01 Apr 1988

Micro Database Management System Language

Karen Yingling Tam

George Winston Zobrist
Missouri University of Science and Technology

MICRO DATABASE MANAGEMENT SYSTEM LANGUAGE

K. Y. Tam* and G. W. Zobrist

CSc-88-3

Department of Computer Science

University of Missouri-Rolla

Rolla, Missouri 65401 (314)341-4491

*This report is substantially the M.S. thesis of the first author, completed April, 1988.

ABSTRACT

There are two approaches to solve computational problems in a microcomputer environment:

1. Non-database approach: uses a high level programming language with non-database files as input and/or output files.
2. Database approach: uses the programming language embedded in the micro Data Base Management System(DBMS), with the database defined by the integrated database definition language as input and/or ouput files.

Adopting the appropriate approach in any single application may save cost and time. This paper compares the two different approaches while solving the same Control Section (CSECT) Interaction Hierarchy problem and suggests which to use when.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF ILLUSTRATIONS

I. INTRODUCTION

As microcomputer technology continues to improve and is widely accepted by users, management of data in a microcomputer environment has become an important data processing subject. In responding to this data management need, micro Data Base Management Systems(DBMS) have been developed and available since early 1980's.

Surveys show that the most popular Micro DBMS today is dBASE III PLUS[1]. It provides the basic DBMS features such as data independence, central control of data, reduced redundancy, and some degree of data integrity. In addition, it also offers a self-sufficient high level programming command language. By incorporating the related data base management functions of dBASE III PLUS, this command language provides microcomputer end users with a very powerful programming tool.

The primary objective of this study is to investigate the capabilities of today's micro DBMS command language by implementing a Control Section(CSECT) Interaction Hierarchy Report project using dBASE III PLUS. In order to achieve the above objective, this paper first presents an overview of the background and features of non-DBMS high level languages and one of the most popular micro DBMS command languages - DBASE III PLUS.

The purpose of the CSECT Interaction Hierarchy Report is to assist software maintenance programmers with their planning and implementation efforts. Details of the dBASE III PLUS methodology for this CSECT project is presented in the following section. A comparison of the dBASE III PLUS and PL/I approaches for this same project is made to demonstrate the benefits of using a micro DBMS command language instead of a traditional high level language.

II. HIGH LEVEL PROGRAMMING LANGUAGES

The hardware capabilities of computers have grown tremendously during the past two decades. These developments have been roughly paralleled by progress in programming languages. The benefits from improved hardware technology cannot be fully utilized unless complementary languages are developed to increase the usefulness of advanced computers.

A. DEVELOPMENT OF PROGRAMMING LANGUAGES

The programming language generations are grouped chronologically and are also categorized by their levels which are defined by their distance from machine languages. Machine language is the first generation of the programming language. It came with the very early commercial computers in the early 1950's. When using machine language, the programmer must keep track of actual numerical addresses of storage locations for instructions and data. The coding of the program has to be at the 0's and 1's level, which makes it very difficult to read and maintain.

The next generation language developed was assembly language. With assembly language the programmer uses symbolic names, or mnemonics, to specify machine operations. There is a one-to-one correspondence between machine language instruction and assembly language instruction. As with machine language, it can be used to develop programs which are highly efficient in terms of storage space and processing time. It also allows the programmer to more fully utilize the computer's potential.

Despite the improvements over machine language, assembly language is still difficult to use. It requires a high level of skill to be used effectively. A considerable effort is required in order to learn assembly instructions, and the language demands many instructions to perform a modicum of processing.

High level languages are the third generation of programming languages. These have been developed for people interface, whereas low level languages are oriented to the computer. The instruction syntax adopted in high level languages is close to English. Instructions written in a high level language must be translated into machine language to be used by the computer. This makes it easier for programmers to express what they want the computer to do without having to directly specify how the machine instructions should be assembled to do it.

High level languages have been procedure-oriented and are largely divided between business and scientific. COBOL by Codasyl, is a commercial business-oriented language. FORTRAN and ALGOL are examples of scientific programming languages. However, PL/1 incorporates most of the features found in COBOL and FORTRAN.

