTION
CC.REPORT.ID$	This variable must be added to the report definition data if it is to be used in the report. It is usually placed on the report header and is used by the operator of the report to enter a special REPORT ID at the time the report is to be written.
CC.PAGE%	This variable is used to print and increment the page number. If a page number is to appear on the report, then it must be defined by the operator on the report definition file. The starting page number is always 1. This may be overridden by the operator at the start of every report to be written.
CC.PAUSE%	This variable is used to control the pause at the top of every page feature of the report writer. The operator will be prompted for an option to pause at the top of every page in the event that single leaf paper is used.
CC.START%	This variable is used by the report routine to bypass the break control subroutine on the first record read.
HCODE	This variable is used to control the printing of the report heading during the time the report is generated. If HCODE=0, then the report heading will be printed. If HCODE <> 0, then the report heading will not be printed. This variable will be set zero at the start of every report.
REPORT.WIDTH%	This variable represents the width of the report page for the purpose of centering printed statements. This variable is defined on the report control record.
REPORT.DEPTH%	This variable represents the length of the report page. It is defined in the report control record. If the report page length is not defined, then the default will be the report page length defined on the configuration file of the generated system at run time.

RCODE	This variable is a general return code for most generated subroutines. This variable communicates the status of execution of subroutine to other calling subroutines. In the report writer, it functions as an indicator of the following:

RCODE<-1
All RCODES with a value less than -<9 indicate that a subtotal control break has been encountered. Subtotal breaks are organized in a hierarchy dependent on the control break definitions.

***RCODE=-10
Indicates that the first subtotal break condition has been met. This will cause the following:
1 - The subtotal computational code to be executed.
2 - The first subtotal print statements to be printed.
3 - The clearing oF all numeric variables prefixed by (STOT.) that appear in the first subtotal print statements.

***RCODE=-20
Indicates that the second subtotal break condition has been met. This will cause the following:
1 - The same as RCODE=-10
2 - The second subtotal statements to be printed.
3 - The clearing of all numeric variables prefixed by (STOT.) and used in the second subtotal print statements.

RCODE =-30
RCODE=-40
"
" same process as in RCODE=-10 and RCODE = -20

	***RCODE=-1 Indicates the end of file. This triggers the execution of all the subtotal code, print statements, and clearing of subtotal variables and: 1 - Processing of grand total computational code. 2 - Processing all grand total print statements. 3 - Clearing all variables with a prefix (TOT.) appearing in the grand total print statements.
***PCODE	This variable is used to control the printing of detail, subtotal, and grand total statements. If PCODE=0, then print statements will be executed as usual. If PCODE<>0, then PCODE will be reset to zero at a print statement, and print statement will be bypassed. PCODE is a temporary suppressor of print statements and must be reset to suppress each individual detail, subtotal, and grand total print statement executed. Subtotal and grand total accumulation variables prefixed by (STOT.) and (TOT.), respectively, will not be cleared if they appear in the suppressed print statement.
***UCODE	This variable controls the updating of subtotal and grand total accumulation variables. If UCODE=0, then subtotal and grand total figures will be updated. If UCODE<>0, then such variables will not be updated. UCODE is reset to zero when the processing for the current period is complete.

*** Given that a variable (accumulate) was defined as a subtotal
and a grand total, the following code will be generated:

1- THE PRINT STATEMENTS

```
REM SUBTOTAL PRINT STATEMENT
     IF PCODE<>0 THEN PCODE=0 RETURN
     PRINT
     PRINT STOT.ACCUMULATE
     STOT.ACCUMULATE=0

REM GRAND TOTAL PRINT STATEMENT
     IF PCODE<>0 THEN PCODE=0 RETURN
     PRINT
```

```
          PRINT TOT.ACCUMULATE
          TOT.ACCUMULATE=0
```

2- THE UPDATE STATEMENTS

```
     REM SUBTOTAL UPDATE
          IF UCODE<> 0 THEN RETURN
          STOT.ACCUMULATE=STOT.ACCUMULATE+ACCUMULATE

     REM GRAND TOTAL UPDATE
          IF UCODE<>0 THEN RETURN
          TOT.ACCUMULATE=TOT.ACCUMULATE+ACCUMULATE
```

CHAPTER 11. ENTERING POST/CLOSE COMPUTATIONS

The purpose of posting and closing is to facilitate periodic cross-file transfer of information. Other facilities generated by PEARL Level 3 to conserve disk space during the posting and closing process are:

1. Reinitializing the transaction file.

2. Posting transactions to a historical file.

11.1 THE RELATIONSHIPS BETWEEN POSTING AND CLOSING PROGRAMS.

There are three main processes involved in posting and closing. The function of each is to provide periodic transfer of information. The periodicity provided is:

1. Current period processing (posting).

2. Ending the current period (period-end closing).

3. Ending the current year (year-end closing).

The frequency of execution of the above processes is totally under the control of the user. The process of transferring data from one periodic category to the next is irreversible.

11.2 THE GENERATED ROUTINES.

Three routines are generated, along with three included subroutines. Each routine or subroutine is identified by the transaction file prefix and a routine number.

<u>Example:</u>

Transaction file prefix = TR

The POSTING routines:

```
     Main Routine = TR200.BAS
     Subroutine   = TR20000.BAS
```

The PERIOD END CLOSING routines:

```
     Main Routine = TR300.BAS
     Subroutine   = TR21000.BAS
```

The YEAR END CLOSING routines:

```
     Main Routine = TR400.BAS
     Subroutine   = TR23000.BAS
```

The posting and closing routines will not be generated unless both a transaction file and a master file have been defined. If an historical transaction file has also been defined, then it will be processed by the closing routines.

User-defined data elements are not manipulated and such manipulations are dependent on the insertion of user-defined code as described in sections 11.4, 11.5, and 11.6.

Example:

A data variable on the master file was defined as MASTER.TOT.ITEM, and it is to be updated at transaction posting time by a transaction file record containing the data variable TRANSACTION.ITEM. The relationship of the two variables is:

```
MASTER.TOT.ITEM = MASTER.TOT.ITEM + TRANSACTION.ITEM
```

To have the master record updated, the following steps are taken:

1. Select option 8 on the PEARL Level 3 main menu (ENTER/EDIT POST/CLOSE COMPUTATION).

2. Select option 1 (TO ADD RECORDS).

3. Select a sequence number between 0000 and 0499.

4. Enter the following code:

```
MASTER.TOT.ITEM = MASTER.TOT.ITEM + TRANSACTION.ITEM
```

11.3 THE CONTROL OVER POSTING AND CLOSING.

Posting, current period closing, and year-end closing are monitored through four control variables maintained on the first control records of the files involved. Given that the file prefix is xx, the control variables are:

Variable	Description	File Location
xx.S.POST.COUNT%	Posting record count	Master Transaction
xx.S.JE%	Last transaction record number posted	Master Transaction
xx.S.CLOSE.COUNT%	Period-end closing record count	Master Transaction (History)
xx.SH.JE%	Last transaction record number closed (posted to history file)	Master Transaction (History)

11.3.1 Posting Controls.

Posting of transactions to the master file is permissable at any time provided that all previous posting and closing processes were successful. Otherwise, an error will occur.

When the posting process starts, the following control variables will be set:

1. Transaction file

   xx.S.POST.COUNT% = -999

2. Master File

   xx.S.POST.COUNT% = 0

At the end of posting, the posting variables on both files will be set to:

xx.S.POST.COUNT% = number transactions posted

xx.S.JE% = last transaction posted

11.3.2 Period-End Closing Controls.

Period-end closing will occur only if all transactions on the transaction file have been posted to the master file, and if all previous processing of the files has been successful. Otherwise, an error will occur.

If an historical transaction file was also defined and generated, the check is made to verify that the history file is also in sync with the transaction and master files.

When the period-end closing process starts, the following control variables will be set:

1. Transaction file

   xx.S.CLOSE.COUNT% = -999
   xx.SH.JE% = 0

2. Master file

   xx.S.CLOSE.COUNT% = 0
   xx.SH.JE% = 0

3. Historical file

   xx.S.CLOSE.COUNT% = -999
   xx.SH.JE% = 0

When the period-end closing process is completed, the transaction count variables will be set to the number of transactions processed, and the last transaction number will be updated on all files processed.

In addition, the posting variables will be set to zero.

11.3.3 Year-End Closing.

Year-end closing will only occur if all transactions have been successfully posted and closed. An error will occur if these conditions do not hold true and if any previous processing was not successfully completed.

Year-end closing treats the control variables in the same manner as period-end closing except that at the end of a successful process, all variables are initialized to zero.

11.3.4 Error Codes

-999 = Previous posting failed

-998 = Previous closing process failed

-997 = Transaction and master files are not in sync

-996 = Files are not in sync

-995 = Posting not completed

11.4 THE POSTING ROUTINES

The posting routines process the transaction file and the master file. The following is the menu provided, preceded with a description of what each option provides:

```
0      TO RETURN TO MAIN MENU
1(P)   TO POST TRANSACTIONS TO THE MASTER FILE
2(P)   TO CLEAR CURRENT DATA FROM MASTER FILE
```

(If the option entered is followed by a "P"; then a printed report will be generated.)

11.4.1 Option 1 - TO POST TRANSACTIONS TO THE MASTER FILE.

Posting transactions to the master file proceeds as follows:

1. Open transaction file,

2. Open master file,

3. Read unposted transaction record,

4. If end of file, then 9,
5. Read the corresponding master file record,
6. Process user detail code (see Appendix B and Example 11.2), *
7. Write back the last master file record,
8. Go to 3,
9. Process end of file user defined code (see Appendix B and Example 11.2), **
10. Close master file,
11. Close transaction file,
12. Return to menu.

An error will occur in the event that a master file record referenced by a transaction file record was not found.

The user may insert free-form code for the processing of data (see Appendix B for details). There are two points at which code may be inserted. These are:

		Sequence Number	Description	Address
*	1	0000 - 0499	Detail posting code	20100 - 20399
**	2	0500 - 0999	EOF posting code	20400 - 20499

11.4.2 Option 2 - CLEAR CURRENT DATA FROM MASTER FILE.

This option will provide a means to restart the posting process in the event the previous posting process failed or errors were detected in the transaction file. To restart the posting process, all previous posting data must be reversed or deleted.

The clearing process is as follows:

1. Open master and transaction files,
2. Read master file record,
3. If end of file, then 7,
4. Process user-defined backout code, *
5. Write back last master file record read,
6. Process EOF user-defined backout code, **

7. Reset control variables on both files to the starting values,

8. Close both files,

9. Return to menu.

The inserted code:

	Sequence Number	Description	Address
*	1000 - 1499	Detail backing out code	20600 - 20899
**	1500 - 1999	EOF backing out code	20900 - 20999

11.5 PERIOD-END CLOSING.

The period-end closing routines process the master file, transaction file, and the historical transaction file (if defined). The purpose of the closing routines is to provide the user with a process by which data may be periodically categorized and to reinitialize the transaction file once all information has been distributed from it and validated on the master file.

If an historical transaction file is defined, then for each transaction record closed, an historical file record will be added to the historical transaction file.

The period-end closing process proceeds as follows:

1. Open master, transaction, and history file,

2. Read master file record,

3. If end of file, then 7,

4. Process user-defined master file detail code, *

5. Write back last master record read,

6. Go to 2,

7. Read transaction file record,

8. If end of file, then 12,

9. Process user-defined detail transaction code, **

10. Add a historical transaction record,

11. Go to 7,

12. Process EOF user-defined code, ***

13. Reinitialize transaction file,

14. Process transaction reinitialization code, ****

15. Close all files,

16. Return to menu.

The inserted code:

	Sequence Number	Description	Address
*	2000 - 2499	Master file detail code	21100 - 21299
**	2500 - 2999	Transaction detail code	21300 - 21499
***	3000 - 3499	End of file code	21500 - 21599
****	3500 - 3999	Transaction reinitial-ization Code	22100 - 22499

11.6 YEAR-END CLOSING.

The year-end closing process allows the user to re-initialize all files for a new accounting year:

The year-end closing process proceeds as follows:

1. Open master file,

2. Read master file record,

3. If end of file, then 6,

4. Process user detail code, *

5. Go to 2,

6. Process end of file code, **

7. Close master file,

8. Open transaction file,

9. Read in control records,

10. Process user-defined transaction initialization code, ***

11. Initialize new transaction file,

12. Close transaction file,

13. Open history file,

14. Process user-defined historical transaction initial-ization code, ****

15. Reinitialize a new historical file,

16. Close historical transaction file,

17. Return to menu.

The inserted code:

	Sequence Number	Description	Address
*	4000 - 4499	Master file initial- ization	23200 - 23299
**	4500 - 4999	End of master file code	23300 - 23399
***	5000 - 5499	Transaction file YEC initialization code	23500 - 23599
****	5500 - 5999	Historical file YEC initialization code	23700 - 23799

CHAPTER 12. VALIDATING FILE RELATIONSHIPS

When option 13 on the main PEARL selection menu is selected, the PEARL control file is scanned to determine if there are any logical errors in the way the connections have been specified by the user between the MASTER FILE, the TRANSACTION FILE, and the HISTORICAL TRANSACTION FILE. You may select to have all of the the validation messages printed, or to only have the errors which are detected displayed on the terminal. The following validations occur during processing:

1. If cross validation processing has been selected, the variable name must be of the correct format.

2. When cross validation processing has been selected, the length and the format of the keys must be the same in all cases where they are used.

CHAPTER 13. DATA ELEMENT REPORTS

Menu entry number 9 on the main selection menu may be used to load the program which lists the data element and phrase selection specifications. When this program is loaded, the following menu is displayed to the operator:

```
 0      RETURN TO MAIN MENU
 1      DETAIL LIST OF DATA ELEMENTS BY RECORD/EDIT POSITION
 2      SUMMARY LIST OF DATA ELEMENTS BY VARIABLE NAME
 3      GENERATE REPORTS FOR ALL FILES
 F=nn    TO RESET FILE ID
```

13.1 DETAIL LIST OF DATA ELEMENTS BY RECORD/EDIT POSITION

In the detailed listing, a full report of each data element defined on the current file (as displayed in the header of the screen) will be provided. For those data elements which have phrase selection, the selected phrases will also be listed, or an error message displayed if no phrases have yet been defined. The data record and each control record will be listed on a separate page. At the end of the data element listing, the file definition information is printed. The display-only fields described in Sections 3 and 4 above will contain current information on the data record length, data element count, etc. Also, an estimated length will be displayed for each control record. If any of the control records have an estimated length greater than the data record length, the data elements in the control records should be redistributed, or the data record could be padded out. If this is not done, there is a danger that one or more of the generated programs may be aborted during execution due an attempt to write a record to the file longer than the declared record length.

When this report is generated, a number of logical errors in the construction of the control data may be detected. If this happens, error diagnostic messages will be printed with error codes which can be referenced in the ERROR CODES section of this manual.

13.2 SUMMARY LIST OF DATA ELEMENTS BY VARIABLE NAME

The summary report lists only the variable code name and description along with its type, position in the record, field length, sequence number and edit control. There is one data element per line. This report includes all the data elements which have been defined on all files, from the data record and from all the control records, in alphabetical order by variable code name. This serves as a handy cross-reference list during the development process.

13.3 GENERATE REPORTS FOR ALL FILES

When this option is selected, the detail reports for all files are listed, followed by the summary list. Upon completion of processing, the current file id will be set to the highest file id number which has been defined.

CHAPTER 14. MENU CONTROL DATA REPORTS

When option 10 on the main control menu is selected, the program to list menu control data will be loaded, and you will be provided with the following menu:

```
  0     RETURN TO MAIN MENU
  1     LIST MENU CONTROL INFORMATION
  2     CREATE DEFAULT MENU CONTROL INFORMATION
```

14.1 LIST MENU CONTROL INFORMATION

When this option is selected, the menu control information will be listed. Each submenu which is to be generated will be listed on a separate page.

14.2 CREATE DEFAULT MENU CONTROL INFORMATION

This option allows you to direct the system to create the menu options based on the files you have defined. This function is also provided in the program used to define the menu control information. See Chapter 9 for a description of the processing which is done.

CHAPTER 15. REPORT CONTROL DATA REPORTS

The option to print report definition control is invoked by selecting main menu option number 11. When this program is loaded, the following menu is provided:

```
  0     RETURN TO MAIN MENU
  1     TO LIST REPORT CONTROL DATA FOR CURRENT REPORT
  2     TO SELECT NEW REPORT ID (CURRENT REPORT=00)
  3     TO LIST CONTROL FOR ALL REPORTS
  F=nn   TO RESET FILE ID
```

15.1 TO LIST REPORT CONTROL DATA FOR CURRENT REPORT

This option will create a report of the control data for the current report which has been selected.

See Chapter 10 for a description of the report control information.

15.2 TO SELECT NEW REPORT ID

Use this option to change the report id prior to selecting option 1 to list the control for a specified report.

15.3 TO LIST CONTROL FOR ALL REPORTS

When this option is selected, the control for all reports which have been defined in the system will be listed.

CHAPTER 16. POST/CLOSE COMPUTATIONAL CONTROL REPORTS

When option 12 on the main control menu has been selected, the program to list the computational control data for posting and closing program generation will be loaded and the following menu will be displayed:

```
  0     RETURN TO THE MAIN MENU
  1     LIST POST/CLOSE COMPUTATION CONTROL CODE
  F=nn   TO RESET FILE ID
```

When option 1 is selected, the computational control code which has been entered to control processing during the posting and closing process will be listed.

Refer to Chapter 11 for a description of the points at which the user defined CBASIC code is inserted in the posting and closing programs which are generated.

CHAPTER 17. SYSTEM GENERATION CONTROL MENU

Code generation is coordinated by the generation control program, which is chained to from the main menu program. In turn, it chains to each of the seven code generation programs each of which chain back to the generation control program upon completion.

Generating all of the modules within a system may take from 10 - 15 minutes to several hours. The length of time depends on the number of files and data elements defined in the system.

Generated module files replace existing files by the same name, if any.

When the user selects option 15 on the main menu, the system generation program will be loaded and the following menu will be displayed:

```
 0     RETURN TO MAIN MENU
 1     GENERATE ENTIRE SYSTEM (ALL FILES)
 2     GENERATE ENTIRE SYSTEM (CURRENT FILE ONLY)
 3     GENERATE I/O ROUTINES
 4     GENERATE DISPLAY/EDIT ROUTINES
 5     GENERATE REPORT SUBROUTINES
 6     GENERATE SYSTEM INITIALIZATION AND CONTROL UPDATE PROGRAMS
 7     GENERATE FILE UPDATE/EDIT AND REPORT MAINLINE PROGRAMS
 8     GENERATE MENU SELECTION PROGRAM
 9     GENERATE POSTING AND CLOSING ROUTINES
 10    TO SET UP COMPILE SUBMIT FILE
F=nn TO RESET FILE ID
```

The following is a summary of the processing which occurs for each selection:

17.1 GENERATE ENTIRE SYSTEM (ALL FILES)

When this selection is made, all routines will be generated for all files which have been defined.

17.2 GENERATE ENTIRE SYSTEM (CURRENT FILE ONLY)

If the option is to generate the entire system for the CURRENT FILE ONLY, only the programs to process the file indicated in the display header will be generated. (When this option is selected, generation of the main menu program will not be done. Also, the posting and closing programs will be generated only if the current file has been defined as a transaction file.)

17.3 GENERATE I/O ROUTINES

Within this program, the following three options are available:

17.3.1 Create the Mainline I/O Routine

This routine begins at the subroutine label specified as the base for the I/O routines. It will have entry points to open and close a file, get the next or a specific record, add a record or replace the last record accessed, delete a record, set scan limits (indexed files only), to replace the first file control record, and to enter and edit the record key (indexed files only).

The name of the module created will be the file prefix + I/O subroutine base label. For example, RR10000.BAS.

17.3.2 Create Include Files for Control Records

One module will be created for each control record defined. This file will be used as an inline routine at the point control records are read or written to the file.

The name of each file created will be the file prefix + "CREC" + n, where n is the relative control record number. For example, RRCREC1.BAS, RRCREC2.BAS, etc.

17.3.3 Create the Short I/O Subroutine (Open/Close Only)

One module will be created with the same label as the routines in the main I/O subroutine to open and close the file. This subroutine may be used in those cases where a program is to process only the control records but none of the data records on the file.

The name of the file which is created will be the file prefix + the I/O subroutine base label + "S". For example, RR10000S.BAS.

17.4 GENERATE DISPLAY/EDIT SUBROUTINES

One display and one edit subroutine are generated for the data record, and for each control record which has been defined for each file. The two options available in this program allow for the generation of either all of the display routines, or all of the edit routines for the current file.

17.4.1 Create the Record Edit Subroutines

When this option is used, the routines generated will be the ones used to prompt the operator for the input of data from the console during initial entry of data. These same routines are used for editing the data once it has been stored on the data file.

The names of the routines generated depend upon the edit/display subroutine base, and the control data display edit subroutine base. The following conventions are used:

The name of the subroutine to edit the primary data record will be the file prefix + (display edit subroutine base + 100). For example, if the display edit subroutine base is 30000, the display edit subroutine name would be RR30100.BAS.

The names of the subroutines to edit the control records will be the file prefix + ((control data display edit base X (control record number - 1)) X 1000 + 100. For example, if the control data display edit base were 40000, and three control records had been defined, the routines generated would be RR40100.BAS, RR41100.BAS, and RR42100.BAS.

17.4.2 Generation of the Record Display Subroutines

For each entry/edit routine generated, a corresponding display routine will also be created. The name of each subroutine generated will have a value of 100 less than the edit routine used in conjunction with the display routine generated. Using the same examples as above, a display routine for the main data record would be generated with a name of RR30000.BAS, and for each of the control records, the names would be RR40000.BAS, RR41000.BAS, and RR42000.BAS.

17.5 GENERATE REPORT SUBROUTINES

This program is used to generate the subroutines required for report generation. The user has the option to generate all of the report subroutines associated with the current file, or to generate a specific report subroutine as indicated by the report identification number.

Each report subroutine contains a primary entry point which controls the scanning of the file during report processing. In addition, entry points to space to the top of form, print the report heading, print each record in the report, compute totals, and to perform user specified computations are provided (See chapter 10.) Each routine will be assigned a file name of prqq000.BAS, where qq is the report routine base defined, divided by 1000. If the default base of 50000 is used, the file name would be pr50000.BAS.

The following options are provided in this routine:

17.5.1 Create a Report Subroutine

When this option is selected, the operator will be requested to enter the report identification for the report subroutine to be generated.

17.5.2 Create All Report Subroutines

When this option is selected, all of the report subroutines for the file which is currently being processed will be generated.

17.6 GENERATE SYSTEM INITIALIZATION AND CONTROL UPDATE PROGRAMS

Three options are available in this program. The first option provides for creation of the inline routines which initialize file control parameters. The second option provides for the generation of the mainline and subroutines used to initialize a system. The third option allows the user to generate the control data edit mainline program. (This program is used to edit the data in the file header records.)

17.6.1 Create the File Control (INTL) Data File

Two modules are created when this option is used. The first module created is used inline at the beginning of all program generated to set file control paramaters. These parameters include the name of the file, the disk drive for the file, the length of the records on the file, and the file number. Also included in this module are dimension statements required for data elements which use array validation, and the initial values to be used for entry, editing, and display of these variables.

The name of this module is the file prefix + "INTL" for example, RRINTL.BAS.

The second file created by this option is the inline program code used to set the current version of the program, and to initialize any common variables previously initialized. The name of this module is COM + the system code. For example, COMXXX.BAS. (Note that the system code is different than the file prefix, and that often this module could be used for programs which process several files, but which logically all belong to the same system.)

17.6.2 Create the File Initialization Routines

When this option is used, two routines are generated. The first is the subroutine used to initialize the file, and the second is the mainline program used for file initialization.

The name of the subroutine will be the file prefix + (I/O subroutine base plus 900). For example, if the I/O subroutine base were 10000, and the file prefix were RR, the the name of the generated file would be RR10900.BAS.

The name of the mainline module generated will always be the file prefix + "000". For example, RR000.BAS.

17.6.3 Create the Control Data Mainline Edit Program

This program is used to edit the control records on the file. The mainline program is used in conjunction with the display edit subroutines described in Section 10. The name of the program generated is the file prefix + "000A". For example, the name might be RR000A.BAS.

NOTE:

This program will have a menu entry for each control record defined to the system.

17.7 GENERATE FILE UPDATE/EDIT AND REPORT MAINLINE PROGRAMS

This program is used to generate the update and edit mainline, the file reorganization mainline and subroutines (for indexed files only), and the mainline program to control report generation processing.

17.7.1 Create the File Update Mainline Program

The program module generated allows the user to add, edit, or delete records from the file.

The name of the module generated is the file prefix + "100". For example, RR100.BAS.

17.7.2 Create the File Reorganization Processors

The file reorganization mainline, and the control subroutine are both generated by this option.

The name of the file reorganization mainline is the file prefix + "000R". For example, RR000R.BAS.

The name of the file reorganization subroutine is the file prefix + 800 greater than the I/O subroutine base. For example, if the I/O subroutine base were 10000, the name of the generated module would be RR10800.BAS.

17.7.3 Create the Mainline Report Writer Program

The program generated here is used in conjunction with the report subroutines which were generated (see Section 17.6). A single report generation control program is generated for each file processed and is assigned a name with a prefix of the unique file id, and a suffix of the file number. For example, if the unique file is is RR and the file number is 10, the program name would be RRP0010.BAS.

A menu will be provided in this program such that the operator can select any of the reports which have been defined for the specified file.

17.8 GENERATE MENU SELECTION PROGRAM

This program has a single option, and that is to generate the main selection menu program. The name of the program module which is generated will be the same that was specified when the system control information was placed on the PEARL control file. For example, if the main menu program name specified was DOITALL, then the program module which is created will be DOITALL.BAS.

Once the system is compiled, it could then be executed by issuing the command RUN DOITALL.

17.9 GENERATING POSTING AND CLOSING ROUTINES

This program provides three options to the operator. The first is to generate the routines to post data from the transaction file to the master file. The second is to generate the period-end closing routines. The third is to generate the year-end closing routines.

17.9.1 To Create Posting Routines

The selection of this option will cause the posting main line program, and the associated subroutine to be generated. The mainline program is a value of 200 prefixed by the unique file id for the transaction file, and the subroutine is a value of 20000 prefixed by the unique file id for the transaction file. For example, if the unique file id is IT, the two routines generated would be IT200.BAS and IT20000.BAS.

17.9.2 To Create Period-End Closing Routines

The selection of this option will cause the period-end closing main line program, and the associated subroutine to be generated. These routines will be assigned names of 300 and 21000 respectively prefixed by the unique file id for the transaction file. For example, if the unique file id is IT, the two routines generated would be IT300.BAS and IT21000.BAS.

17.9.3 To Create Year-End Closing Routines

The selection of this option will cause the year-end closing main line program and the associated subroutine to be generated. These routines will be assigned names of 400 and 23000 respectively prefixed by the unique file id for the transaction file. For example, if the unique file id is IT, the two routines generated would be IT400.BAS and IT23000.BAS.

17.10 TO SET UP COMPILE SUBMIT FILE

This option allows the operator to create a SUBMIT control file which can be used to compile the generated system. The options which are avaiable through this feature, and a discussion of its use are provided in the following chapter.

CHAPTER 18. CREATING SUBMIT CONTROL TO COMPILE GENTERATED SYSTEM.

18.1 SUMMARY OF THE INTENT OF THE SUBMIT FILE PROCESS

Each generated PEARL Level 3 system must be compiled before it can be executed. The creation of a SUBMIT file will make this process easier and more automatic.

PEARL Level 3 generated systems are modular. Each system consists of a minimum of five interdependent main programs. Each main program must be compiled independently converting the .BAS source file to an .INT (intermediate) file which can be executed. The SUBMIT file which is created by PEARL contains the CP/M operator commands to compile the programs in the generated system.

In order to create a submit file, you must select option 10 of the SYSTEM GENERATION CONTROL MENU. If the default control information is used, the SUBMIT file generated will be placed on disk drive [B] (generated source code output disk unit) and will have a name of [COMPILE]. If CP/M is the operating system which is used, a file type of .SUB should be used. (However, some operating systems which are not standard CP/M operating systems, but are derivatives of CP/M, may require a different file type. Refer to your operator's manual if the SUBMIT command does not work as specified below.)

The following examples demonstrate the use of the option to create submit control files.

18.2 EXAMPLE

A system has been generated that processes a random access file with a file prefix of [XX] designated. The main menu program was named [EXAMPLE].

The generated SUBMIT FILE will have the name of [COMPILE.SUB] and will contain the following commands:

THE COMMAND	PROGRAM DESCRIPTION
CBAS2 EXAMPLE	MAIN MENU PROGRAM
CBAS2 XX000	FILE INITIALIZATION
CBAS2 XX000A	FILE CONTROL RECORD(S) UPDATE
CBAS2 XX100	UPDATE DELETE DATA RECORDS
CBAS2 XXP0001	REPORT ON DATA FILE RECORDS

When the operator has completed the generation process of the system source code and the SUBMIT file, he would then exit to CP/M and enter the command:

```
SUBMIT B:COMPILE
```

This will cause all of the generated programs to be compiled.

18.3 MODIFYING THE SUBMIT CONTROL PARAMETERS

The compile process involves processing of two file types. These are the input files which are of the type [BAS] and the output files which are of the type [INT]. At the time the generated system is acutally executed, only the .INT files are required.

At the time the SUBMIT file is created, the operator may specify which drives the INT files will be placed on, which drives the BAS files will be obtained from, and what the name of the SUBMIT control file will be. The following menu is provided when option 10 on the SYSTEM GENERATION MENU is selected. (The defaults are based on the control information placed in the PEARL control file system control data.)

```
          COMPILE/SUBMIT SETUP CONTROL

          1. MAIN PROGRAM SOURCE DISK         B
          2. INTERMEDIATE FILE OUTPUT DISK    B
          3. CBASIC COMPILE DIRECTIVES        $$B
          4. SUBMIT FILE TYPE                 SUB
          6. FILE COPY COMMAND FILE NAME      PIP
          7. FILE COPY DIRECTIVE              [V]
          8. FILE COPY DIRECTIVE POSITION     PIP X.BAS=Y.BAS[V]
      ENTER LINE TO EDIT OR RETURN TO CREATE  SUBMIT FILE
```

The default parameters tell CP/M and the CBASIC compiler that:

1. THE MAIN PROGRAM SOURCE FILES [BAS] ARE TO COMPILED ON DRIVE [B].

2. THE COMPILE OUTPUT FILES [INT] ARE TO BE PLACED ON DRIVE [B].

3. DISPLAY COMPILE DIAGNOSTICS ONLY DURING COMPILE TIME. (The $$B is set to $B by the SUBMIT processor so that the CBASIC compiler receives only one "$" in the compiler directive.)

4. THE SUBMIT FILE TYPE IS [SUB]. THIS IS THE CP/M FILE TYPE FOR SUBMIT FILES. (The users of non-standard CP/M systems must consult their operating system manuals for instructions on naming conventions of SUBMIT type files.)

5. FILE COPY COMMAND FILE NAME.

   "PIP" is the default for the file copy program. If the operator selects the main program source disk as A, then the main programs will be copied, one at a time, from drive B to drive A before compilation.

If the file copy program name used is other than "PIP", then it may be changed here.

6. FILE COPY DIRECTIVE.

   [V] is the directive to verify copies made during the "PIP" process. The operator may change the file copy directive if it is different.

7. FILE COPY DIRECTIVE POSITION.

   The file copy directive defaults to the end of the copy statement. Other systems may require its placement at other positions.

18.4 MODIFYING THE OUTPUT DRIVE FOR THE INT FILES

An operator is generating a system which has two 8-inch double density floppy disk drives. The system drive is designated as [A], and the default data drive as [B]. The operator has determined that there is not enough space on drive B for the INT output files. Therefore, line 2 on the selection menu is edited to indicate that the INT files should be placed on drive A during the compile step.

18.5 MOVING THE MAIN PROGRAM .BAS FILES

In some cases, the programmer may want to move the main program .BAS files from the generated output diskette to another drive. (This will often be the case if the combined size of the source files exceeds two diskettes.) In this case, main program [BAS] files may be maintained on a series of diskettes, each of which are placed on drive [A] during compiles, and a single subroutine diskette which is always placed on B during compiles. In this case, he should change control option [1] to [A]. This will cause the main program source [BAS] files to be moved to drive [A] before they are compiled.

18.6 CHANGING THE COMPILER DIRECTIVES

The default compiler directives are set to give no listing to the operator, but to list the line number on which any errors occur on the terminal. The following are some examples of optional directives which can be supplied (refer to your CBASIC manual for additional options.)

$$BF	List errors on the console, list program on the printer.
$$F	Provide a full listing on both the printer and the console.
blank	List the output on the console only.

NOTES:

1. When a double dollar sign is provided as control within a SUBMIT control file, this control is resolved to a single dollar sign when the submit file is executed by CP/M. Thus, when a $$B is provided as the compile directive, it is supplied to the CBASIC compiler in the form $B.

2. SUBMIT control data may be supplied at the time the SUBMIT file is used by the operator. In order to set this control information up, for example, a $1 may be entered as the compiler directive. This indicates that the $1 will be supplied by the operator at the time the submit control file is executed as the first positional parameter in the command. This command would be entered by the operator as follows:

   ```
   SUBMIT COMPILE $B
   ```

   All occurrences of $1 will be replaced by a $B when the COMPILE.SUB file is executed.

CHAPTER 19. MOVING A GENERATED SYSTEM TO PRODUCTION STATUS.

When development has been completed, and the system is moved to a production status, it is important that standard procedures be followed in order to maintain the integrity of the program source and the programs themselves. Experience has shown that if standard procedures are not followed, many hours of programmer time and effort can be spent tracking down bugs which are introduced into a system after development was completed. Also may hours of time can be spent trying to determine if the programs which are being executed by an end user correspond to the source files which are maintained on separate diskettes, sometimes at another location.

The following are suggested procedures which may be used to mimimize lost production time attempting to track down these kinds of problems.

19.1 USE LEVEL AND DATE REMARK

The first line of each source file generated by PEARL has a modification level and a date indicated. Each time a module is updated, increase the level number by one, and place the new date in this record. This will allow someone who is working with the system to quickly know the last time a module was updated.

19.2 ASSIGN AND UPDATE VERSION NUMBERS

The COMxxx.BAS file (where xxx is the system identification) contains the current version number of the system. This version number is displayed on the first screen displayed when a system is initially loaded, and is also displayed at the top of each screen when a new program is loaded.

The version number which is displayed will be the version number in the COMxxx.BAS file the last time the program was compiled.

If this version number is incremented by one each time a new version of the system is distributed to an end user, and a listing or a log of the changes which were made to each version is maintained with the source diskettes, it will aid both the end user and the programmer to match the programs which are being used to the source programs.

19.3 SOURCE PROGRAM MAINTENANCE

In most cases, the .BAS source files will be maintained on separate diskettes and stored in a separate location from working system diskettes. The source programs as well as the working system diskette should be clearly labeled as to the current version, and date the version was completed.

19.4 MAINTAINING A LOG OF CHANGES

It is advisable in most cases to maintain a written log of all changes which have been made to a system once it is placed in production status and distributed to an end user. Such a log is also useful in recording requests for modifications to a system so that a number of requests can be incorporated and tested at one time.

19.5 DISKETTE ORGANIZATION

In Chapter 3, it is suggested that you set up eight diskettes for PEARL programs, data files and the end user system. The following is a discussion of the steps which can be followed to use these eight diskettes. These steps go from the system defintion step through the generation, compile, and setting up a system in production. This discussion assumes that two disk drives are being used, and that A is the SYSTEM DISK and B is the DATA DISK.

<u>STEP 1. Define a System</u>

A: PEARL SYSTEM DISK	Contains PEARL system programs.
B: CONTROL DATA DISK	To begin with, this is a blank diskette. After system definition, this disk contains PEARL.CCT and PEARL.NDX

<u>STEP 2. Generate a System</u>

A: PEARL GENERATION DISK	Contains PEARL system programs.
B: CONTROL DATA DISK	To begin with, this disk contains PEARL.CCT and PEARL.NDX. After system generation, this disk contains the generated .BAS files.

<u>STEP 3. Setup the APPLICATION SYSTEM MAIN PROGRAMS Disk</u>

A: PEARL COMPILE MASTER	This contains CP/M system programs, the CBASIC compiler, and the common .BAS source files provided with PEARL.
B: APPLICATION SYSTEM MAIN PROGRAMS	This disk is a duplicate copy of the PEARL COMPILE MASTER disk.

Once the PEARL COMPILE MASTER disk has been setup, you can make a copy of it labeled (APPLICATION SYSTEM MAIN PROGRAMS). Doing this will eliminate the need to have to clean it up (delete unwanted source files) each time a new system is generated.

STEP 5. Setup the GENERATED System SUBROUTINE DISK

A: CONTROL DATA DISK	Contains PEARL.xxx control files and the generated .BAS files. (It is convenient to also have a CP/M operating system and the PIP.COM program on this diskette.)
B: GENERATED SYSTEM SUBROUTINE DISK	A blank diskette to begin with. At this point, the .BAS files, and the COMPILE.SUB files should be copied from the CONTROL DATA DISK to this disk.

While not always necessary, it is often convenient to maintain all of the .BAS files which have been generated on a diskette separate from the CONTROL DATA DISK. This will be necessary in those cases where the PEARL.xxx files are extremely large and where the generated system is quite large.

STEP 6. Compile the Generated System

A: APPLICATION MAIN PROGRAMS	Prior to compile processing, this disk contains the common subroutines, the compiler, and CP/M system programs. After compile submit processing, the .INT files off the APPLICATION SYSTEM MAIN PROGRAMS disk should be pipped over onto this disk. The mainline .BAS files may also be pipped over onto this disk.
B: GENERATED SYSTEM SUBROUTINES	This disk contains the generated system .BAS files.

On large systems it may be expedient to distribute the source programs from the generated system between two diskettes. A good procedure to use here is to place the .BAS files for the mainline modules on the main programs disk, and the subroutines only on the subroutine disk. If this is done, it is a wise idea to delete the mainline modules from the generated system subroutine disk so there are no duplicate modules on separate diskettes.

STEP 7. Placing the Final System in Production

A: APPLICATION SYSTEM DISK	This disk contains the RUN.COM and CP/M operating system programs. After processing, this disk also contains the .INT files from the generated system.
B: APPLICATION SYSTEM MAIN PROGAMS	This disk contains the mainline source programs, common subroutines, and system programs. During this step, the .INT files are moved to the APPLICATION SYSTEM DISK.

In this final step, the .INT files are copied from the main programs diskette to the final APPLICATION SYSTEM DISK. This step may be required a number of times if the size of the system is so large that all of the .INT files cannot fit on the APPLICATION SYSTEM MAIN PROGRAMS disk.

In some systems, multiple main programs diskettes and multiple APPLICATION SYSTEM DISKETTES may be required, depending upon the size of the diskettes used and the size of the generated system.

APPENDIX A

DATA ENTRY AND EDITING FEATURES

The data entry and editing routines provided in PEARL generated programs are designed for ease of use by the operator and to provide a maximum amount of editing at the time of data entry.

The majority of the entry and editing functions used for entry of control data by PEARL were generated by the PEARL program generator programs. Therefore, the general description of the data entry and editing routines apply both to the PEARL programs and the application systems generated by PEARL.

1. TO DUPLICATE A LINE

Each time a prompt is provided for the entry of data, the data most recently entered in that field will be displayed enclosed in brackets immediately following the prompt. If you wish to duplicate the field, simply depress the return key without entering any data.

2. BACKING UP A LINE

a. During Entry.

If you wish to back up a line to replace incorrectly entered data, enter a < in place of the data which would normally be entered. If you wish to backup several lines, enter <n where n is the number of lines to back up. There are some "invisible" lines which may make it necessary for you to enter a larger number than supposed. For instance, entering a "<3" might only take you back two lines. Simply enter a "<" to get to the appropriate line.

b. Upon Completion of a Menu.

Upon completion of the system prompts and user responses for a particular option, PEARL will display in full detail what a you have entered for a particular option. A phrase is be provided that will allow the you to edit any incorrectly entered fields: ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE. When you choose the field number to be edited, it will be displayed on the screen with the system and your original response on the same line. You can simply type in the desired data. For instance, a typographical error has been made and you wishes to correct it:

```
MENU DISPLAYED

1: SYSTEM DESCRIPTION               CUSTOMER CONNTACT FILE
2: MAIN MENU PROGRAM NAME           CONTACT
3: SYSTEM CODE                      BCF
4: NAME OF END USER                 USER'S NAME
etc.
ENTER FIELD NUMBER TO EDIT OR RETURN TO TERMINATE
```

User notices CONTACT has been misspelled.