Statistics developed by Microelectronics and Computer Technology Corporation of Austin, Texas, show that about 80 percent of the programs in a computer facility use 2 percent of the machine cycles[2]. About 2 percent of the programs use 50 percent of the machine cycles. The remaining 18 percent use 48 percent of the machine cycles. Programming productivity is the issue in the 80 percent group. These programs are the targeted for a new class of programming language - the fourth generation languages (4GL).

While similar to third generation languages, 4GL's are different in that the number of programmed instructions required to get information is typically much less. 4GL's are often referred to as "very high level" languages since they exhibit the highest level of machine independence. Most 4GLs are interactive, nonprocedural, and are capable of database upkeep functions. The two languages selected for the CSECT Interaction Hierarchy Report, PL/1 and dBASE III PLUS, can be categorized as third generation high level procedural language and fourth generation nonprocedural language respectively.

B. TYPES OF HIGH LEVEL LANGUAGES

Since the first high level language was developed in the late 1950s, a number of additional high level languages have been introduced. Programming languages are often categorized into four areas: 1) procedural and nonprocedural; 2) general-purpose and special-purpose; 3) interpreted and compiled; and 4) batch and non-interactive.

1. Procedural Vs. Nonprocedural

A procedural language is one in which the user specifies a set of executable operations that are to be performed in sequence and which specify a procedure. Nonprocedural is a relative term. The closer the user can come to stating his problem without specifying the steps for solving it, the more nonprocedural the language.

All third generation high level languages are procedure oriented. The data manipulation language in dBASE III PLUS can be used in either programming or command mode. When commands are used in the command mode, such as "FIND EMPLOYEE 12345", the language is called nonprocedural. The same command can be incorporated into a dBASE program where the language is used as other high level languages as a procedural language.

2. General Vs. Special-Purpose

A general-purpose language is designed with no specific type of application in mind. A special-purpose language is one designed to satisfy a single objective. The objective might involve application area, the ease of use for a particular application, or the efficiency of the compiler or object code. Most languages are created to serve a specific purpose. Examples are COBOL for business data processing, PASCAL for teaching programming concepts and LISP for list processing. Special-purpose languages enable programmers to solve narrowly defined problems.

A built-in micro DBMS language like the one in dBASE III PLUS is also a special-purpose language. It is a command language designed for simplifying the construction of complex database management functions.

3. Interpreted Vs. Compiled

High level languages must be translated into machine language before they can be executed. This is usually accomplished in one of two ways: with a compiler or with an interpreter.

A compiler translates the program in its entirety. The result is a machine language program which can then be executed as many times as desired. An interpreter translates the source program one line at a time, first translating the line and then executing it. The cycle is repeated for each line of the program. Compiled programs usually run faster than interpreted programs. This is because that each line of a compiled program is translated once and only, regardless of how many times it is executed.

An interpretive language is better in the aspect that it permits interaction with the program during execution. This simplifies testing and verification of program logic and structure. BASIC and dBASE III PLUS are examples of interpreted languages.

4. Batch Vs. Interactive

Batch programming is most often used to solve problems for which immediate responses are not required. Most batch programs are used to solve specific problems that occur according to some predetermined schedule.

Interactive programming allows the programmer or end user to communicate directly with the computer in a conversational fashion. An interactive language will report an error for an incorrect input instantly upon entering a line. The programmer can correct the error while the purpose of the line is still in mind. Batch programs usually produce an error report at the end of the input data set. The programmer then corrects the input data offline and executes the program again. dBASE III PLUS can be used to implement both batch and interactive applications.

C. GENERAL FEATURES OF HIGH LEVEL LANGUAGES

1. Data Representation

All computers process data in one form or another. A constant is a data value that does not change. A variable can be thought of as a place to store a data value. Unlike constants variables can take on new values. In most high level languages a particular variable can hold only one type of data(real, integer, or string). Some languages require the user to declare in advance the variables that will be used in the program and what type each of these variables will be. Other languages incorporate default type variables based on the first letter of the variable's name.

High level languages that are very particular about the types of variables used, how they are declared, and how they are used are called strongly-typed languages. Examples of these are assembly and PL/1. Languages that are less sensitive to such matters are said to be loosely-typed such as dBASE III.