User Enters:

    1

System Responds:

```
SYSTEM DESCRIPTION [CUSTOMER CONNTACT FILE]
```

User Enters:

    CUSTOMER CONTACT FILE

System Responds:

The menu will be redisplayed with the corrected entry.

3. ESCAPE

Depending upon the point at which the ESCAPE key is used, one of the following will occur:

a. ADD MODE

When the ESCAPE key is used during addition of records to a file, add processing will be terminated, and control will return to the last menu displayed. The current record being processed will not be added to the file.

b. EDIT MODE

If an ESCAPE key is used in place of data, or in place of the line number to be edited, the current record being processed will not be replaced on the file. You will be prompted to enter the key of the next record to be processed.

c. EDIT OPTIONS

When an edit screen is displayed, each line will indicate the selected edit option for the data elements appearing on that line. An example of each option available is as follows:

```
 1: Name                         John Doe

*1: Name                         John Doe

 *: Name                         John Doe
```

In the first case, the name field may be edited by entering a 1 when the line number to be edited is requested. The edit may occur at the time a record is being added or at the time a record is being edited.

In the second case, the field may be edited only when a record is being added to the file. If an attempt is made to modify the field when a record is being edited, it will be ignored by the editor.

In the third case, no edit line number is provided. The data element description and value associated with the data element is provided only as information to the operator. (This option is provided for data elements which can be modified only under program control.)

APPENDIX B

SYSTEM CONFIGURATION CONTROL DATA

The SYSTEM CONFIGURATION CONTROL DATA is stored in a data set named **SYSTEM.DDD** which will always be expected by the PEARL main menu program or the main menu programs of generated systems. Upon initial entry to a main menu program, an attempt will be made to locate this file. If the file is not located, then you may select to enter the information contained in the file, or to specify a drive other than drive A as the default system drive.

The following is a summary of how this file is created and how the control information on the file is used by PEARL and the PEARL-generated programs:

1. SYSTEM.DDD

If the SYSTEM.DDD file cannot be located, the following message will be displayed:

```
SYSTEM CONFIGURATION DATA COULD NOT BE LOCATED
Enter an ESCAPE to create a new file on DRIVE A:, or,
ENTER DEFAULT SYSTEM DRIVE [A]
```

If an ESCAPE is entered, followed by a RETURN, then you will be prompted to enter the system configuration data as described below. You may optionally enter the drive which will be the default system disk. If the configuration data is located on the specified drive, it will be loaded and processing will continue. If it is not located, the operator may then enter an ESCAPE to create the configuration file on a drive other than Drive A.

2. CONFIGURATION CONTROL FILE DATA FIELDS

TERMINAL TYPE (SCREEN CLEAR)

One of the entries specified may be selected. If the terminal is not in the list, zero may be entered. You will then be requested to enter the string to clear the screen on the terminal being used.

NOTE:

In the event none of the terminals provided in the list are being used, you may select option zero to enter a string of characters to be used as the screen clear character string. If the values needed for the terminal being used are not available from the keyboard, HEX values may be entered by preceeding the string with the prefix "=H" followed by hex values. For example, the string "=H1B1A0000" would provide a screen clear string of 1B (decimal 27), 1A (decimal 26) followed by two null characters.

The default terminal list may be extended by the programmer to include additional terminals by editing the CINTL.BAS source file provided in the distribution source. However, these modifications will apply to the generated programs only.

FORMS CONTROL OPTION

If a zero is specified, the Form Feed (FF or CHR$(12)) will be used to advance to the top of form. If other than zero is specified, the program assumes this is the number of lines in each page on the printer device being used.

DISKETTE CAPACITY IN K BYTES

This is the number of K bytes of storage on the DEFAULT DATA DISK DRIVE (see description below).

MESSAGE LEVEL

This defines the level of diagnostic messages which will be created during the execution of PEARL and the PEARL-generated programs.

REPORT DEPTH IN LINES

This is the number of lines to be printed on a page of a report before advancing to top of form.

INSTALLATION NAME

The installation name may be used on PEARL-generated reports. It is stored in the CC.INSTALL$ variable in common, and is available to all programs within a PEARL-generated system.

DEFAULT SYSTEM DISK DRIVE

This is a single-character disk drive identification. PEARL, as well as all PEARL-generated programs, assume that all programs will be found on this disk unit. In addition, this is the drive on which the system configuration data file will be placed (SYSTEM.DDD).

DEFAULT DATA DISK DRIVE

This is the disk drive on which the PEARL control files (PEARL3.CCT and PEARL3.NDX) will be placed. In addition, the user may specify during system definition that files processed by PEARL-generated programs will be placed on this disk unit.

DATE FORMAT

This option allows you to specify MM/DD/YY or DD/MM/YY as the display format for the date. (Because dates are stored internally as a floating point number in the form YYMMDD, this option may be changed at any time with no effect on the dates in PEARL-generated files.)

PASSWORD

If password protection is required, you should enter the password to be used. If a password is used, you will be required to enter the password each time PEARL is initiated, a generated main menu program is loaded, or a request is made to edit the system configuration data.

3. VARIABLE NAMES

The information entered in the system configuration control data is available to all PEARL generated programs in the common area in the following variables:

TERMINAL.TYPE

CC.SCREEN.CLEAR$	The string used to clear the terminal screen.
CC.TERM.TYPE%	The number entered to indicate the type of terminal.
CC.CON.WIDTH	The width of the terminal screen.
CC.CON.LINES	The number of lines on the terminal screen.

FORMS CONTROL OPTION

CC.FORMS.CONTROL%	A zero to indicate the form feed character (decimal 12) should be used to space to top of form, or the number of lines on each page from perforation to perforation.

DISKETTE CAPACITY IN K BYTES

CC.DISK.SIZE%	The number as entered by the operator.

MESSAGE LEVEL

CC.MEGLEVEL%	The number as entered by the operator.

REPORT DEPTH IN LINES

CC.PAGE.LEN%	The number as entered by the operator.

INSTALLATION NAME

CC.INSTALL$	The name as entered by the operator.

DEFAULT SYSTEM DISK DRIVE

CC.SYS.DISK$	A single digit alpha character indicating the drive on which the SYSTEM.DDD file and all of the programs will be placed during execution of PEARL or a generated system.

DEFAULT DATA DISK DRIVE

CC.DATA.DISK$	A single digit alpha character indicating the default data drive on which the PEARL control file will be placed, or all data files to be placed on the default data drive.

DATE FORMAT

CC.DATE.FORMAT%	A value of 0 or 1 indicates the manner in which the date is to be formatted for display. (All dates are stored internally in the format yymmdd in a floating point variable. This option is used only to control output formatting of the date when performed by the FN.DATE$ function.)

APPENDIX C

UTILITY SUBROUTINES

Several utility subroutine modules are provided with the PEARL program package. These routines are used by the generated programs through the use of GOSUB program statements to provide data input and editing functions, load system configuration data from a control file, make copies of data and index files, etc.

The following is a summary of each of the included subroutine files, the paramaters needed, if any, and the values returned by each routine.

1. COM01E.BAS -- COMMON VARIABLE ALLOCATION

This file contains the common allocation inserted in each PEARL-generated program prior to the first line of executable code. The data in the common area (date, flag and configuration data) is used to pass data from one program to another. (For example, the date entered in the main menu program is available to all programs in the CC.DATE and CC.DATE$ variables). Also, contained in this file are non-common initialized variables and function definitions.

The following is a list of each of the variables in the common area and a summary of what each variable contains:

CC.MEMORY	The least amount of free memory available prior to closing any file.
CC.SERIAL.NO%	Not used.
ENTRY.FLAG%	This variable is set to one after the system configuration control data has been loaded to eliminate the need to load that file each time an entry to the main menu program is executed.
CONFIG.FLAG%	This variable is set to 1 after the configuration file has been created and initialized.
CC.MSGLEVEL%	This variable controls the level of diagnostic messages generated during execution of a PEARL generated program (See Chapter 4, page 21, Step 1).
CC.DATE	The current system date in the format yymmdd.
CC.DATE$	The current system date in the format mm/dd/yy.

CC.DISK.SIZE%	The number of bytes of storage / 1000 available on each diskette (See Appendix B.)
CC.FORMS.CONTROL%	A zero if the standard form feed control character should be used, or the number of lines from the top of a page to the top of the next page (See Appendix B.)
CC.SCREEN.CLEAR$	The character string required to clear the screen on the video terminal being used (See Appendix B).
CC.TERM.TYPE%	The terminal type (See Appendix B.)
CC.CON.WIDTH	The number of characters to fill one line on the console screen.
CC.CON.LINES	The number of lines to fill the entire screen.
CC.PAGE.LEN%	The number of lines of output to appear on a page prior to spacing to top of form (See Appendix B.)
CC.FILE.DRIVE$	Not used.
CC.SYS.DISK$	The disk drive which has been defined as the default system drive. (See Appendix B.)
CC.DATA.DISK$	The default data disk drive. (See Appendix B.)
CC.CHAIN.TO$	The program to which chaining is to occur, or if the chain has occurred, the program which is currently executing.
CC.CHAIN.FROM$	The program from which the chain has occurred.
CC.CHAIN.LIST$	A list of programs which will be chained to one after the other as controlled by the 90400 routine.
CC.MAIN.PROG$	The main menu program for the system.
CC.SYSTEM$	The system identification.
CC.STATUS$	Not used.
CC.COMPANY.NAME$	Not used.
CC.MSG$	A diagnostic or informational message pending display on CRT.

CC.DATA.FORMAT%	A control code to indicate the output format of dates when formatted by the FN.DATE$ function. (See Appendix B.)
CC.REPORT%	This variable is used to implement and verify the sort process. When a sort is requested for a report, CC.REPORT% will be equal to the report menu number requested. (See Appendix I for detail.)
CC.SORT%	This variable is used to implement and verify the sort process. It will be equal to the sort control menu number. (See Appendix I for detail.)
CC.SORT.DRIVE$	This variable designates the drive on which sorted files are to be found by the report writer. This variable is updated at sort time by the sort control record. It will be equal to the sort output drive as defined on the sort control record.
CC.REV1,2,3 CC.REV%1,2,3 CC.REV$1,2,3	These variables (three strings, three integers, three real numbers) are reserved in the common area for use during program modification to pass variables from one program to another.

Several predefined functions are used by the PEARL programs for string manipulation. The descriptions for these functions and the parameter requirements are as follows:

FN.CNTR$(A$,X)

The value returned by this function is a string which has the value in A$, centered with leading and trailing blanks, for a total length specified in X.

The variable SPACE$$ has been set to a 60 character blank string and is used to pad the string with the leading and trailing blanks. X may be defined as CC.CON.WIDTH.

FN.DATE$(DATE)

The value returned by this function is a date in the format mm/dd/yy or dd/mm/yy (depending on the value of CC.DATE.FORMAT%.) The input variable is a date in the format yymmdd. All dates are stored internally in a yymmdd format for greater than or less than comparisons.

The variable (DATE) is converted to the string D$. If the length of D$ is less than 6, the left side of D$ is packed with zeros until the length of D$ equals 6. Then DATE$ is built by pulling first the month section of D$ followed by the day section and year section and separating them with slashes.

2. FNFILEE.BAS -- (PREDEFINED FUNCTIONS)

The functions contained in this file are used in conjunction with the file I/O routines generated by the system generator. There definitions and parameter requirements are as follows:

<u>FN.RSPA(STRING$,SPACES$)</u>

The value returned by this function is the number of nonblank characters found in the variable STRING$. The SPACES$ variable must previously have been set to spaces or blanks. STRING$ is first tested for its length. It is then tested to see if there are any trailing blanks. If not, then STRING$ is returned. If so, M% is equal to the string length and K% is equal to the string length divided by 2. The string is tested to find out if half of it is blanks. If not, the string is tested in a loop until the length of the string without the blanks is found and STRING$ is then returned.

<u>FN.CNUM$(SSSS,VALUE,NAME$)</u>

The value returned by this function is a numeric string into which leading zeros have been inserted. The length of the string is determined by the SSSS variable, while VALUE provides the numeric value. In the event the string, when converted, exceeds the length specified by SSSS, the value returned is all character zeros, and a message is output indicating that the length of the variable specified in NAME$ has been set to zero due to excessive length of the variable.

<u>FN.CSTR$(SSSS,V$,SPACES$)</u>

The value returned by this function is the string provided in V$ normalized to length specified in SSSS. If the original length of the string is greater than SSSS, then V$ will be truncated on the right. If the original length of V$ is less than length, then the output will be right-padded with blanks (or the value found in SPACES$ which must set prior to use of the function.)

3. CENTRYE.BAS -- PROGRAM ENTRY AND STATUS DISPLAY

This routine is used on entry to all PEARL generated programs to display the name of the system, the program specified in the CC.CHAIN.TO$ variable, the system status, the function of the program being executed, and any outstanding message scheduled for display to the operator.

An initialize command is also issued to refresh the memory map associated with disk I/O. (The initialize command can be suppressed by setting the CENTRYE.FLAG variable to 1 prior to entry into this routine.)

No labels are associated with this routine because it is used in-line rather than as a subroutine.

4. CHAIN.BAS -- %CHAIN CBASIC COMPILER DIRECTIVES

The CHAIN.BAS file holds the CBASIC2 compile directive that is required for chained programs. This directive sets the size of the main program's CONSTANT, CODE, DATA, and VARIABLE areas. The values provided with the PEARL package are set for aplications to be run on 48K systems. Applications needing larger systems will require changes in the CHAIN.BAS file (consult CBASIC2 Manual for directions). The %CHAIN directive was set up as a separate file to be included into main programs in order to facilitate changes to fit the particular application generated.

5. C24000E.BAS -- POST/CLOSE REPORT WRITER

This routine is included in the posting and closing routines to list out on the printer or terminal a list of transaction records posted to the master file. Example:

```
TRANSACTION (256) POSTED TO MASTER FILE RECORD (SMITH)
```

All messages printed are initialized in the calling routines. The following is a list of the variables that are printed by this routine:

<u>Line 24080</u>

CC.MS$(1) CC.MS$(2) CC.MS$(3)	These variables are printed at the conclusion of a print process. These variables are printed centered on the page.

<u>Line 24100</u>

CC.MS$(4) CC.MS$(5)	These variables are printed at the top of each form (heading). They are both printed consecutively and centered on the page.
CC.PAGE%	Page number at tab position 65.
CC.COMPANY.NAME$	Centered at top of page.
CC.DATE$	Current date.

Line 24800

CC.MS$(6)	
CC.MS$(7)	
CC.MS$(8)	
CC.MS$(9)	
CC.MS$(10)	These variables are detail variables for the report. They are printed consecutively starting at tab position 10. If the composite length is greater than 70 characters, then they will be split into two lines with the second line starting at tab position 20. If the composite is greater than 130 characters in length, three lines will be printed.

NOTE:

Calling address 24000 with a GOSUB will dimension the subscripted variables the first time. Subsequent calls will set all subscripted variables to nulls.

6. C82000E.BAS -- CONSOLE INPUT ROUTINES

This include file contains a number of entry points used to input data from the keyboard, and to edit or validate that data. Unless otherwise specified, the data provided on return is:

V$	String input.	
VF	Floating point input.	
V	Integer input, truncated beyond decimal.	
V%	Integer input.	
RCODE	0	Normal input processing completed.
	-1	Escape or line backup character was first character input.
	+1	Integer overflow, (integer input is always provided in both V and V%)
CFLAG%	0	Null string input
	>1	Length of string input
CC.BACKUP	When a "<" character has been entered as the first line of the input, a one, or if the "<" character was followed by a number, that number.	

The following entry points are provided:

82005	Input character must either be a "Y", "N", escape, backup control, or return. All other input is ignored.
82006	Input character must either be a "Y" or "N".

82010 Input character must be a single alpha character, escape, backup control character, or return. No more than a single alpha character will ever be returned in V$.

82075 A date in internal storage format is provided in V on entry. (Format = yymmdd.) On return, the date will be returned in MONTH%, DAY%, and YEAR%. (See the description of FN.DATE$ in COM01E.BAS for more information).

82110 A single character is input into V$. Any character is allowed. If an escape is input, RCODE will be returned as a -1. If no character is available for input, V$ will be returned as a null and CFLAG% as a 0.

82170 A string of characters in input and returned in V$. All double quotes and line feeds are stripped out of the input string prior to return to the calling routine.

This routine sets the CFLAG% variable equal to the length of the input string, RCODE = to -1 if the first character in the input is an escape or a backup control character, and CC.BACKUP equal to the number immediately following the backup control character, if one exists.

82174 Returns a date in V. The date will be in the format yymmdd, and will have been validated as a valid date. The input string may contain any nonnumeric delimiter between the month/day and day/year fields. For example, the following input formats are valid:

```
mm/dd/yy
mmddyy
mm dd yy
```

This routine forces input of a date, and will not return to the calling routine until a valid date has been provided. That is, this routine will never return a null string.

82175 This routine is the same as the 82174 routine for date input and validation processing, except that the escape, backup control character, or a return with no data, will be processed and control indicators set appropriately.

82180 This routine will accept a floating point input from the console and return it in VF.

82182 This routine will accept a floating point input from the console and return it in VF. The variable is rounded to the nearest hundredth prior to return. (For example, an input of 123.456 will be rounded to 123.46.)

82185 This routine is used to input an integer value. All positions to the right of the decimal point are truncated. The value is returned in both V% and V. When the data input exceeds the maximum allowed for an integer value (32676), then RCODE is set to a value of 1 prior to return.

7. C82400A.BAS -- CREATE QSORT CONTROL FILE

This file is used by the sort control program (XSORTX, see Appendix I). Its purpose is to provide QSORT with the information needed for a specific sort process. This provision is made through the creation of a file containing the required information and of type (.SRT). The variables needed are:

```
Inputs:
  SRT.CTL.FILE$      Control file name; may include
      drive id.  File type of ".SRT" will be appended
      by the subroutine to form the actual file name.
      Example:   "B:BLAHBLAH"
  SRT.IN.D$            Input file drive.  Null string
      for currently logged drive; else one character
      for range "A" to "D".
  SRT.IN.N$            Input file name (1-8 chars).
  SRT.IN.T$            Input file type (0-3 chars).
  SRT.OUT.D$           Output file type (see remarks
      above for input file drive).
  SRT.OUT.N$           Output file name (1-8 chars).
  SRT.OUT.T$           Output file type (0-3 chars).
  SRT.WORK.DRIVE$      Work file drive.  Same remarks
      as above.
  SRT.OUT.BACKUP$      Output backup flag.  If "Y", then
      if there is already a file with the same name
      as the output file name, rename it to have a file
      type of ".BAK" before creating the new output
      file.
  SRT.CON.DISPLAY$     Console display flag.  If "Y",
      then displays sort statistics on the console.
  SRT.CHANGE.DISK$     Disk change flag.  If "Y" then
      QSORT will pause just prior to creating the sorted
      output file while the operator inserts a new
      disk in the output file drive.
  SRT.REC.LEN%         Length in bytes of the records
      on the input file.
  SRT.KEY.CNT%         Number of keys to sort on (1-5).
  SRT.CTL%(3,5)        Sort control data.
      SRT.CTL%(0,n)    Starting position of key n.
      SRT.CTL%(1,n)    Length of the key.
      SRT.CTL%(2,n)    0 for ascending sort, 1 for
                       descending.
```

```
    SRT.CTL%(3,n)  0 for binary numeric order (each
                   byte an unsigned integer from
                   0-255), 1 for alphanumeric order
                   (lowercase treated same a upper
                   case for ordering purposes).
    Where n ranges from 1 to SRT.KEY.CNT%
```

8. C82500A.BAS -- CREATE SUBMIT ($$$.SUB) FILE

This file is used by the sort control program (XSORT, see Appendix I) to create a submit control file ($$$.SUB) that will be executed upon the first exit to the operating system.

CROMENCO users should change Line 47

from:

```
  CREATE SUB.CTL.DRIVE$+"$$$.SUB" RECL 128 AS 20
```

to:

```
  CREATE SUB.CTL.DRIVE$+"SUB.$$$" RECL 128 AS 20
```

The variables required are:

```
Subroutine to generate a SUBMIT file.
Input:
   SUB.CTL.DRIVE$    Drive to put the $$$.SUB file on
          (null, A:, B:, C:, or D:)
   SUB.CTL.COUNT%    Number of CP/M commands.
   SUB.CTL$(SUB.CTL.COUNT%-1) CP/M commands, to be
          executed in order of ascending subscript
          starting at 0).
```

```
Example:
      SUB.CTL.DRIVE$="A:"
      SUB.CTL.COUNT%=3
      SUB.CTL$(0)="ERA B:*.SAC"
      SUB.CTL$(1)="QSORT "GARBUJ"
      SUB.CTL$(2)="RUN REGEN700"
      GOSUB 82500
      STOP
```

Upon execution of the STOP statement, the ERA, QSORT and RUN commands will be executed in that order.

If SUB.CTL.DRIVE$ is set to other than "A", the "$$$.SUB" file will be created but will not be executed until the disk containing it is inserted into drive A.

The commands in SUB.CTL$ are written to file "$$$.SUB" as 128-byte fixed-length records in the reverse order of execution, so that the first command desired is the last record. Whenever CP/M is ready for a command (such as upon cold boot, or upon finishing execution of another command), it checks to see if there is a "$$$.SUB" file on drive A; if so, rather than read a command from the console, it reads the last record on the $$$.SUB file, shortens the file by one record, then executes the command. It keeps doing this until the file is shortened to zero length, at which point it is deleted.

The format of the records on the $$$.SUB file is as follows:

```
First byte - length N of the command
N bytes    - the command itself
1 byte     - zero
1 byte     - $
```

9. C83000E.BAS -- LOAD, EDIT, OR CREATE SYSTEM CONFIGURATION DATA

This included source file contains the subroutines used to create, edit, and load the system configuration data. The entry points provided in this routine are described below:

83000	If the CONFIG.FLAG% variable is equal to zero, an attempt is made to load the system configuration data. If the file cannot be located, it will be created, and the operator will be prompted to enter the required data.
83100	Create the system configuration control file and prompt the operator to enter the required data.
83150	Control editing of the system configuration control data.
83200	Display the configuration control data to the operator.
83300	Enter or edit configuration control data. If OPTION.CODE = 1 on entry, then the operator will be prompted to enter all data. If OPTION.CODE = 2, the operator will be prompted to edit existing data.
83400	This routine will open the control file and read the single record into memory. If the file cannot be located and opened properly, a return code of - 1 is provided to the calling routine.
83500	The configuration control record is replaced on the file and the file is closed.
83600	Validate the password if one has been defined.
83700	Masks the password with pound signs (#) when password is entered through terminal.

83800 Sets up the screen clear string, and sets the first bit in each character to zero.

83900 Converts a HEX input string from numeric to an ASCII character string. The input variable is V$ (HEX). The output variable is V$ (ASCII).

10. CINTL.BAS -- SET VALUES FOR SYSTEM CONFIGURATION DATA

This file contains the statements to dimension and set values for the variables used in C83000E.BAS.

The following is a listing of each of the variables and what their purpose is:

C.FILE	Unique file number for the system configuration file. Initially set to 2.
C.FILE.DRIVE$	Disk drive the system configuration file will be on. Initially set to 'A:'.
C.FILE.MASK$	Names the system configuration file. Initially set to 'SYSTEM.DDD'.
C.FILE.LEN$	Sets the record length of SYSTEM.DDD. Initially set to 500.
CC.TERM.TYPE.COUNT	Sets the value for dimensioning CC.TERM.TYPE.VAL$ array. Initially set to 5.
CC.TERM.TYPE.VAL$(X%)	Sets terminal type for SYSTEM.DDD. Zero thru 5 are set to user-defined clear screen, Sol, Hazeltine, Beehive, Soroc and Intertec, respectively.
CC.DELAY$	Puts a string of five CHR$(0)'s together.
CC.TERM.CLEAR$(X%)	Defines clear screen command for all but user-defined clear screen.
CC.MSGLEVEL.COUNT	Sets dimension of 2 for CC.MGLEVEL.VAL$ array.
CC.MSGLEVEL.VAL$(X%)	Sets variables 0, 1 and 2 equal to 'SUPPRESS ALL MESSAGES', 'OPEN/CLOSE/ CHAIN MESSAGES' and 'DISPLAY I/O TRACES', respectively.

11. C86900E.BAS -- SWIFT FILE COPY ROUTINE

This included subroutine is used to make a copy on disk of any data file or index created by PEARL-generated programs. It serves the function of the CP/M PIP command without having to exit the program. It is specific to PEARL-generated files only. The multi-buffering options offered by CBASIC are used to reduce the number of program halts than occur while the program is waiting for data to be transferred from disk to computer memory, and from memory to the disk. Special handling to provide for the imbedded line feed which occurs in the index files is included.

This routine will validate that sufficient space is available on the output diskette prior to initiating the copy.

The parameters required by this routine are as follows:

SOURCE.FILE.DRIVE$	The source file drive including the delimiter (For example, "A:").
SOURCE.FILE.NAME$	The name of the source file including the file type (For example, "MYFILE.DAT").
DESTINATION.FILE.DRIVE$	The output file drive in the same format as the input file drive.
DESTINATION.FILE.NAME$	The output file name in the same format as the input file name.
CC.DISK.SIZE%	The capacity of the output diskette in K bytes.
SAVE.CORE	The number of bytes of memory required for execution overhead not including memory required for CRUN or the intermediate file in memory. (Any free memory over and above this amount will be used for buffers.)

The return codes provided by this routine are as follows:

0	No errors, copy successful.
86901	The input and output files are the same.
86905	The source file could not be opened.
86906	Insufficient space on the output diskette.
86907	Insufficient memory space to allocate buffers.
86980	An index file copy validation indicated that the

number of records copied was not equal to the number of records expected in the index file.

12. C90000E.BAS -- MISCELLANEOUS UTILITY ROUTINES

The routines included in this file are used throughout the PEARL-generated programs to perform screen clear, pause to wait for operator input prior to continue, to monitor the keyboard for an interrupt from the operator, to validate the location of a program, and to then chain to another module.

The entry points provided are as follows:

90000 Screen clear. This is accomplished by sending the variable CC.SCREEN.CLEAR$ to the terminal.

90100 Monitor keyboard for operator interrupt. Used in conjunction with the report writer for processing.

The following data is returned in KEY.CHAR:

0	NO DATA WAITING
1	ESCAPE CAUGHT
2	RETURN CAUGHT
< 0	The complement of the value of any other character received from the keyboard.

90200 This routine sends a message to the terminal operator requesting that the operator press return to continue. The input of any character will cause processing to continue. The value of the character input will be returned in the PAUSE% variable.

90201 The same as 90200 except that a prompt is not issued to the operator.

90300 This routine will attempt to chain to the program specified in the CC.CHAIN.TO$ variable. If the program cannot be located, the operator will be instructed to change the program diskette and press return. The routine will then try again to chain to that program.

If the operator enters an ESCAPE rather than a RETURN, this program will attempt to chain to the program MAIN.

13. C90400E.BAS -- CHAIN TO NEXT PROGRAM IN CC.CHAIN.LIST$

This routine is included in all programs and is used to chain from one program to the next.

90400 This routine uses two variables.

CC.CHAIN.LIST$ This variable is used to set up a

	chain of programs to be executed without the return to the main menu.
CC.CHAIN.TO$	This variable is returned with the name of the next program to be chained to.

90500 This routine is used to turn on the printer. The variable used is:

PRT$	If this variable is set to "YES", then the printer will be initialized for output. Any other character will not initialize the printer.

90501 This routine will unconditionally initialize the printer for output.

90600 This routine will initialize the console for output.

NOTE:

The subroutines described in this section have been provided with a level number and the date the routine was last modified. If you modify any of the routines to fit your own needs, we suggest that you also modify the level number and the date. This will allow you to monitor changes in your own system.

APPENDIX D

INDEX PROCESSING SUBROUTINES

1. SUMMARY

The INDEX processing subroutines provided with PEARL Level 3 provide a facility to maintain an index file in which keys (paired with pointers to data records) can be inserted, deleted and retrieved from a random access file with a minimum amount of processing overhead. The keys are character strings 0 to 63 characters in length, and the pointers are integers in the range 0 to 32767. The keys are stored in the index in ascending lexical order, just as words are ordered in a dictionary. Keys are comprised of printable ASCII characters (including the space character); the "alphabetical order" used in the index "dictionary" is that of the ASCII character set.

The index is organized so that any key/pointer pair may be found directly without reading the entire index. Further, index routines may be instructed to allocate node buffers which are managed by a least-recently-used buffer replacement algorithm, so that references to index nodes can frequently be memory references rather than disk accesses to the file containing the index. This speeds up index processing considerably.

The utility of such a facility is seen in the following example:

> Suppose that a customer master file contains 1000 records. If there is a requirement to access a specific customer by name, and accesses to the file can be made only by record number, it might take two or three minutes or more to scan the file sequentially to find the specific customer. However, if the file is indexed by customer name using this index system, the same customer master record can be retrieved from the file for processing in less than a second.

2. SUMMARY OF SOME OF THE FUNCTIONS AND FEATURES PROVIDED WITH THE INDEX ROUTINES

2.1 Capability to process variable length keys (up to 63 characters per key).

2.2 Capability to insert with test for uniqueness (returns error code if key already exists), insert with no tests, retrieve pointer associated with specified key (returns error code if no such key is in this file), and return key/pointer pairs from the index sequentially, in lexical order of keys.

2.3 Capability to process multiple index files simultaneously.

2.4 Capability to maintain multiple index buffers in memory to reduce number of disk accesses required.

2.5 Capability to generate up to 8 levels of indexing in a tree-structured index. (Number of levels are determined dynamically as keys are added to the file.)

2.6 Capability to dynamically reuse file space when keys are deleted.

3. FILE STRUCTURE

The index information is maintained in a structure called a B-tree, which is made up of nodes. Each node contains a number of keys and associated pointers. The first node is called the root, or level zero, node. The pointers in the root node point to level one nodes, and so forth up to the current number of levels in the index tree. The pointers in the last-level nodes (called leaf nodes) are not pointers to nodes in the index, but the pointers supplied by the user during key insertion subroutine calls. These user pointers can be used to point to record numbers in a user file, or for any other purpose the user desires, as long as they are integers in the range 0 to 32767. This B-tree structure is physically realized as a CBASIC random-access file with a record length of 256; each node of the index occupies one record. In practice, each node tends to be about half full; so each record contains as many variable-length keys with pointers as will fit in about 128 bytes.

The first record of the index file is a control record specifying such items as the record number of the root node of the index, the current number of levels in the index tree, the date last updated, etc. (This data is read into memory at the time the file is opened, and rewritten to disk at the time the file is closed.) The nodes in the index are referred to by node number, with the root node having a node number of zero. The actual record number is obtained by adding the record number of the root node (given by one of the fields in the control record) to the node number. The records between the control record and the root record are available to the user for additional control records, if needed.

The following information is maintained in the control record of each index file:

IS.C.ROOT%()	Record number of root node.
IS.C.LEVELS%()	Number of levels in the index.
IS.C.LAST%()	Last physical record on the index file.
IS.C.LAST.DATA()	Last record used on the data file.
IS.C.LEN.DATA()	Length of data file record.
IS.C.OPEN()	Number of times file has been opened.
IS.C.CLOSE()	Number of times file has been closed.
IS.C.DATE()	Date file last updated.

IS.C.USER1()	Reserved for use by the user.
IS.C.USER2()	Reserved for use by the user.

The "()" is shorthand for "(IS.ID%)"; since multiple indexes can be simultaneously handled, all variables specific to a given index must be subscripted by the "current index number", IS.ID%. The index routines use only IS.C.ROOT%, LEVELS%, LAST%, OPEN and CLOSE. IS.C.LAST.DATA, LEN.DATA, and DATE are used by programs which PEARL generates. The user must not alter the values of ROOT%, LEVELS% or LAST% if the index is to operate properly. IS.C.ROOT%(IS.ID%) may be specified only at the time an index file is created, to vary the space between the control record and the start of the index structure for user purposes.

The format of each node in the index is as follows:

KP KP KP ... KP LF BLANKS CR LF

for a total of 256 bytes. CR and LF are ASCII characters, and BLANKS refers to the ASCII spaces inserted by the CBASIC runtime to pad the record out to its declared length. KP stands for a Key-Pointer pair, which looks like:

L KEY L PPP
where

L	is a one-byte integer whose value is 32 plus the number of characters in KEY (0-63); hence L ranges from 32 to 95, which lies within the range of printable ASCII characters.
KEY	is a key supplied by the user during a key insertion subroutine call, 0 to 63 characters long.
PPP	comprises 3 bytes representing a positive integer in the range 0-32767 as 3 digits in base 32 notation. Each byte has an integer value of 32 plus the value (0-31) of the corresponding base-32 "digit", yielding values in the range 32-63, all in the printable ASCII character range.

The key/pointer pairs within a node are always in lexically increasing order by key, from left to right.

The last KP entry for certain nodes may have the format:

L PPP
where

L	has a value of 31, signifying a "record length" of -1.
PPP	is as above.

This format serves as an "infinity pointer" in the root and intermediate nodes of the index, but never in the leaf nodes.

4. USER INTERFACE GUIDELINES

Four source files are provided with the PEARL for INDEX processing. These are:

4.1 ISINTL.BAS Initialization and Dimensioning of Control Data.

Must be %INCLUDEd in any program that uses the INDEX routines.

4.2 ISUPDATE.BAS Update and Access of Index Files.

%INCLUDE this file in any program that accesses/inserts/deletes keys into an index.

4.3 ISCAN.BAS Read Index Data File. (No Update Capability).

Use this source file in place of ISUPDATE.BAS when only index reads need to be done. This saves some space by deleting the insert/delete logic. The control record is still updated when the index is closed.

4.4 IS73000.BAS Create an Index File.

Must be included in any program that creates an index file. Index files are not created automatically by the open routine; they must exist beforehand.

In order to make sure that the DIM statements for the INDEX subroutines come before any references to the associated arrays as the compiler scans the source program, %INCLUDE ISINTL should be inserted in your source program near the top (while this may not be necessary with later versions of the compiler, it is advisable as a precaution). The statement numbers in ISINTL are in the 74000 range. For readability, the %INCLUDE for ISUPDATE, ISCAN, or IS73000 should be positioned so that their statement numbers (in the 70000 to 77000 range) fit in in numerical order with the other statement numbers in your program.

Prior to calling any of the INDEX routines the calling program should initialize the following variables:

IS.ID%	File Number. Value of 0 through 19 which refers to the current index file being processed.

IS.F.COUNT	Indicator of the number of indexed files.
	One less than the maximum number of index files opened and processed by a given program.
IS.NBUFS%	Indicator of the number of index buffers to be used.
	One less than the number of I/O buffers to be allocated by the INDEX routines.

NOTE

IS.F.COUNT is used in a dynamic dimension statement. IS.ID% is subsequently used as a subscript. If IS.ID% ever exceeds the value provided in IS.F.COUNT, a runtime error will occur when the limits of the DIM statement are exceeded. In order to conserve space, the IS.F.COUNT should be the actual maximum number of files processed, and IS.ID% should be assigned consecutive values beginning at zero.

The INDEX routines will assign a CBASIC file number one greater than the value of IS.ID%. Therefore, the calling routine should be careful not to duplicate the use of these file numbers in other parts of the application program.

The number of buffers specified in the IS.NBUFS% parameter will be shared dynamically by all index files being processed. Try allocating 5 buffers for one index file, or 10 for two index files. Buffers consume about 130 bytes of free space each, on the average. More buffers may result in fewer disk accesses or may not, depending on index size, number of indexes simultaneously in use, and access patterns. IS.NBUFS% may be set as low as 1 if desired, but this will result in much more disk activity.

5. FILE INITIALIZATION

In order to initialize an index file, the ISINTL and IS73000 source files must be included. The following is an example of the use of the INDEX routines to create an index file:

```
%INCLUDE ISINTL
       .
       .
    IS.F.COUNT = 0
    IS.ID% = 0
    IS.NBUFS% = 1
    GOSUB74000                          DIMENSIONVARIABLES
    IS.NAME$(IS.ID%) = "B:TESTFILE"      THE FILE NAME SHOULD
                                         INCLUDE THE DISK DRIVE
                                         BUT NOT THE TYPE. ".NDX"
                                         WILL BE APPENDED BY
                                         THE CREATE PROGRAM.

    IS.C.LEN.DATA(IS.ID%)=xxx           THESE VARIABLES ARE NOT
    IS.C.LAST.DATA(IS.ID%)=xxx          USED BY THE INDEX
    IS.C.USER1(IS.ID%)=xxx              ROUTINES BUT ARE WRITTEN
    IS.C.USER2(IS.ID%)=xxx              TO THE CONTROL RECORD OF
                                         THE CREATED INDEX FILE.

    IS.C.DATE(IS.ID%)=CC.DATE            PICK UP DATE.

    GOSUB 73000                          CREATE THE FILE.
      .
      .
%INCLUDE IS73000
```

In addition to the above variables, the following variables may be used to control processing:

CC.DATE	The current processing date. This variable is stored as a real number by the PEARL programs in the format YYMMDD.
CC.MSGLEVEL%	This variable is used to control the levels of messages generated by the INDEX routines.

0	Suppress all messages to the operator.
1	Open and close messages will be displayed.
2	Each buffer written to the file will be displayed.

NOTE:

When an index file is created it is automatically closed by the create program. If it is to be processed for update processing in the same program, the file must be opened prior to processing requests being sent to the INDEX subroutines.

6. UPDATE AND ACCESS PROCESSING

In order to update and access index data, the ISINTL and ISUPDATE source files must be included in the program. The following entry points are available in these routines:

74000	Dimension and initialization. User must GOSUB 74000 before calling any of the other index routines.
70000	Open an INDEX file.
70001	Close an INDEX file.
70002	Set sequential access range.

70003	Position to key (and return associated pointer).
70004	Add a new key (must not already exist).
70005	Delete first occurrence of specified key with its associated pointer from the index.
70006	Advance to next sequential key (return the key and its associated pointer).
70009	Add a new key (may be duplicate).
70010	Delete key at current position (useful following calls to 70006).
70011	Return current position (returns key and its associated pointer).

The following variables must be provided during the initialization and open processing:

IS.ID%	Current file number, required for all calls.
IS.F.COUNT	Required before call to 74000.
IS.NBUFS%	Required before call to 74000.
IS.NAME$(IS.ID%)	Required before call to 70000.

The following variables are used to reference keys and pointers during index file processing:

KEY$	The current key being processed.
D.R.ID	The pointer associated with the key.

The following are returned by the INDEX routines:

RCODE	A return code indicating status or errors. A value of zero indicates the request was satisfied successfully. A negative value is an information return indicating an exception condition. <u>A positive value indicates that a critical error has occurred</u>.
IS.OFF.END%	Returns logical TRUE (-1) on attempt to position index past highest key, otherwise FALSE (0).

7. CALLING THESE ROUTINES FROM YOUR PROGRAMS, IN DETAIL

The following descriptions include some or all of the following points:

- Required input to routine.
- Returns message.
- Description.
- Example.