2. The Assignment Statement

An assignment statement is used to assign a particular value to a variable. Most languages denote this operation by a symbol called an assignment operator. They use either the equal sign(=) or a colon followed by an equal sign (:=) for the assignment operator. In both PL/1 and dBASE III PLUS, a programmer can write X = X + 1. It does not mean that X + 1 is equal to X. What this statement really says is "Assign the value of X plus one to X".

3. Arithmetic Expression

An arithmetic expression operates on a numeric value according to a given set of rules. In most high level languages an arithmetic expression followed by an arithmetic operator (+, -, *, /, etc.), then followed by another arithmetic expression is also an arithmetic expression.

The expression is one of the key features that distinguishes high level languages from low level languages. In a low level language only one thing can be done at a time, that is one operation per statement. An arithmetic expression in a high level language permits the programmer to accomplish many calculations with only one statement.

4. Logical Expression

A logical expression evaluates to a logical value, that value being true or false. The most common form of logical expression involves relational operators such as >, <, =, <=, >=, and <>. High level languages also feature logical operators such as AND, OR, and NOT. More complex logical expressions can be constructed by combining simpler logical expressions using AND and OR. Ambiguity can be avoided in a complex logical expression by liberal use of parentheses.

5. Input and Output

When programming in a microcomputer environment input data can be input from a keyboard or a diskette. The output can be directed to a screen, a diskette or a printer. Some languages can handle all combinations. In most languages the input function is handled by a READ statement and the output function is handled by a WRITE or a PRINT statement. The various languages differ in how much control the user has over the format of the output. Useful formatting features include the ability to control the number of decimal places printed, the total numer of columns allocated to a number, the number of spaces between printed columns, and so forth.

6. Control Structures

The natural flow of control in a program is sequential. A more complicated control structure is needed for all but the simplest applications. Following are the typical control structures.

a. IF-THEN-ELSE

The IF-THEN-ELSE contol structure allows the program to handle basic decisions. If the logical expression is true, the program executes the statement following THEN and passes control to the statement following ENDIF. If the logical expression is false, the program executes the statement following ELSE. In either case the next statement to be executed is the statement following ENDIF.

b. CASE

The IF-THEN-ELSE statement allows a two-way selection: the program selects one of two sets of statements to execute. Often it is necessary for the program to choose between more than two alternatives. The CASE statement provides a convenient way to do this.

c. Conditional Loops

One thing that computers do especially well is repetition. The control structure that performs repetitive tasks in a computer language is called a loop. There are two major types of loops in high level language, the indexed loop and the conditional loop.

Whereas the indexed loop executes a group of statements a specified number of times, the conditional loop executes a group of statements and tests against the specified condition each time through the loop until a specified condition is met. A few languages offer a variant of the conditional loop in which the conditional testing takes place at the bottom of the loop rather than at the top.

d. GOTO

The GOTO statement allows program control to be transferred to any arbitrary place in a program. While it provides a great convenience, indiscriminant use of the GOTO statement can lead to programs that are hard to read as well as difficult to debug and modify. In some languages GOTO is needed in order to emulate control structures such as PERFORM-UNTIL (in COBOL) that are not directly implemented.

7. Subprograms

a. Subroutines

It is often more convenient to divide programs into more-or-less self-contained segments or modules. Such modules are called subroutines. A subroutine can be placed within the program or be external to the program. Subroutines are usually activated by a CALL statement. When the subroutine has finished, program control returns to the statement following the CALL statement. Parameters and arguments can be used to pass values back and forth between the subroutine and the calling program.

There are several advantages to using subroutines: 1) The use of a subroutine permits large tasks to be divided; 2) Since a CALL statement can occur as many times as necessary in a program, the use of subroutines can often save considerable coding; 3) A commonly-used subroutine can be easily transported from one program to another.

b. Functions

Functions are similar to subroutines except in the manner in which they are invoked and in the manner in which values are returned to the invoking program. Some functions are supplied as part of a language such as square root(SQRT) in FORTRAN. The function is invoked by writing its name in an expression as if it were simply another variable. An example in FORTRAN is X = SQRT (4.0).