74000 Dimensions arrays and initializes buffer controls.
Required input: IS.F.COUNT and IS.NBUFS%.
Example:

```
   IS.F.COUNT=1.0   Rem make room for 2 indexes, 0 and 1
   IS.NBUFS%=10     Rem allow plenty of buffers
   GOSUB 74000      Rem initialize
```

70000 Open an index file.
Required input: IS.ID% and IS.NAME$(IS.ID%)
Returns:

```
   RCODE = 0 if successful.
         = 70191 if file not found, or if it is empty
   IS.C.LAST.DATA(IS.ID%) = value read from file
   IS.C.LEN.DATA(IS.ID%)  = value read from file
   IS.C.OPEN(IS.ID%)      = value read from file + 1
   IS.C.CLOSE(IS.ID%)     = value read from file
   IS.C.DATE(IS.ID%)      = value read from file
   IS.C.USER1(IS.ID%)     = value read from file
   IS.C.USER2(IS.ID%)     = value read from file
   IS.OPEN(IS.ID%) = 1.0
```

Description:
If CC.MSGLEVEL% > 0, displays an opening message. The file is opened as a CBASIC random access file with a record length of 256. The control record (record number 1) is read into the control variables. The OPEN count is incremented, and the control record is written back out to the file (hence, if processing bombs before the file is closed, the OPEN count will be one greater than the CLOSE count, as they appear on the file). If the first record does not contain the proper sequence of numbers separated by commas, the CBASIC runtime will terminate processing with an error code.
Example:

```
   IS.ID%=1
   IS.NAME$(IS.ID%)="B:TEST7"
   GOSUB 70000  Rem opens index file B:TEST7.NDX
```

70001 Close an index file.
Required input: IS.ID%
Returns:

```
   IS.C.CLOSE(IS.ID%) = IS.C.CLOSE(IS.ID%) + 1
   IS.OPEN(IS.ID%) = 0
```

Description:
Increments IS.C.CLOSE(IS.ID%), then writes the control record back to the file. Writes all outstanding node

```
         buffers for that file out to the file.  If CC.MSGLEVEL%
         > 0, displays a closing message.  Executes a CBASIC
         CLOSE of the specified file.
      Example:
         IS.ID%=4
         CC.MSGLEVEL%=1  Rem activate close message
         GOSUB 70001

70002 Set sequential access range.  Used with 70006.
      Required input:
         IS.ID%    Index file number to which this applies
         IS.LOW$   Beginning of range to set
         IS.HIGH$  End of range to set
      Returns:
         RCODE = 0 normally
               = 70191 if error in index pointers
      Description:
         Positions the index specified by IS.ID% to key IS.LOW$
         (see 70003 for detailed description).  Sets
         IS.HI.CUTOFF$(IS.ID%) = IS.HIGH$ for future reference.
      Example:
         IS.ID%=2  Rem select index file number 2
         IS.LOW$="APPLE"
         IS.HIGH$="BANANA"
         GOSUB 70002  Rem next call to 70006 with IS.ID% set to
                      Rem 2 will return the first occurrence of
                      Rem "APPLE" if it exists, or the next
                      Rem higher key in index 2 if not.

70003 Position to first occurrence of specified key.
      Required input:
         IS.ID%  index file number to which this applies
         KEY$    key value to look up
      Returns:
         RCODE     = 0  if key found
                   = -1 if key not found
                   = 70191 if internal error in index
         D.R.ID    = 0 if key not found
                   = pointer value supplied by user when this
                     key/pointer pair was originally inserted in
                     the index, if key is found.  If there is
                     more than one key/pointer pair with key
                     equal to KEY$, the one which occurs first in
                     the index is positioned to.  (In the case of
                     multiple occurrences of keys in the index,
                     the one most recently inserted always
                     becomes the first occurrence--i.e., "last
                     in, first out".)
         IS.OFF.END% = -1 if positioned off the end of the
                     index, else = 0.
      Description:
         Positions the index specified by IS.ID% to the first
         occurrence of key KEY$; or if it is not there, to the
         smallest existing key which is still greater than KEY$
         (i.e., the key which would come directly after KEY$ if
```

KEY$ were in the index). If KEY$ is greater than all the keys in the index, the index is positioned off the end.

Example:
```
IS.ID%=0  Rem select first index file number
INPUT "Enter test key: ";KEY$
IF RCODE = 0 THEN \
  PRINT KEY$;" found with D.R.ID of ";D.R.ID \
ELSE PRINT KEY$;" not in index."
```

70004 Insert a unique key in the index.

Required input:

IS.ID%	index file number to which this applies.
KEY$	the key to insert.
D.R.ID	the pointer value (an integer in the range 0-32767) to associate with this insertion.

Returns:

RCODE	= 0 if successful
	= -1 if key already exists (no insertion done)
	= 70191 if index file error.

Description:

Positions to the key which would follow KEY$ after insertion. If this key equals KEY$, foregoes the insertion and returns RCODE = -1. Else moves all the keys in the node from this key to the end over to make room for KEY$ and its associated pointer, and inserts it. If there is not enough room in the node, splits it into two nodes and adjusts pointers in higher-level nodes accordingly.

Example:
```
IS.ID%=3
INPUT "Enter key and D.R.ID: ";KEY$,D.R.ID
GOSUB 70004  Rem insert unique
IF RCODE <> 0 THEN PRINT \
  "Sorry, key already exists--not inserted"
```

70005 Delete first occurrence of key from index.

Required input:

IS.ID%
KEY$

Returns:

RCODE	= 0 if successful
	-1 if KEY$ not in index
	70191 if error in index file.

Description:

Positions to first occurrence of KEY$. If it exists, deletes the key/pointer entry from the node and moves the remaining keys in the node down to take up the space. A sequence of deletions may result in a node becoming completely empty (i.e., LF, 253 blanks, CR, LF). However, subsequent insertions of keys which fall in the same range as the node originally contained will go back into that same node, thus reusing the space. The INDEX routines are programmed to handle empty nodes correctly.

Example:
```
   PRINT "Enter index number (0 to ";IS.F.COUNT;
   INPUT "): ";IS.ID%
   INPUT "Enter key to delete: ";KEY$
   GOSUB 70005  Rem delete first occurrence of KEY$
   IF RCODE <> 0 THEN PRINT \
     "Sorry, nothing to delete."
```

70006 Advance to next sequential key.
Required input: IS.ID%
Returns:

RCODE = 0 if there is a next key and it is lexically less than or equal to the IS.HIGH$ specified in the last 70002 call for this index file number (and remembered in IS.HI.CUTOFF$(IS.ID%)). For example, where CAMEL comes before CAT; this is to be distinguished from CBASIC's string < and > operators, under which the shorter string is always the lesser, so that ZZZ comes before APPLE and CAT comes before CAMEL. The lexical < test is implemented in the INDEX routines by truncating the longer string to the length of the shorter, and then using CBASIC string comparison. If they are equal over the shorter string, then the shorter is taken as the lesser.)
= -1 if there is no next key or the next key is greater than the last IS.HIGH$ set for this file.

KEY$ = the next key if there is one, or the null string if we have advanced off the end of the index.

D.R.ID = pointer value associated with next key, or 0 if off end of index.

IS.OFF.END% = -1 if we have advanced off the end of the index, otherwise 0.

Description:

Positions index to the next entry, if there is one, and extracts the key and pointer values for return to the user. Note that multiple entries with the same key value are nevertheless distinct; if the index is positioned at the first occurrence of CAT, and the key CAT occurs three times in the index, a call to 70006 moves over to the second occurrence, and another call moves over to the third occurrence. Each call returns a different D.R.ID (unless they were all inserted with the same pointer value for some reason).

Example:
```
   INPUT "Enter which index: ";IS.ID%
   INPUT "Enter lower limit";IS.LOW$
   INPUT "Enter upper limit";IS.HIGH$
   GOSUB 70002  Rem set sequential access range
   PRINT "HERE ARE ALL THE KEYS IN INDEX ";IS.ID%;
   PRINT " BETWEEN ";IS.LOW$;" AND ";IS.HIGH$;
   PRINT ", INCLUSIVE:"
   GOSUB 70006  Rem get next key
   WHILE RCODE = 0
     PRINT KEY$,D.R.ID
     GOSUB 70006  Rem get next key
   WEND
   PRINT "END OF LIST"
```

70009 Insert a (possibly non-unique) key into an index.
Same as 70004 except RCODE will never be -1. If KEY$ is the same as the key in an existing index entry, a new entry will nevertheless be made with the D.R.ID specified on this call. The new entry will become the first occurrence of that key in the index, so that the 70003 and 70005 calls specifying the same key will now reference this latest insertion. The 70006 calls passing through all the occurrences of a multiply-occurring key will get the most recent insertion first; i.e., in backwards order from the way they were inserted.

70010 Delete key at current position.
Required input: IS.ID%
Returns:
   KEY$ = key value of entry deleted.
   D.R.ID = pointer value in deleted entry.
   IS.OFF.END% = 0 if there was a key to delete; else -1.
Description:
   If off end of specified index, returns IS.OFF.END%=-1, KEY$="", D.R.ID=0. Otherwise removes key/pointer entry at current position and moves remaining entries in node down to close up the space, as in 70005 description. This call provides the only convenient way to delete a middle occurrence of multiply-occurring keys, short of deleting all the preceding occurrences to be able to get at it. One simply positions to the desired entry using 70002 and 70006 (get next) calls, then calls 70010 to delete the entry. Also if you are scanning through an index using the D.R.ID's to reference invoice records on an invoice file, for example, and you want to delete references to invoices which are all paid up, you can call 70006 to get the next D.R.ID, look up the invoice, and if you want to delete it, call 70010; then call 70006 to get the next one, and so on.

70011 Return current position.
Required input: IS.ID%
Returns:
   KEY$
   D.R.ID
   IS.OFF.END%
Functions just described above for 70010, except no delete is done. If no reference has been made yet to the index (except for the necessary call to 70000 to open the file), it has an implicit position at the very first key of the index.

NOTES:

1. INDEPENDENCE OF INDEXES. Simultaneously opened indexes are completely independent of each other. All pointers and flags which must retain their values across calls to the INDEX routines are stored in array variables subscripted by IS.ID%. Each index maintains its last position and status regardless of intervening processing calls on other indexes. Everything is kept separate by IS.ID%. Thus, a series of calls to 70006 with IS.ID% = 0 returns exactly the same series of keys whether index 0 is the only one open, or the calls are interspersed with opens, closes, inserts and deletes for indexes 1, 2, etc.

2. INDEX POSITION AND ADVANCEMENT. Each index always has a current position. When first opened, the position is at the first key of the index; subsequent processing calls move the position around within the index. Note that calls to 70006 (get next) sometimes return the current position, rather than advancing to the next key, in order to provide natural operation. Current position is returned when:

   a. the immediately preceding call for this index was to 70002 (set range), so that the first key of the range won't be skipped, and

   b. the immediately preceding call for this index was to 70005 or 70010 (delete), since the removal of the key at the current position has moved the succeeding key down to the current position. Executing an advance would cause that moved-down key to be skipped.

   Otherwise, the position is advanced to the next key.

3. REORGANIZATION. After a considerable number of key deletions have been done, there tends to be much wasted space in an index. It is a good idea to occasionally reorganize the index, using 70006 calls to read all the keys in sequence from the old index while inserting them into the new index. Since an index is usually used to reference records in an associated data file, this is an excellent opportunity to rewrite the data file also. Records may be rewritten in order of key rather than the order they were originally written, and records which have been marked as deleted may be omitted.

Here is a sample program using the INDEX routines.

```
REM
REM  INDEX DEMO PROGRAM
REM
%INCLUDE ISINTL
     PRINT "INDEX demonstration program"
     INPUT "How many indexes?";N
     IS.F.COUNT=N-1
     INPUT "How many node buffers?";IS.NBUFS%
     GOSUB 74000  Rem dimension and initialize
REM
REM  Create and open the index files for processing
REM
     FOR IS.ID% = 0 TO IS.F.COUNT

     PRINT "Enter file drive:name for index ";IS.ID%;
     INPUT ": ";IS.NAME$(IS.ID%)
     GOSUB 73000  Rem create and close the file
     GOSUB 70000  Rem open file for processing

     NEXT IS.ID%
REM
REM  Input and distribute keys to indexes
REM
     PRINT "Enter keys to be inserted in indexes."
     PRINT "Enter a negative index number when finished."
     INPUT "IS.ID%,KEY$,D.R.ID=";IS.ID%,KEY$,D.R.ID
     WHILE IS.ID% >= 0
       GOSUB 70009  Rem insert key in specified index
       INPUT "IS.ID%,KEY$,D.R.ID=";IS.ID%,KEY$,D.R.ID
     WEND
     PRINT "Finished entering keys"
     PRINT "Here are index contents in lexical order of keys:"
     FOR IS.ID% = 0 TO IS.F.COUNT

     PRINT "FOR INDEX ";IS.ID%
     IS.LOW$=""   Rem lowest possible key value
     GOSUB 70002  Rem set access range
     GOSUB 70006  Rem get first key of index
     WHILE NOT IS.OFF.END%
       PRINT KEY$,D.R.ID
       GOSUB 70006  Rem get next key
     WEND
     PRINT "END OF INDEX"
     PRINT

     NEXT IS.ID%
     PRINT "End of demonstration"
%INCLUDE ISUPDATE
%INCLUDE IS73000
     END
```

8. ERROR CODES

The following error codes are returned by the INDEX routines in the RCODE variable when a critical error is encountered.

70191 1.) AN ATTEMPT WAS MADE TO OPEN AN INDEX FILE WHICH AS NOT LOCATED ON THE SPECIFIED DISKETTE.

2.) AN ATTEMPT WAS MADE TO READ A RECORD BEYOND THE END OF THE INDEX FILE.

CAUSE: In either case, the error code is returned because of the end of file exit provided by CBASIC. In the first case, the file was not located on the diskette. In the second case, an attempt to read a record which was beyond the end of file occurred.

RESPONSE: In the second case, the error occurred due to an invalid pointer in the index file. The most likely reason for this error is that the file was not closed properly on a previous update.

73091 FILE TO BE CREATED ALREADY EXISTS.

CAUSE: The error occurs in the index initialization routine at the time a request is made to create an index file when an index file by the same name already exists on the diskette.

RESPONSE: Either delete the old index file which exists on the diskette being used to create a new file, or use a diskette with no index file on it.

APPENDIX E

PEARL MESSAGES AND CODES

There are three levels of error codes and messages returned by PEARL. These are:

(W) Informational and warning messages. These messages are not critical, but do require action be taken by the user in order for processing to continue.

(E) Non-critical errors. A condition has been detected resulting in the termination of a program but can be corrected by the user so the program can be rerun.

(C) Critical errors. When an error in this category occurs, it is due to a program logic error, bad data on the file, or some other situation which cannot be corrected by the operator.

111 (xx yy) Errors Outstanding.

CAUSE: Indicates that there are errors outstanding on the control records. xx is the number of errors outstanding, and yy indicates the file number to which there errors are related.

RESPONSE: Correct errors and run a data element report to verify that all errors have been corrected; then proceed to generate system.

Notes:

1. If the operator wishes, the system may be generated even if outstanding errors exist. However, the generated system may not complete, or may not execute properly if this is done.

2. Errors are detected by executing the report processor and listing the data elements which have been defined. If this report is not run prior to generating a system, the programs generated will not execute and/or compile properly.

   Once the errors have been corrected, the report processor must be run again to clear the error flag from the control file.

119 (W). FIELD SIZE OF xxxx EXCEEDED FOR dddddddddddddd.

CAUSE: The size of a numeric data element exceeds the space allocated for that data element in the data record. The number of digits allowed for the data elements is provided in xxxx, and the description of the data element is provided in dddddddddddddd.

REPONSE: Edit the record and reduce the value of the data element so it is no longer than the specified field size.

<u>140</u> <u>(W).</u> <u>FILE ACCESS METHOD ERROR.</u>

CAUSE: A file was defined with an unknown access method.

RESPONSE: Delete the file or redefine it to:
1 - Random
2 - Indexed

<u>143</u> <u>(W).</u> <u>MASTER FILE ACCESS METHOD ERROR.</u>

CAUSE: A master file was defined as other than indexed.

RESPONSE: Change the access method to indexed or redefine the file from master file to any other appropriate file type.

<u>144</u> <u>(W).</u> <u>FILE ACCESS METHOD ERROR.</u>

CAUSE: A miscellaneous indexed file was defined as a random access file.

RESPONSE: Change the access method or change the file type.

<u>150</u> <u>(W).</u> <u>FILE ACCESS METHOD ERROR.</u>

CAUSE: A transaction, historical transaction, or a miscellaneous random access file was designated as indexed.

RESPONSE: Change file type, or file access method.

<u>170</u> <u>(E).</u> <u>MULTIPLE MASTER FILES.</u>

CAUSE: More than one master file was defined.

RESPONSE: Delete all master files except one.

<u>171</u> <u>(E).</u> <u>NO MASTER FILE.</u>

CAUSE: A transaction file was defined, and a master file was not.

RESPONSE: Change the definition of the transaction file or add a master file to the system.

<u>172</u> <u>(E).</u> <u>NO TRANSACTION FILE.</u>

CAUSE: An historical transaction file was defined, and a transaction file was not.

RESPONSE: Redefine the historical transaction file or add a transaction file to the system.

<u>180</u> <u>(W).</u> <u>SYSTEM NOT INITIALIZED.</u>

CAUSE: An attempt to load the system configuration control information failed. This was due to the fact that the system control file (B:PEARL.CCT) was not located.

RESPONSE: Place the diskette with the appropriate PEARL control data file on disk unit B and re-run the request, or use option 1 in the main selection menu to create a system control file.

<u>200</u> <u>(C).</u> <u>SYSTEM FAILURE.</u>

CAUSE: An attempt failed to add a data record to the data definition file (PEARL.CCT). The data record was added properly, but the index pointer to the record was not. A record exists on the file that has no index pointer. This is an internal program failure.

RESPONSE: Refer to Appendix K.

<u>300</u> <u>(W).</u> <u>FILE HAS NOT BEEN DEFINED.</u>

CAUSE: When an attempt to edit or add specifications for a file as made, the file control record was not located.

RESPONSE: Use option 3 on the main menu to add file control information and to create a file control record.

<u>1000</u> <u>(W).</u> <u>FILE HAS NOT BEEN CLOSED PROPERLY.</u>

CAUSE: During a previous attempt to update the file, normal exit processing from the program was not used, and therefore, the control file was not closed properly. This could occur as a result of a power failure, a user's reboot, exit in the middle of a program, turning off the power on the computer without returning to the operating system, or if a program error causes abnormal termination of a program after an update process has been initiated by the user.

RESPONSE:
1. Recover the control file using a backup copy from another diskette.
2. Enter CONTINUE in response to the prompt to continue processing. When this option is taken, there is a chance some of the data entered during the previous update will be lost, or that some of the pointers in the control file index will be incorrect. If the latter is the case, the program may terminate during subsequent processing and the file may not be recoverable.

3. Reinitialize a new file and re-enter the control information.

1090 (W). CONTROL FILE COULD NOT BE LOCATED.

CAUSE: The PEARL.CCT control file could not be located on the diskette on disk unit B:.

RESPONSE: Rerun the program with the correct PEARL.CCT file on the diskette on DISK unit B:.

1291 (C). LOW INDEX KEY RANGE ERROR.

1292 (C). HIGH INDEX KEY RANGE ERROR.

When either 1291 or 1292 is received as an error code, the following information will also be provided:

```
ATTEMPT TO ACCESS A RECORD OUT OF FILE RANGE.
LOW RECORD KEY IS lllll.
HIGH RECORD KEY IS hhhhh.
RECORD BEING ACCESSED IS aaaaa.
FILE BEING PROCESSED IS B:PEARL.CCT.
```

CAUSE: An entry in the PEARL.NDX file is pointing to a record on the PEARL.CCT file not within the range of records on the control file. lllll indicates the low record number on the file, hhhhh indicates the high record on the file, and aaaaa indicates the record the program was attempting to access.

This error is a result of the file not having been closed properly after being updated. The pointer to a record was placed in the index, but because the PEARL.CCT file was not closed properly, the data record associated with the index pointer was not within the range of records which can now be accessed from the control file.

RESPONSE: Refer to RESPONSE for error code 1000.

1293 (C). KEY VALIDATION ERROR DURING I/O PROCESSING.

CAUSE: The record which has been accessed from the PEARL.CCT file has a key which is not the same as the KEY pointer in the PEARL.NDX index file.

RESPONSE: Refer to error 1000.

1360 (C). INVALID RECORD TYPE ON CONTROL FILE.

CAUSE: A record was accessed from the control file of an unknown type and cannot be processed by the PEARL programs. This can be a result of a disk I/O error not detected by the operating system. It could also be a result of a logic defect in the PEARL programs.

RESPONSE: Refer to Appendix K.

1809 (C). ERROR 1809. RECORD NOT FOUND. key-record.id

CAUSE: An attempt was made to delete a record, but the record was not located. **key** indicates the key of the record being deleted; **record.id** is the record number of the record on the PEARL.CCT file which could not be located.

RESPONSE: If this error has occurred after the file has not been closed properly, it is most likely due to an incomplete update. The control file should be recovered from a backup file.

This error may also be caused by a bad copy of a PEARL program, or a disk I/O error on the diskette which contains the PEARL control file. Contact CPU for assistance.

3135 (W). STORAGE FORMAT RESET

CAUSE: The storage format is not compatible with the validation option which has been selected. This is a warning to the operator that the storage format has been reset to a storage format which is compatible with the validation option which has been selected.

RESPONSE: If a different storage format is required, first edit the validation option to type zero, then edit the storage format field as desired.

6490 (W). XXXXX FILE COULD NOT BE LOCATED.

CAUSE: The program logic tables (PLT03x.NDX) could not be opened for processing during program generation.

RESPONSE: Insure that the diskette containing the PLT03x.NDX file is on disk A during the program generation phase.

24065 (C). MENU RECORD INVALID.

CAUSE: An attempt to read a menu detail record failed. The index key is invalid.

RESPONSE: Refer to Appendix K.

27010 (C). FILE CONTROL RECORD INVALID.

CAUSE: An attempt to validate the record accessed as a file control record failed.

RESPONSE: Refer to Appendix K.

27020 (E). **ERROR 27020-(n)**

CAUSE: The maximum number of files (n) which can be processed during full system generation with all files has been exceeded.

RESPONSE: The programs for each file must be generated one at a time using option 2 on the generation menu.

27050 (E). No files located.

CAUSE: An attempt was made to GENERATE AN ENTIRE SYSTEM, but no files (or the current file) have been defined.

RESPONSE: Define the files.

27190 (C). **ERROR 27190 (ppp 1111 ff)**

CAUSE: An error was encountered while processing a file list during GENERATION OF ENTIRE SYSTEM (ALL FILES). The ppp indicates the position in the list, 111 is the list, and ff is the current file.

RESPONSE: Make a hard copy of the error with the message level set at 2. Contact the CPU PEARL support unit.

27200 (E). **ERROR 27200**

CAUSE: More than one master, transaction or history file was defined.

RESPONSE: Redefine description file.

27201 (E). **ERROR 27201**

CAUSE: No master file was defined when a transaction file was defined.

RESPONSE: Define a master file.

27202 (E). **ERROR 27202**

CAUSE: No transaction file was defined when an historical transaction file was defined.

RESPONSE: Define a transaction file.

28010 (C). REPORT CONTROL RECORD INVALID.

CAUSE: An attempt to access a report control record was made, but the record accessed did not validate as a report control record.

RESPONSE: Refer to Appendix K.

30020 (C). DATA ELEMENT RECORD KEY IS INVALID.

CAUSE: An attempt was made to access a data element record, but the record was not a data element record.

RESPONSE: Refer to Appendix K.

35020 (C). FILE CONTROL RECORD NOT LOCATED.

CAUSE: While building the default menu control, an attempt was made to locate a file control record, but the record could not be found.

RESPONSE: Verify that the PEARL control file has been properly closed after update processing. (Refer to error 1000.) If it has, contact CPU for technical assistance.

35040 (E). **ERROR 35040. MULTIPLE TRANSACTION FILES.**

CAUSE: More than one transaction file has been defined. Because the logic of the generated system only allows one transaction file, the menu control data cannot be properly generated.

RESPONSE: Reduce the number of transaction files to 1, then re-execute the option to generate the default menu control data.

35041 (E). **ERROR 35041. MASTER FILE NOT LOCATED.**

CAUSE: An attempt has been made to generate the default menu control data, but no master file has been defined with a file number lower than the transaction file.

RESPONSE: Define a master file with a number less than the transaction file, or edit the menu control data that has been created so that it is correct according to programmer expectations.

50028 (E). **ERROR 50028-n. CROSS FILE VALIDATION ERROR.**

CAUSE: An error in the use of the cross file validation option has been detected. The nature of the error is as follows as indicated by the n portion of the error message:

1. The format of the variable code name is not valid.

2. The variable name is not defined on the master file.

3. The variable is not defined on the master file as the primary key.

RESPONSE: Make required modifications to the data element control data. Refer to CROSS FILE VALIDATION specification requirements in the section of the manual dealing with the entry of data element control information.

50085 (W). EXCEEDING CONTROL RECORD LENGTH.

CAUSE: The combined length of the variables in a control record exceeds the length of the data records in the file.

RESPONSE: Either redistribute the variables in the control record to other control records, delete some of the variables in the control record, or increase the size of the data records in the file by increasing the length and/or number of variables defined in each data record.

50150 (W). STORAGE FORMAT ERROR.

CAUSE: The key field of a RANDOM access file is not a numeric key. (The first field of the data record is always used as the key for the record.)

RESPONSE: Either change the storage format to floating point or integer for the first data element in the record, or insert a variable at the beginning of the record which has a storage format of floating point or integer.

50151 (E). **ERROR 50151. MAXIMUM OF 34 EDITABLE FIELDS EXCEEDED**

CAUSE: More than 34 data elements which can be edited have been defined in the PEARL control file.

RESPONSE: Reduce the number of data elements to 34 or less.

50210 (W). ARRAY VALIDATION ERRORS DETECTED.

CAUSE: The ARRAY VALIDATION option has been specified for the data element, but no entries have been specified for validation.

RESPONSE: Change the validation option associated with the data element or add the required entries used for array validation to the control file.

54020 (C). REPORT INDEX KEY ERROR.

CAUSE: A report data sequence key does not match the key on the data record.

RESPONSE: See Appendix K.

57200 (W). EDIT MASTER FILE RECORD KEY

CAUSE: The data element specified as a master file primary key was defined as uneditable.

RESPONSE: Redefine the master file primary key from no edit to edit.

57201 (W). TRANSACTION FILE CROSS VALIDATION KEY TYPE ERROR.

CAUSE: A data element on the transaction file does not have the same variable type as the master file primary key.

RESPONSE: Change transaction file cross file validation key type to match the primary key type on the master file.

57202 (W). EDIT TRANSACTION FILE CROSS FILE VALIDATION KEY.

CAUSE: A cross file validation key defined on the transaction file is not editable.

RESPONSE: Redefine the transaction file cross file validation key from no edit to edit.

57203 (W). LENGTH OF TRANSACTION CROSS FILE VALIDATION KEY.

CAUSE: The transaction file cross file validation key is not the same length as the length of the master file primary key.

RESPONSE: Change length of transaction file cross file validation key to match the length of the master file primary key length.

57204 (W). HISTORICAL TRANSACTION FILE CROSS FILE VALIDATION KEY TYPE ERROR.

CAUSE: The historical file cross file validation key does not have the same variable type as the master file primary key variable.

RESPONSE: Change the historical transaction file cross file validation key type to match the master file primary key type.

57205 (W). EDIT HISTORICAL FILE CROSS FILE VALIDATION KEY.

CAUSE: The historical transaction file cross file validation key is defined as noneditable.

RESPONSE: Redefine the historical file cross file validation key from no edit to edit.

57206 (W). HISTORICAL TRANSACTION FILE CROSS FILE VALIDATION KEY LENGTH ERROR.

CAUSE: The historical transaction file cross file validation variable does not have the same length as the master file primary key.

RESPONSE: Change the length of the cross file validation key to match the length of the master file primary key.

57240 (W). MASTER FILE PRIMARY KEY ERROR.

CAUSE: The first variable on a file defined as a master file is not defined as a primary key.

RESPONSE: Change the master file designation from master file to some other file type, or place a primary key as the first variable on the master file.

70191 (C).

1) AN ATTEMPT WAS MADE TO OPEN AN INDEX FILE WHICH WAS NOT LOCATED ON THE SPECIFIED DISKETTE.

2) AN ATTEMPT WAS MADE TO READ A RECORD BEYOND THE END OF THE INDEX FILE.

CAUSE: In either case, the error code is returned because of the end of file exit provided by CBASIC. In the first case, the file was not located on the diskette. In the second case, an attempt to read a record which was beyond the end of file occurred.

RESPONSE: In the second case, the error occurred due to an invalid pointer in the index file. The most likely reason for this error is that the file was not closed properly on a previous update.

99901 (C). LEVEL VALIDATION ERROR.

CAUSE: An attempt has been made to use PEARL programs at one level with a menu control program at another level.

This error may also occur if the version of the CRUN programs being used is 2.03 or earlier.

RESPONSE: Contact CPU and provide them with the identity of the dealer or person from whom you purchased your copy of PEARL. Also provide them with the serial number displayed when you signed on to the main menu program.

<u>99900(C). INTERNAL SYSTEM ERROR.</u>

PLEASE CONTACT CPU DIRECTLY (503-363-8929) Ask for the PEARL liaison person in Customer Support.

APPENDIX F

ERROR CODES FROM GENERATED PROGRAMS

ERROR CODE

-999 Previous posting process failed.

CAUSE:
1. Power failure.
2. Master file record deleted.
3. Master file integrity lost.

RESPONSE:
1. Use the Master File Reorganization program to regenerated the master file.
2. Recover data files from backups.

-998 Previous closing process failed.

CAUSE: System failure.

RESPONSE: Recover all data files from backup copies previous to the last closing process and reinitialize the closing process.

-997 Transaction and master files are not in sync.

CAUSE: The transaction data file is of a different posting or closing period than that of the master data file.

RESPONSE: Recover both the data files from backups that are in sync and bring them up to date by repeating any processing that has occurred.

-997 Master and historical data files are not in sync.

CAUSE: Master and historical data files are not in sync. They are of different posting or closing periods.

RESPONSE: Recover historical and/or master data files of the same posting or closing period and update them if needed.

-995 Posting not completed.

CAUSE: An attempt was made to close a period or a year before all transactions have been posted to the master file.

RESPONSE: Post unposted transactions.

-2 An attempt was made to process a deleted record as a nondeleted record.

CAUSE: The operator attempted to edit a transaction file record that has previously been deleted.

RESPONSE: Check the record for deleting the specified transaction.

1 A value of + or -32768 was exceeded for an integer.

CAUSE: A value of + or - 32768 was exceeded for an integer.

RESPONSE: Do not enter a value less than or greater than 32768.

(Data File input/output errors)
(io = 10 - master file, 11 - transaction file, 12 - history file)

io = I/O Routine Address Base Number

io000 Data File Open Error.

CAUSE: The last time the data file was processed, it was not closed properly. This may cause the file to be incomplete.

RESPONSE:
1. Master File:
   a. Enter "CONTINUE" and proceed.
   b. Build a new master file index using the reorganization program.
   c. Recover master file and index from backups and bring them up to date.
2. Transaction or Historical Files:
   a. Enter "CONTINUE" and proceed.
   b. Recover data files from backups and bring them up to date.

io090 A Data File Not Found.

CAUSE:
1. Data file is not created.
2. Wrong disk in appropriate data file drive.

RESPONSE:
1. Create a new data file by selecting option 1 on the main menu and following procedures in Appendix D.

2. Place correct data file disk in the appropriate disk drive.

io095 Data File Lost Integrity.

CAUSE: Internal file integrity control variables are no longer valid.

RESPONSE:

1. Master File:

   a. Build a new data file using the reorganization option on the main menu.

   b. Recover master file and index from backups and bring them up to date.

2. Transaction and History Files:

   Recover data files from backups and bring them up to date.

io291 A Transaction Record Number Request Too Small.

CAUSE:

1. A transaction data file record number was requested that had a value smaller than the lowest record number on the file.
2. Wrong transaction file is being processed.
3. Transaction file lost its integrity.

RESPONSE:

1. Make sure that your request is valid.
2. Correct file.
3. Recover transaction file from backups and bring it up to date.

io292 A Transaction Record Request Too Large.

CAUSE: Same as io291 except that the transaction value is too large.

RESPONSE: Same as io291.

io293 A master file index key does not match the key present on the master data record.

CAUSE: Master file index lost its integrity.

RESPONSE: Regenerate a new master file index by selecting option to reorganize the master file on the main menu and option 2 on the reorganization menu.

io300 A master file deleted record pointer does not point to a delete record.

CAUSE: Master file internal control variables have lost integrity.

RESPONSE: Regenerate new master data and index files by selecting the reorganization option on the main menu and option 1 on the reorganization menu.

io670 Secondary key not found on master file index.

CAUSE: Same as io293.

RESPONSE: Same as io293.

io671 Secondary key not found on master file index.

CAUSE: Same as io293.

RESPONSE: Same as io293.

20020 A master file record to which transactions were to be posted could not be located.

CAUSE:

1. The master file lost its integrity.
2. The master file record to which transactions were to be posted was deleted.
3. The transaction file lost is not a current one.

RESPONSE: When one of the following steps have been executed, backup transactions and then repost to the master file.

1. Regenerate new master file and index using the reorganization option on the main menu and option 2 on the reorganization menu.
2. a. Check for correct master file on disk.

   b. The record has been mistakenly deleted -- reenter it.

   c. Delete the bogus transaction file record.
3. Recover the master and transaction files from backups and bring them up to date and repost.

Errors Related to Indexed File Reorganization Process

29080 Master Data File in Error.

CAUSE: The file control variables related to file integrity were invalid while a process of reorganization was in progress.

RESPONSE: Enter IGNORE.

29085 An attempt to rename a master file reorganization work file ($$$.DATE) to the original master file name failed. data file name failed.

CAUSE: System error.

RESPONSE: Attempt to rename manually as follows:

```
A>REN B:EM030.DAT=B:$$$.DAT
```

and make a new copy of your system disk from backups.

29185 The attempt to rename the master reorganization work index file ($$$.NDX) to the original master file index name failed.

CAUSE: System error.

RESPONSE: Attempt to rename the index work data file to the master index name as follows:

```
A>REN B:EM030.NDX=B:$$$.NDX
```

and make a new copy of your system disk from backups.

29200 The index file data file was not closed properly the last time it was processed.

CAUSE: Same as io000.

RESPONSE: Enter "CONTINUE" and proceed.

29600 Master File Index No Good.

CAUSE: An attempt to reorganize a master file failed because the index file was unusable.

RESPONSE: Return to reorganization menu and select option to regenerate master file index. Then proceed to reorganize the master file.

70191 1) An attempt was made to open an index file which was not located on the specified disk.

2) An attempt was made to read a record beyond the end of the index file.

CAUSE: In either case, the error is returned because of the end of file exit provided by CBASIC. In the first case, the file was not located on the disk. In the second case, an attempt to read a record which was beyond the end of file occurred.

RESPONSE: In the second case, the error occurred due to an invalid pointer in the index file. The most likely reason for this error is that the file is not closed properly on a previous update. Find the index. If it cannot be found, regenerate an index by selecting the reorganization program on the main menu.

86905 A file copy during reorganization failed.

CAUSE: A request was made to reorganize or regenerate a master file index using a disk drive other than the data file drive as the reorganization work drive. The work file ($$$.DAT or $$$.NDX) could not be located.

RESPONSE: Reorganize or regenerate the index using the data disk drive as the reorganization work drive.

86906 A file copy failed because of insufficient space on the data disk drive.

CAUSE: No space available on disk.

RESPONSE: Delete unnecessary files from data drive disk and attempt to reorganize or rebuild the index.

86980 A file copy failed because an index file was improperly copied.

CAUSE: System failure.

RESPONSE: Regenerate index file.

APPENDIX G

RESERVED VARIABLE NAMES

The following is a list of the variables which should not be used as variable names in files created by PEARL Level 3. If any of these variables are defined as file variables, errors may result in the generated programs.

-B-

BUFFERS.X

-C-

CC.ADD
CC.ADD.NDX
CC.AVAIL.DELETED
CC.BACKUP
CC.CHAIN.FROM$
CC.CHAIN.LIST$
CC.CHAIN.TO$
CC.CODE
CC.COMPANY.NAME$
CC.CON.LINES
CC.CON.WIDTH
CC.D
CC.DATA.DISK$
CC.DATE
CC.DATE$
CC.DATE.FORMAT%
CC.DELETED
CC.DELETED.COUNT
CC.DISK.SIZE
CC.DISK.SIZE%
CC.DRIVE$
CC.F
CC.F.D$
CC.F.M$
CC.FILE
CC.FILE.DRIVE$
CC.FORM.SIZE
CC.FORMS.CONTROL%
CC.INDX$
CC.INSTALL$
CC.JE%
CC.LINES%
CC.MAIN.PROG$
CC.MEMORY
CC.MS$
CC.MSG$
CC.MSGLEVEL%
CC.NAME$
CC.NDX
CC.NEXT.DELETED
CC.P.OR.C$
CC.PAGE%
CC.PAGE.LEN%
CC.PAUSE%
CC.PC%
CC.PRT
CC.READ
CC.READ.NDX
CC.RECORDS.P
CC.REPORT%
CC.REPORT.ID$
CC.REV1
CC.REV1$
CC.REV1%

CC.REV2
CC.REV2$
CC.REV2%
CC.REV3
CC.REV3$
CC.REV3%
CC.SCREEN.CLEAR$
CC.SERIAL.NO%
CC.SK.ADD
CC.SKEY%
CC.SORT%
CC.SORT.DRIVE$
CC.START%
CC.STATUS$
CC.SYS.DISK$
CC.SYSTEM$
CC.T%
CC.TAB%
CC.TAB1
CC.TAB2
CC.TERM.TYPE%
CC.TYPE$
CC.VK$
CC.X$
CC.XX$
CCREC1$
CCREC2$
CENTRYB.FLAG
CFLAG%
CONFIG.FLAG%

-D-

D.R.ID
D.R.ID.10200
D.R.ID.29600
DAY%
DESTINATION.FILE$
DESTINATION.FILE.DRIVE$
DESTINATION.FILE.NAME$

-E-

ENTRY.FLAG%

-F-

FALSE%
FLAG.74000%
FREE.SPACE
FUNCTION$

-H-

HCODE
HIGH$
HRL%

-I-

IS.A$
IS.BP$
IS.BUF%
IS.BUF.INFO$
IS.C$
IS.C%
IS.C.CLOSE
IS.C.DATE
IS.C.LAST%
IS.C.LAST.DATA
IS.C.LEN.DATA
IS.C.LEVELS%
IS.C.MAX.DATA
IS.C.OPEN
IS.C.ROOT%
IS.C.USER1
IS.C.USER2
IS.F.COUNT
IS.FILE%
IS.HI.CUTOFF$
IS.HIGH$
IS.ID%
IS.INFO.POS%
IS.JSU%
IS.KK$
IS.LL%
IS.LO.CUTOFF$
IS.LOW$
IS.LVL%
IS.NAME$
IS.NBUFS%

IS.NKEY$
IS.NODE%
IS.NODE.BUF$
IS.OFF.END%
IS.OPEN
IS.PTR$
IS.PTR%
IS.REC%
IS.RK$
IS.SLEN%
IS.SPLIT$
IS.SPLIT%
IS.TEMP.KEY$
IS.W%
IS.WRT%
IS.XP%

-J-

J.CTL

-K-

K82170%
KCODE
KEY$
KEY.CHAR

-L-

LENGTH.ERROR.MESSAGE$
LLEN%
LOW$

-M-

M.DRID
M.J%
M.KEY$
M.X$
MANY.SPACES$
MID.STRING$
MO.MAX
MONTH%
MONTH.MAX.FLAG
MSG$

-N-

NRECS%
NUMB.RECS

-O-

OPTION.CODE

-P-

PAUSE%
PCODE
PROGRAM$
PROGRAM.DISK$
PRT$

-R-

RCODE
RECORDS.PRINTED
REPORT.DEPTH%
REPORT.WIDTH%
RES1
RES2
RES3%
RES4%
RES5$
RES6$
RETURN$
RL%
ROOT73000%

-S-

SAVE.CORE
SAVED
SOURCE.FILE$
SOURCE.FILE.DRIVE$
SOURCE.FILE.NAME$
SOURCE.SIZE
SPACES$
SS%
START.30110
START.40110
START.41110
STRING2$
SYSTEM.ID$

-T-	-U-
T.DRID	UCODE
T300	
T400	-V-
T410	VER.DATE$
TC$	VER.NO$
TCODE	VF
TKEY$	
TM.K$	-X-
TRUE%	XP%
TYPE$	
TYPE.OPERATION	-Y-
	YEAR%

APPENDIX H

USER DEFINED DATA FILE STRUCTURE

The data file defined and processed by PEARL-generated programs consists of two record types.

1. THE CONTROL RECORDS

PEARL defines one control record. This control record is the first record on the data file. The user may define up to eight (8) additional control records for a total of nine (9). Each control record begins with an integer enclosed by two double quotes. The integer specified numbers the control record from "1" to "9". This numbering scheme is provided so that sorts may be performed on the file while maintaining the integrity of the control records sequence. This is done by first sorting on the second character on the record (the integer).

There are two control records generated for files designated as MASTER, TRANSACTION, and HISTORICAL transaction files. The first control record will contain single file maintenance control variables in addition to any variables added by the user. The second control record will contain variables used to control cross file processing and checks in addition to any variables added by the user.

The CBASIC code used to access a control record is as follows:

```
REM
REM      INPUT FILE CONTROL RECORD
REM
10050    READ #RM.FILE,1;
%INCLUDE B:RMCREC1-
        (RMCREC1.BAS) LV 00 AS OF 09/10/80** (PEARL (VERSION 3.01.00,
        SN CPU 1235))**
         CCREC$,            CONTROL RECORD COUNTER/ID
         REM.SORT           SORT CONTROL COUNTER
         RMC.LEN,           RECORD LENGTH
         RMC.LOW,           LOW RECORD KEY
         RMC.LAST,          LAST RECORD USED
         RM.UPDATE.FLAG,    UPDATE IN PROCESS FLAG
         RES1,        RESERVED FOR EXPANSION
         RES2,        RESERVED FOR EXPANSION
         RES3%,       RESERVED FOR EXPANSION
         RES4%,       RESERVED FOR EXPANSION
         RES5$,       RESERVED FOR EXPANSION
         RES6$,       RESERVED FOR EXPANSION
         RM.UPDATE.RESET   :REM OPEN ERROR RESET COUNTER
REM       ******** END OF RMCREC1.BAS **************
```

Figure 1

APPENDIX H

This record appears on the data file as follows:

"1",record length,low record key,last record used,update in process flag,,,,,,open error reset count

Access to control records is provided by the pr000 (file initializating program), and pr000A (control record update program) where pr = the unique file identifier (ID).

1.1 The Random Access File Control Record Variables.

CCREC1$	This variable holds the control record number. The maximum value is nine (9).
xxC.SORT	This variable is a sort flag. It is incremented by a value of 1 each time the data file is updated. The maximum value is (999) after which it will be reset to 1. The report program uses this variable to check the status of sorted files.
xxC.LOW	The first record number on a random access file.
xxC.LAST	The relative record number of the last record on file.
xx.UPDATE.FLAG	This variable is set to 1 everytime the data file is opened, and reset to zero when the data file is closed using the generated I/O routine.
xx.UPDATE.RESET	This variable is equal to the number of times xx.UPDATE.FLAG has been reset after an irregular file closure has been detected.

1.2 The Indexed File Control Record Variables.

CC.REC1$	Same as random file.
xxC.SORT	Same as random file.
xxC.LAST	Same as random file.
xx.UPDATE.FLAG	Same as random file.
xx.UPDATE.RESET	Same as random file.
xx.IS.BUFS%	This variable is used to set the number of buffers the index system is to use during the index key processing.
xx.NEXT.DELETED	This variable is equal to the relative record number of the last record deleted.

	Deleted records are used first if any new records are added to the file. Deleted records are linked.
xx.AVAIL.DELETED	This variable is equal to the number of deleted records on file.

1.3 The Master File Control Record Variables.

A master file will have the same variables as an indexed file with the addition of the following posting and closing control variables:

xx.S.POST.COUNT	This variable is equal to the number of transaction records posted during the last posting period.
xx.S.JE	This variable is equal to the last transaction record number posted during the last posting period.
xx.S.CLOSE.COUNT	This variable is equal to the number of transaction records closed during the last period-end closing process.
xx.SH.JE	This variable is equal to the last transaction record number processed during the last period-end closing process.

1.4 The Transaction File Control Record Variables.

A transaction file will have the same file control variables as a random access file with the addition of the following:

xx.S.POST.COUNT	See master file.
xx.S.JE	See master file.
xx.S.CLOSE.COUNT	See master file
xx.SH.JE	See master file.

1.5 The Historical Transaction File Control Record Variables.

A historical transaction file will have the same file control variables as a random access file with the addition of the following closing control variables.

xx.S.CLOSE.COUNT	Same as master file.
xx.SH.JE	Same as master file.

2. THE DATA RECORD

Each data record consists of a one-string field enclosed within two double quotes. The second character in the record (following the first double quote) is the letter A for active records, and the letter Z for deleted records. When sorting the file, first sort on the second character in the record. This facilitates the proper placement of control records at the start of the file followed by all active data records with deleted records being placed at the end of the file.

The user-defined data fields start at position 3 on the data record. Each data field extends for the length it was defined to be. No special characters separate one field from the next. The position, length, and data field type are calculated and coded into the generated data file I/O routines.

The first data field in the record specifies the relative data record numbers for a random access file or the record key for an indexed file.

The data record appears on the data file as follows:

```
"A string" SPACE CR LF
```

where "string" is the data content of the record.

Access to data records for editing and adding purposes is provided through pr100 programs (data record update).

APPENDIX I

USING THE SORT EXIT UTILITY (XSORTX)

PEARL Level 3 provides for the sorting of data files using the command program QSORT (a product of Structured Systems). Only non-indexed direct access file report routines generated by PEARL Level 3 will have the ability of sorting files. The main purpose of sorting files is to order data records in order by specific key fields. The QSORT program is not provided with the PEARL Level 3 package.

The following points will be covered in this appendix:

1. QSORT - the control information needed.
2. XSORTX - the interface.
3. How to set up XSORTX.x
4. How to formulate the sorted file name.
5. How to execute a sorted report.
6. A sequential description.
7. The variables involved.
8. The sort routines and subroutines.

1. QSORT

A sort generally involves two data files. One is the input file and is usually the original data file. The other is an output file and is the outcome of the sort process. In order for a file to be sorted, the following information must be provided:

1. The name of the input file. This includes:

   a. The disk drive (A, B, C, . . . L, M, . . .).

   b. The file name.

   c. The file type (DAT).

2. The name of the output file:

   a. The disk drive (A, B, C, . . . L, M, . . .).

   b. The file name (see Section 4).

   c. The file type (SOR).

3. The length of the data records on the file.

4. The position and length of four or less sort keys.

5. The name of the QSORT control file.

6. A sort work disk drive.

7. Whether disks are to be switched during the sort.

2. XSORTX (THE QSORT INTERFACE)

XSORTX is a program that maintains sort control information for various data files and facilitates the execution of sorts with a minimum of instruction while maintaining the program's execution environment. The XSORTX program may be executed from a generated main menu, a generated report writer program, or from CP/M.

The menu:

```
 0 -  RETURN TO CALLING PROGRAM ( )
 1 -  UPDATE SORT CONTROL RECORDS
 2 -  REPORT ON SORT CONTROL RECORDS
 3 -  SORT BY ITEM NUMBER ON TRANSACTIONS
 4 -
 .
 .
 .
12 -
```

2.1 XSORTX maintains a file called YSORTY.DAT on drive A. This file is used to:

1. Save the common area control variables prior to a sort. This is necessary because in order to execute QSORT, control is handed back to CP/M in which case the common variable area is lost.

2. Restore the common area variables after a sort.

3. Store each of the ten sort control sets on a separate data record.

2.2 When an option to sort is invoked, XSORTX will do the following:

1. Retrieve the appropriate sort control record.

2. Check to see if the correct configuration of data files and programs necessary for the sort are present.

3. Create a QSORT control file.

4. Create a submit file to execute QSORT, then rerun to XSORTX, when the sort is complete.

3. HOW TO SET UP XSORTX

Although XSORTX can be executed independently, it does so very cautiously, especially when it is executed for the first time. This is caused by its dependence on common area variables and the presence of its data file. When it is executed for the first time, it will prompt for the information that it needs clearly.

To execute XSORTX, do one of the following:

3.1 Enter: **RUN XSORTX**

3.2 When in the generated system main program, Enter: **=XSORTX**

3.3 When in a generated system report write routine, Enter: **1S**

All these prompts will get the user into XSORTX if it is located on the logged system drive.

The Entry of Control Data.

To enter the control data for any sort requires the user keep a PEARL Level 3 Data Element Report handy. This report will provide the information needed to add the QSORT input data file information.

The remainder of information needed is:

1. Output file drive (mandatory).
2. Output file name (see Section 4).
3. Output file type (SOR).
4. Sort work disk drive (mandatory).
5. Create a backup file (Y/N).
6. Change disks during sorts (Y/N).

1, 4, 5, and 6 are dependent upon the creativity of the user. 3 must be "SOR" for PEARL Level 3 report routines.

4. HOW TO FORMULATE THE SORT FILE NAME

The key to executing a sort from a PEARL Level 3 report writer is the formulation of the sorted file name. The sorted file name is a composite of three variables. These are:

1. The input data file name.
2. The report menu option.
3. The XSORTX menu option.

The formula (using the above line numbers):

OUTPUT FILE NAME = 1 + 2 + 3

If the input data file name is longer than five characters, then truncate it to five characters. (Trunacate right-most characters.)

<u>Example</u> <u>1:</u>

1. Input data file name = TRA.

2. The report menu option = 2.

3. The XSORTX menu option = 5.

The sorted output data file name = TRA25

<u>Example</u> <u>2:</u>

1. Input data file name = PAYROLL.

2. The report menu option = 1.

3. The XSORTX menu option = 3.

The sorted output data file name = PAYRO13

This convention was adopted to allow for the generation of multiple sort files for each report menu option. Every sort option will be validated against the output file name specified on the sort control record. This is done internally be XSORTX using the same formula as described in Section 4. If the validation fails, then the sort will not be executed.

A single report may process more than one sorted file. Each report menu option invokes a single report writer subroutine. Specification of two sort control records for a specific report menu option will allow a single report routine to process more than one sorted file.

5. HOW TO EXECUTE A SORTED FILE REPORT

To execute a sorted report, the user must first define the sort control data by entering XSORTX by any one of the conventions described in Section 3 of this appendix.

Given that the XSORTX menu is as follows:

```
0 - RETURN TO CALLING PROGRAM ( )
1 - UPDATE SORT CONTROL RECORDS
2 - REPORT ON SORT CONTROL RECORDS
3 - SORT BY DEPARTMENT AND CUSTOMER
4 - SORT BY INVENTORY NAME
```

And the report writer menu is:

```
0 - RETURN TO MAIN MENU
1 - REPORT ON TRANSACTIONS
2 - REPORT ON DEPARTMENT ACTIVITY
```

And you need to generate report option 1 using the original data file, then select option 1 (no sort).

If you need to generate report option 1 with the same data but sorted by department and customer, then ENTER: **1S3**

This entry informs the system of three things:

1. 1 generate report on transactions.

2. **S** sort the file.

3. **3** sort the file using sort option **3** on the XSORTX menu..

Note that if the data file name and type are **TRANSACT.DAT**, the sorted file name is **TRANS13.SOR**.

6. A SEQUENTIAL DESCRIPTION

The following is a sequential description of the sorted report process. The starting point is when the user is at the point when he wants to enter his report option.

1. Enter option,

2. If an S was entered following the report option, then go to 5,

3. Print report,

4. Return to menu,

5. If no sort option specified, then go to XSORTX,

6. Construct sort file name,

7. Open sorted file,

8. If sorted file is found and opened, then go to 10,

9. Prompt to confirm that no sorted file exists, then sort file,

10. Compare sorted file with original,

11. If identical, then go to 3 (no need to sort),

12. If not identical, then go sort the file,

13. Go to 3.

7. THE VARIABLES

CC.REPORT%	This variable will be equal to the report writer menu option chosen by the operator. It is used to formulate the sorted file name. It is defined in the common area, thus, the value content is transferred to XSORTX.
CC.SORT%	This variable will be equal to the sort option. It is initially set to zero and is only updated when the user specifies a sort, and sort option. It is defined in the common area, and its value is transferred to XSORTX.
xxC.SORT	xx is the file prefix. This variable is defined on the file control record. This variable is used to validate if an existent sorted file is identical to the current data file to be sorted. The value of xxC.SORT is incremented by a value of 1 each time the data file is updated. A sort will be executed only if the value of xxC.SORT on the data and sorted file are not equal.
CC.SORT.DRIVE$	This variable is defined in the common area and it indicates the disk drive on which the sorted file is to be located. It is updated from the particular sort control record processed. If it is equal to a null, then the current data file drive designation is assumed.
Sort Control VARIABLES	Consult Appendix C.
Submit File VARIABLES	Consult Appendix C.

8. THE SORT ROUTINES AND SUBROUTINES

Random access file routines with sort supporting report writers will process sorted files. The following are routines and subroutines supporting the sorting of data files.

700 Where xx is the prefix identifier for the file. This subroutine is located in the report writer main program. It will interpret the operator report entry. If a sort was requested, then a check for

the correct sort file on disk will be made, and if a sort is required, then XSORTX will be chained to.

XSORTX	This routine will do the following: a. Save the common area variables. b. Validate the sort options requested by operator. c. Create QSORT sort control file. d. Create a submit file to execute QSORT, then return control to XSORTX. e. Restore common area variables. f. Return control to calling report routine. g. Maintain sort control records (maximum of 10). h. List QSORT control records.
C82400A	This subroutine is included in XSORTX.BAS and will create a QSORT control file.
C82500A	This subroutine is included in XSORTX.BAS and will create a submit file ($$$.SUB) for execution of a batch process upon exit to CP/M.

XSORTX is designed to create submit files for CP/M Operating Systems. In order for such submits to be executed under other operating systems, the source code for the XSORTX.BAS and complementary files must be modified to support these systems. XSORTX includes the following modules:

1.	XSORTX.BAS	Main Program
2.	SR10000.BAS	I/O Routine
3.	C82000E	Console Input
4.	C82400A	Generate QSORT Control File
5.	C82500A	Generate a Submit File

APPENDIX J

PROGRAM LIST UTILITY (LIST)

The program list utility may be used to list CBASIC programs. Up to four levels of INCLUDED modules may be specified. Each line on the printed output indicates the line number from the included module, as well as the line number in the program as it will be compiled. An option is provided to suppress the listing of included modules.

The list utility program may be executed by entering the command:

RUN LIST

When the program is loaded, the operator will be prompted to enter the current date. The date is entered in the format mm/dd/yy. Delimiters may be used to separate the month, day and year if desired.

NOTE:

When this program is loaded, a check will be made to determine if the SYSTEM.DDD file is present on Disk A. If it is, the system configuration control data (see appendix B) will be loaded into memory. In this case the alternate date formatting control may be used.

The following menu will then be displayed to the operator:

```
0     TERMINATE PROCESSING
1     RESET PAGE LENGTH [ 60 ]
2     RESET FORM LENGTH [ 0 ]
3     RESET LINE LENGTH [ 109 ]
4     NEW TITLE [ ]
5     TO LIST THE TABLE OF CONTENTS FOR n PROGRAMS
6     TO CHANGE FILE TYPE [.BAS]
7     CONTINE WHEN INCLUDED MODULE NOT FOUND [N]
8     HONOR SUPPRESS PRINT [Y]
9     PROCESS INCLUDED FILES [Y]
10    LIST SPECIFIED PROGRAMS
```

1. RESET PAGE LENGTH

This is the number of lines which are printed in the body of each page, and does not include the interpage gap.

2. RESET FORM LENGTH

This is the number of lines on each page including the interpage gap. If a zero is specified, the program assumes that the standard form feed character should be used to space to the top of form.

3. **RESET LINE LENGTH**

The line length is the maximum length of an output line. If a line is longer than the specified length, it will be truncated, and a "T" will be placed in the margin immediately preceeding the line of source code.

4. **NEW TITLE**

If a title is entered, it will be included in the header line for each page which is printed.

5. **TO LIST TABLE OF CONTENTS FOR n PROGRAMS**

When a set of programs are listed, two page numbers are printed on each page. The first number is the page within the program, the other is the page number within the entire set of listings. A running table of contents is maintained for all programs listed until this option is selected.

If a -5 option is selected, or if the current table of contents is listed, the current table of contents will be cleared, and the program count will be reset to zero.

The line which will appear on in the table of contents for each program will either be the first line in the program, or if the control TITLE=(description) is encountered within the first four lines in a program, that description which is found within the parenthesis will be used.

6. **TO CHANGE FILE TYPE**

This program may be used to list any type files. However, if includes are used, the program will expect that the type of the included files is the same as the type indicator on the main file.

7. **CONTINUE WHEN INCLUDED MODULE NOT FOUND**

If an "N" is specified here, processing will terminate if an included file cannot be located. If "Y" is specified, then a message will be displayed on the terminal when an included module is not found, but listing of the rest of the file will continue.

8. **HONOR SUPPRESS PRINT**

If a "Y" is specified, then those included files which have the phrase "SUPPRESS" in the %INCLUDE statement will not be printed. If an "N" is specified, then all included files will be listed.

9. PROCESS INCLUDED FILES

If an "N" is specified, then no included files will be included in the listing. If "Y" is specified, then all included files will be included in the listing.

10. LIST SPECIFIED PROGRAMS

The operator may enter either a list of files, a single file, or may specify the name of a file with a file type of (.LST) which contains a list of the files to be processed. For example:

```
A:LIST
```

Would cause the program LIST.BAS on disk A to be listed.

```
A:LIST,B:PEARL3,B:A000
```

Would cause the three programs specified to be listed.

```
=PROG.LST
```

Would cause the file PROG.LST to be loaded to obtain the list of files to be listed. Each record on this file would then contain the name of one program to be listed.

APPENDIX K

REPORTING PROGRAM DEFECTS TO CPU

Before being released for distribution, the PEARL programs have been tested and used for a period of time. During these months, a number of logic defects in the programs have been detected and fixed. However, due to the complexity of the programs, and the great number of combinations which can be processed by the system, there is always the possibility that additional defects in the programs may be uncovered as the programs are used by more and more people.

If you detect an error in the logic of the PEARL programs themselves, or in any of the Generated Programs, we would appreciate hearing from you. If you can provide us with enough information to duplicate the error, we will then be able to correct that error.

Please use the following procedures to report program defects to CPU.

1. Make sure that the defect can be duplicated. If it cannot be duplicated, chances are that the error is a result of a worn out diskette, improper operating procedures, or a bug in either CBASIC or the CP/M operating system.

2. Prepare a hard copy of all of the input provided by the user which caused the error to occur. This should include a listing of the control file, a description of the sequence in which data was entered into the system, and a description of the error which occurred. If possible, generate a hard copy of the error when it occurs with the MESSAGE LEVEL set to a value of 2.

3. Send CPU the serial number of the program being used at the time the error occurred, the version number of the program, the dealer from whom the program diskette was purchased, and the date of purchase.

In return for providing the above information, CPU will provide to the end user the following:

1. For all customers who send in documentation on a program defect in PEARL, a program diskette will be provided containing the corrected version of the program.

2. All registered owners of PEARL will receive a newsletter summarizing all program defects found and corrected. Each end-user may obtain an updated version of the PEARL programs by returning his/her original program diskette directly to CPU.

APPENDIX L

INDEXED FILE REORGANIZATION

Indexed file reorganization is a process to protect and increase the efficiency and integrity of data files that are in use. There are two steps to file reorganization:

1. Generation of a new index for a data file.

2. Reorganization of the data file.

You may also define alternate file drives as reorganization work drives in the event that there is insufficient space for the regeneration of the reorganization process on the defined data file disk drive.

The following is a brief discussion of each process.

<u>The Example:</u>

An indexed file was generated by PEARL Level 3.
The file description is "MASTER File".
The file name is "MASTER.DAT".
The generated index file is "MASTER.NDX".

1. THE GENERATION OF NEW DATA FILE INDEX.

The main advantage to using the file reorganization is to recover a data file in the event of a system failure. System failures are critical for indexed files because of the synchronous relationship existing between the index and the data files generated by PEARL Level 3.

1.1 The Menu.

1(D) Generation New Master File Index

where D = alternate disk drive to be used.

The entry of (1) will generate a new index file using the data file drive as a reorganization work drive.

The entry of (1A) will specify the alternate drive A as the reorganization work drive on which a formatted drive may be placed.

1.2 The Process.

The steps are taken as follows:

1. Delete the old index as present on the data drive.

2. Create a new index file, $$$.NDX, on the file drive or the operator-specified alternate reorganization work drive.

3. Load the new index with the data file keys.

4. Rename the work index $$$.NDX to MASTER.NDX.

2. THE REORGANIZATION OF THE DATA FILE.

The process is useful in the event that the data file has an unusually large number of unused or deleted records. Reorganization helps recover used, and therefore lost, disk space.

2.1 The Menu Option.

2(D) Reorganize the MASTER file and index.

where (D) = alternate disk drive to be used.

The entry of 2 will cause the creation of a new data file using the original data file and index, and the subsequent generation of a new index for the reorganization data file. Data file errors will also be corrected if detected.

The entry of 2A will cause the reorganization process to use drive A as a work drive.

2.2 The Process.

The steps are taken as follows:

1. A reorganization data file will be created as $$$.DAT.

2. The work file will be loaded with all active data records from MASTER.DAT in order by primary key.

3. Delete MASTER.DAT and MASTER.NDX.

4. Rename $$$.DAT to MASTER.DAT.

5. Generate a new index for MASTER.DAT.

2.3 Reorganization Error Codes.

ERROR	DESCRIPTION
29080	The (input) data file during reorganization was in error.*
29085	The attempt to rename the work file $$$.DAT to the data file name failed.

29180	The data file control record control variables did not evaluate the values expected after a new index was created.*
29181	The control record on the data records is in error as to the number of deleted records on the file.*
29185	The attempt to rename the work output index file ($$$.NDX) to the data file index name failed.
29200	The data file was not closed properly the last time it was opened.*
29600	An attempt was made to reorganize a data file with an index file that was no good. Construct a new index first, then reorganize the data file.*

*Errors that are corrected by the system upon action by the operator.

Example of How to Reorganize File Described in Chapter 4.

```
          CUSTOMER CONTACT FILE(A:CONTACT) VERSION 1.0
                  MAIN SELECTION MENU-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)

0.  RETURN TO CP/M
1.  INITIALIZE CUSTOMER CONTACT FILE
2.  EDIT CUSTOMER CONTACT CONTROL FILE
3.  UPDATE CUSTOMER CONTACT FILE
4.  GENERATE CUSTOMER CONTACT REPORTS
5.  COMPRESS CUSTOMER CONTACT FILE
6.  CHANGE SYSTEM DATE
7.  EDIT SYSTEM CONFIGURATION DATA
ENTER DESIRED FUNCTION BY NUMBER:
```

User Enters:

    5

System Responds:

```
LOADING PROGRAM TO COMPRESS CUSTOMER CONTACT FILE

          CUSTOMER CONTRACT FILE(A:CF000R) VERSION 1.0
            CUSTOMER CONTACT REORGANIZATION-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)

0    TO RETURN TO MAIN MENU
1D   GENERATE NEW CUSTOMER CONTACT INDEX
2D   REORGANIZE CUSTOMER CONTACT AND INDEX
?
```

<u>Generating New Index and Data File Using Default Data File Drive</u>

User Enters:

1 (where D = default data file drive. In this case "B", so no drive need be specified)

System Responds:

```
GENERATING A NEW CUSTOMER CONTACT INDEX
```

This is the only message that will be returned. The reorganization menu will again be displayed when reorganization is complete.

This is an example of the system response if you had selected option 2 rather than option 1:

```
               CUSTOMER CONTRACT FILE(A:CF000R) VERSION 1.0
                 CUSTOMER CONTACT REORGANIZATION-MM/DD/YY
                        MINIMUM FREE SPACE=(NNNNN)

0     TO RETURN TO MAIN MENU
1D    GENERATE NEW CUSTOMER CONTACT INDEX
2D    REORGANIZE CUSTOMER CONTACT AND INDEX
?
```

User Enters:

```
     2
```

System Responds:

```
REORGANIZING CUSTOMER CONTACT AND INDEX
GENERATING A NEW CUSTOMER CONTACT INDEX
```

Option 1 would not need to be selected if option 2 had been chosen.

Generating New Index and Reorganizing Data File Using Drive A:

Note that if you wish to use a different data file drive other than the default, you must set the number of buffers on the file control record to 0. This can be done by selection of option 2, EDIT CUSTOMER CONTACT CONTROL DATA, on the Main Selection Menu.

```
          CUSTOMER CONTRACT FILE(A:CF000R) VERSION 1.0
            CUSTOMER CONTACT REORGANIZATION-MM/DD/YY
                   MINIMUM FREE SPACE=(NNNNN)

0     TO RETURN TO MAIN MENU
1D    GENERATE NEW CUSTOMER CONTACT INDEX
2D    REORGANIZE CUSTOMER CONTACT AND INDEX
?
```

User Enters:

```
     1A (where A = default data drive)
```

System Responds:

```
GENERATING A NEW CUSTOMER CONTACT INDEX
B:CUTOMER.NDX IS A COPY OF A:$$$.NDX
```

If 2A had been selected:

System Responds:

```
REORGANIZING CUSTOMER CONTACT AND INDEX
B:CUTOMER.DAT IS A COPY OF A:$$$.DAT
GENERATING NEW CUSTOMER CONTACT INDEX
B:CUTOMER.NDX IS A COPY OF A:$$$.NDX
```

Generating New Index File and Recovering from Errors on Data Files

```
            CUSTOMER CONTRACT FILE(A:CF000R) VERSION 1.0
              CUSTOMER CONTACT REORGANIZATION-MM/DD/YY
                     MINIMUM FREE SPACE=(NNNNN)

0    TO RETURN TO MAIN MENU
1D   GENERATE NEW CUSTOMER CONTACT INDEX
2D   REORGANIZE CUSTOMER CONTACT AND INDEX
?
```

User Enters:

```
     1
```

System Responds:

```
GENERATING A NEW CUSTOMER CONTACT INDEX
ERROR [ 29180 ] IS DETECTED
ENTER [ IGNORE ] TO CONTINUE WITH REORGANIZATION
?
```

User Enters:

```
     IGNORE
```

ADDENDUM TO PEARL LEVEL 3 MANUAL

version 3.02.00

February 23, 1981

Section 3.1, Distribution Diskettes

Paragraph 3, Common Source Utilities

The following program modules are deleted:

C82400A.BAS	CREATE QSORT CONTROL FILE
XSORTX.BAS	SORT EXIT UTILITY MAINLINE
SR42000.BAS	DISPLAY EDIT FOR SORT CONTROL DATA
COM01E.BAS	COMMON VARIABLES

The following program modules have been added:

COM01F.BAS	COMMON VARIABLES
C82400AQ.BAS	CREATE QSORT CONTROL FILE
C82400AU.BAS	CREATE ULTRASORT CONTROL FILE
QSORTX.BAS	SORT EXIT UTILITY MAINLINE (QSORT)
USORTX.BAS	SORT EXIT UTILITY MAINLINE (ULTRASORT)
SR42000Q.BAS	DISPLAY EDIT FOR SORT CONTROL DATA
SR42000U.BAS	DISPLAY EDIT FOR SORT CONTROL DATA

Section 3.3.2, Using Single Density 8-inch Floppy Diskettes.

The first two paragraphs of the NOTES: at the end of this section should be replaced with the following instructions:

Two versions of the SORT exit control program are provided. QSORTX supports use of QSORT to sort data files, while USORTX supports use of ULTRASORT. Depending upon which sort program is being used, the appropriate program should be placed on the APPLICATION system diskette using the name XSORTX. For example, if ULTRASORT is being used, use the USORTX.BAS program to control the sort interface. If QSORT is being used, use the QSORTX.BAS program to control the sort interface.

(Refer to section 3 in Appendix I for a description of how this program is prepared initially for execution.)

The entire source for both of these programs is provided on the distribution diskettes. If you wish to modify the source and to recompile, the appropriate source modules will be included to interface with either QSORT or ULTRASORT during the generation of reports which require sorted files.

APPENDIX E

Add the following new error messages.

120 (W). INVALID FILE (nn).

CAUSE: The file number which was specified in the report control record as the primary file is not the same file number which is currently being processed. The nn value in the message indicates the primary file number specified in the report control record.

RESPONSE: Return to the System Generation Control menu and reset the file id to the file number specified as the primary file for the report subroutine being created.

APPENDIX I

Appendix I has been replaced.

APPENDIX I

USING THE SORT EXIT UTILITY (XSORTX)

PEARL Level 3 provides for the sorting of data files using the command program QSORT (a product of Structured Systems), or Ultrasort-II (a product of Computer Control Systems, Inc.). Only non-indexed direct access file report routines generated by PEARL Level 3 will have the ability of sorting files. The main purpose of sorting files is to order data records in order by specific key fields.

The following points will be covered in this appendix:

1. Sort control information required.
2. XSORTX - the interface.
3. How to set up XSORTX.
4. How to formulate the sorted file name.
5. How to execute a sorted report.
6. A sequential description.
7. The variables involved.
8. The sort routines and subroutines.

1. Sort Control Information

A sort generally involves two data files. One is the input file and is usually the original data file. The other is an output file and is the outcome of the sort process. In order for a file to be sorted, the following information must be provided:

1. The name of the input file. This includes:

   a. The disk drive (A, B, C, . . . L, M, . . .).
   b. The file name.
   c. The file type (DAT).

2. The name of the output file:

   a. The disk drive (A, B, C, . . . L, M, . . .).
   b. The file name (see Section 4).
   c. The file type (SOR).

3. The length of the data records on the file.

4. The position and length of four or less sort keys.

5. The name of the sort control file.

6. A sort work disk drive.

7. Whether disks are to be switched during the sort.

2. XSORTX (THE SORT INTERFACE)

XSORTX is a program that maintains sort control information for various data files and facilitates the execution of sorts with a minimum of instruction while maintaining the program's execution environment. The XSORTX program may be executed from a generated main menu or a generated report writer program, or from CP/M.

The menu:

```
 0 -  RETURN TO CALLING PROGRAM ( )
 1 -  UPDATE SORT CONTROL RECORDS
 2 -  REPORT ON SORT CONTROL RECORDS
 3 -  SORT BY ITEM NUMBER ON TRANSACTIONS
 4 -
 .
 .
 .
12 -
```

2.1 XSORTX maintains a file called YSORTY.DAT on drive A. This file is used to:

1. Save the common area control variables prior to a sort. This is necessary because in order to execute QSORT or USORT2M, control is handed back to CP/M in which case the common variable area is lost.
2. Restore the common area variables after a sort.
3. Store each of the ten sort control sets on a separate data record.

2.2 When an option to sort is invoked, XSORTX will do the following:

1. Retrieve the appropriate sort control record.
2. Check to see if the correct configuration of data files and programs necessary for the sort are present.
3. Create a sort control file.
4. Create a submit file to execute the sort, then return to XSORTX, when the sort is complete.

3. HOW TO SET UP XSORTX

Prior to using the XSORTX program, you must compile either the QSORTX (QSORT interface) or USORTX (USORT2M interface) program. These two programs are provided in source form only. Once compiled, the .INT file should be renamed to XSORTX.INT. In order to compile the program, the source files provided on

2. XSORTX [illegible]

XSORTX [illegible] sort package. [illegible] for [illegible] definition [illegible] will [illegible] the output [illegible] program may [illegible] report [illegible]

[illegible]

[illegible]

3.1 XSORTX [illegible] XSORTX.INT [illegible]

1. [illegible] This is necessary in order to execute QSORT or [illegible] CP/M in which [illegible] the [illegible] available area [illegible]

2. Reset [illegible] area [illegible] files [illegible] sort.

3. [illegible] data [illegible] separate [illegible]

3.2 When an [illegible] will be allowed:

1. [illegible] the appropriate [illegible]

2. [illegible] correct [illegible] data files and parameter [illegible]

3. [illegible] file.

4. [illegible] XSORTX, [illegible] complete.

4. HOW TO SET UP XSORTX

[illegible] QUBIX [illegible] either the QSORTX [illegible] interface [illegible] interface) program. These [illegible] programs are provided in source form only. Once [illegible] file should be renamed to XSORTX.INT. [illegible] control file [illegible] on the COMMON SOURCE UTILITIES diskette should be placed on drive A. Then use the following command sequence:

```
A>CBAS2 USORTX $B
CBASIC COMPILER VER 2.07
NO ERRORS DETECTED
CONSTANT AREA:       150
CODE SIZE:         20000
DATA STMT AREA:        0
VARIABLE AREA:      3000
A>REN XSORTX.INT=USORTX.INT
```

After this step has been completed, the XSORTX.INT file may be placed on a generated program system diskette along with USORTM2.COM (or QSORT.COM) if QSORT is being used as the sort program.

The first time that XSORTX is executed, the control file YSORTY.DAT is created on the system diskette. XSORTX may be executed either from the main menu of a generated application, or from a report program which generates a report using a sorted file. When XSORTX is executed for the first time, it will prompt for the information that it needs clearly.

To execute XSORTX, do one of the following:

3.1 When in the generated system main program, Enter: **=XSORTX**

3.2 When in a generated system report write routine, Enter: **1S**

All these prompts will get the user into XSORTX if it is located on the logged system drive.

<u>The Entry of Control Data.</u>

To enter the control data for any sort requires the user keep a PEARL Level 3 Data Element Report handy. This report will provide the information needed to add the QSORT input data file information.

The remainder of information needed is:

1. Output file drive (mandatory).

2. Output file name (see Section 4).

3. Output file type (SOR).

4. Sort work disk drive (mandatory).

5a. Create a backup file (Y/N). (QSORT version)

5b. Change sort work disk (Y/N). (ULTRASORT-II version)

6. Change disks during sorts (Y/N).

1, 4, 5, and 6 are dependent upon the creativity of the user. 3 must be "SOR" for PEARL Level 3 report routines.

4. HOW TO FORMULATE THE SORT FILE NAME

The key to executing a sort from a PEARL Level 3 report writer is the formulation of the sorted file name. The sorted file name is a composite of three variables. These are:

1. The input data file name.
2. The report menu option.
3. The XSORTX menu option.

The formula (using the above line numbers):

OUTPUT FILE NAME = 1 + 2 + 3

If the input data file name is longer than five characters, then truncate it to five characters. (Trunacate right-most characters.)

<u>Example 1:</u>

1. Input data file name = TRA.
2. The report menu option = 2.
3. The XSORTX menu option = 5.

The sorted output data file name = TRA25

<u>Example 2:</u>

1. Input data file name = PAYROLL.
2. The report menu option = 1.
3. The XSORTX menu option = 3.

The sorted output data file name = PAYRO13

This convention was adopted to allow for the generation of multiple sort files for each report menu option. Every sort option will be validated against the output file name specified on the sort control record. This is done internally be XSORTX using the same formula as described in Section 4. If the validation fails, then the sort will not be executed.

A single report may process more than one sorted file. Each report menu option invokes a single report writer subroutine. Specification of two sort control records for a specific report menu option will allow a single report routine to process more than one sorted file.

5. HOW TO EXECUTE A SORTED FILE REPORT

To execute a sorted report, the user must first define the sort control data by entering XSORTX by any one of the conventions described in Section 3 of this appendix.

Given that the XSORTX menu is as follows:

```
0 - RETURN TO CALLING PROGRAM ( )
1 - UPDATE SORT CONTROL RECORDS
2 - REPORT ON SORT CONTROL RECORDS
3 - SORT BY DEPARTMENT AND CUSTOMER
4 - SORT BY INVENTORY NAME
```

And the report writer menu is:

```
0 - RETURN TO MAIN MENU
1 - REPORT ON TRANSACTIONS
2 - REPORT ON DEPARTMENT ACTIVITY
```

And you need to generate report option 1 using the original data file, then select option **1** (no sort).

If you need to generate report option 1 with the same data but sorted by department and customer, then ENTER: **1S3**

This entry informs the system of three things:

1. **1** generate report on transactions.

2. **S** sort the file.

3. **3** sort the file using sort option **3** on the XSORTX menu.

Note that if the data file name and type are **TRANSACT.DAT,** the sorted file name is **TRANS13.SOR.**

6. A SEQUENTIAL DESCRIPTION

The following is a sequential description of the sorted report process. The starting point is when the user is at the point when he wants to enter his report option.

1. Enter option,

2. If an S were entered following the report option, then go to 5,

3. Print report,

4. Return to menu,
5. If no sort option specified, then go to XSORTX,
6. Construct sort file name,
7. Open sorted file,
8. If sorted file is found and opened, then go to 10,
9. Prompt to confirm that no sorted file exists, then sort file,
10. Compare sorted file with original,
11. If identical, then go to 3 (no need to sort),
12. If not identical, then go sort the file,
13. Go to 3.

7. THE VARIABLES

CC.REPORT%	This variable will be equal to the report writer menu option chosen by the operator. It is used to formulate the sorted file name. It is defined in the common area, thus, the value content is transferred to XSORTX.
CC.SORT%	This variable will be equal to the sort option. It is initially set to zero and is only updated when the user specifies a sort, and sort option. It is defined in the common area, and its value is transferred to XSORTX.
xxC.SORT	xx is the file prefix. This variable is defined on the file control record. This variable is used to validate if an existent sorted file is identical to the current data file to be sorted. The value of xxC.SORT is incremented by a value of 1 each time the data file is updated. A sort will be executed only if the value of xxC.SORT on the data and sorted file are not equal.
CC.SORT.DRIVE$	This variable is defined in the common area and it indicates the disk drive on which the sorted file is to be located. It is updated from the particular sort control record processed. If it is equal to a null, then the current data file drive designation is assumed.

Sort Control VARIABLES	Consult Appendix C.
Submit File VARIABLES	Consult Appendix C.

8. THE SORT ROUTINES AND SUBROUTINES

Random access file routines with sort supporting report writers will process sorted files. The following are routines and subroutines supporting the sorting of data files:

700 Where xx is the prefix identifier for the file. This subroutine is located in the report writer main program. It will interpret the operator report entry. If a sort was requested, then a check for the correct sort file on disk will be made, and if a sort is required, then XSORTX will be chained to.

XSORTX This routine will do the following:

a. Save the common area variables.

b. Validate the sort options requested by operator.

c. Create sort control file.

d. Create a submit file to execute QSORT or USORT2M, then return control to XSORTX.

e. Restore common area variables.

f. Return control to calling report routine.

g. Maintain sort control records (maximum of 10).

h. List sort control records.

C82400Ax This subroutine is included in XSORTX.BAS and will create sort control file. (If the control file is being created from QSORT, a suffix of Q will be used for this source file. If the control file is being created for ULTRASORT, a suffix of U will be used for this source file.)

C82500A This subroutine is included in XSORTX.BAS and will create a submit file ($$$.SUB) for execution of a batch process upon exit to CP/M.

XSORTX is designed to create submit files for CP/M Operating Systems. In order for such submits to be executed under

other operating systems, the source code for the XSORTX.BAS and complementary files must be modified to support these systems. XSORTX includes the following modules:

1.	XSORTX.BAS	Main Program
2.	SR10000.BAS	I/O Routine
3.	SR42000x.BAS	Input/edit sort control parameters
4.	C82000E.BAS	Console Input
5.	C82400Ax.BAS	Generate QSORT Control File
6.	C82500A.BAS	Generate a Submit File

PEARL LEVEL 3, VERSION 3.02.00

Summary of Modifications

February 23, 1981

The following is a summary of the modifications which have been made to PEARL Level 3, version 3.01.00 and are included in the version 3.02.00 release.

1. Addition of ULTRASORT support. ULTRASORT-II is a product of Computer Control Systems, Inc, Largo Florida. This sort program may be purchased from Computer Pathways Unlimited directly or through your local computer dealer.

   The sort program is invoked automatically during report generation. Processing control is maintained in the same control file that is used when QSORT is invoked. The modification to the end user is transparent. Applications which have been generated with version 3.01.00 of PEARL may use this capability by replacing one INT file which is provided with the system and replacing the QSORT.COM file with USORTM2.COM.

2. An incompatibility with version 2.07 of CBASIC has been corrected. Both PEARL3 and PEARL3-generated programs will execute using version CBASIC 2.07. (The new release has been compiled with 2.07, however, and may not be compatible with earlier releases of CBASIC.)

3. The system disk drive may be specified when the PEARL3 program is loaded, or when the main menu program of a generated application is generated. G:RUN G:PEARL3 G, for example, indicates that drive G is the system drive.

4. In generated applications, the period-end closing program will allow an historical transaction file to be spread across multiple diskettes.

5. An undefined label error which occurs during the compilation of a generated application which defines the I/O base for an indexed file with a base other than 10000 has been corrected. (This error did not occur unless secondary keys were defined on the file.)

6. An error which occurs in applications where a string variable is defined as a secondary key in an indexed file has been corrected. (The error results when the data item was entered as a blank and subsequently changed to nonblank (or visa versa) during editing of data on the file.)

7. An error has been corrected which causes the data fields in the control records to be generated incorrectly if the file

PEARL REVISION VERSION 2.01.00

Summary of Modifications [illegible]

[illegible]

The following [illegible] of the [illegible] which have been made to PEARL [illegible] version [illegible] are included in the version [illegible]

1. Addition [illegible] Commodore [illegible] program [illegible] directly [illegible] Pearl [illegible]

   The [illegible] language [illegible] correctly [illegible] control [illegible] the same [illegible] modifications [illegible] which [illegible] or PEARL may use [illegible] file which is provided [illegible] the [illegible] file with [illegible]

2. An incompatibility [illegible] has been corrected [illegible] programs will execute [illegible] has been changed [illegible] compatibility with earlier [illegible]

3. The system [illegible] PEARL program [illegible] program [illegible] general [illegible] example [illegible]

4. In [illegible] will [illegible]

5. [illegible]

6. [illegible]

[illegible]

type has been changed from indexed file to a direct access file (in the file defintion control data).

8. An error has been corrected that causes the source file for a report subroutine to be improperly named when the routine is created independently rather than as part of a complete system generation.

9. An error characterized by frequent pauses during report generation in the generated report writers has been corrected. This error occurs only when the report is being generated for an indexed file, and when secondary keys are defined on the file.

10. COM01E.BAS has been renamed COM01F.BAS. The code in the COM01F.BAS routine circumvents an SS error which occurs using CBASIC version 2.07 when reports are generated.