Functions can also be defined by the user in much the same manner as subroutines are defined. One difference is that the name of the function is usually treated as if it were a variable within the body of the function definition. The value of the function is returned through the function name.

c. Recursion

A function or a subroutine is said to be recursive if it calls or invokes itself. Recursion is different from iteration. Iteration is the repetition of a sequence of instructions until a given condition is met. Each performance is carried to completion, the condition is examined, and a new performance commenced if the result is unsatisfactory. In contrast to this recursion involves a self-nesting. The performance is not carried to completion before the condition is examined. Instead, the condition is examined within the performance. If the result is unsatisfactory, the whole performance is called again as a subroutine of the as yet uncompleted original one.

A recursive definition must always contain one non-recursive alternative or it becomes circular in the vicious sense. This is similar to an iterative process since this must also contain some means of "getting out of the loop" - whether by requiring a number of iterations which can be shown to be finite, or by requiring an exit when a convergence test has been ultimately satisfied.

8. Data Structures

There are many aspects to the use and representation of data structures in the field of computers. Some of the most commonly used data structures are arrays, lists, trees, stacks, and queues[3]. Each of these data structures should be carefully examined and selected to carry out the different data processing needs.

An array is a data structure whose elements may be selected by integer selectors called "indexes". The set of all elements of an array are generally created and deleted at the same time by means of declarations such as DIMENSION A(1,100) in FORTRAN. The execution of the declaration statement causes allocation of a block of storage space large enough to hold the arrays.

Similar to array structure, list structures may be characterized by their accessing creation and deletion operators. In a linear list each list element has an unique successor and the last element has an "empty" successor field. Insertion and deletion of elements in a list is accomplished by: 1) creation of a new list cell; 2) updating pointers of existing list elements and the newly created list elements. Elements of a list are accessed by walking along a pointer chain starting at the head of the list. List structures are flexible storage structures for objects of variable sizes, or tables of fixed-size objects in which insertions and deletions are frequently required.

A tree is a list in which there is one element called the "root" with no predecessor and in which every other element has an unique predecessor. Therefore, a tree is a list that contains no circular lists. In addition, no two list elements may have a common sublist as a successor. Elements of a tree which have no successor are called "leaves" of the tree. Tree elements, just as list elements, are generally accessed by walking along a pointer chain. However, the guarantee that there are no cycles or common sublists makes it possible to define orderly procedures for insertion and deletion of subtrees.

A stack is a linear list in which elements are accessed, created, and deleted in a last-in-first-out (LIFO) order. In order to access an element in a stack it is necessary to delete all the more recently entered elements from the stack. Thus, only the top of the stack is accessible. The two principle stack operations are pop and push.

A queue is a linear list in which elements are created and deleted in a first-in-first-out (FIFO) order. The insert operation can always be performed since there is no limit to the number of elements a queue may contain. The delete operation, however, can be applied only if the queue is nonempty.

9. File Handling

Data stored in files can be organized and accessed in different ways. A sequential file must be read from beginning to end. It is used most often when every record in the file must be processed during a run. To read a record in the middle of a sequential file, the program must read from the first record all the way to the record desired.

Direct access files are frequently called random access files. Any record in a direct access file can be accessed directly. To access a record in a direct access file, the record location must be known. Thus the programmer must set up some means of keeping track of information content and location. This usually requires maintaining an index of some sort. Some languages such as COBOL provide for automatic maintenance of an index for a file. ISAM file is an example. This can remove a significant burden from the programmer.

III. MICRO DBMS PROGRAMMING LANGUAGE

A. OBJECTIVES OF MICRO DBMS

A micro DBMS provides a convenient and efficient means to implement and access a database in a systematic manner. A good micro DBMS should accomplish the following objectives[4]:

1. Data Independence

The most important feature that a DBMS offers is data independence. An application is data dependent if it is impossible to change the way the data is physically stored or how it is accessed without affecting the application drastically. Data independence allows new data items to be added, deleted or the overall logical structure expanded without forcing existing programs to be rewritten. A data field may be stored in a form that will improve performance or economize storage space, whileas different applications can still view it the way they need to. Hardware and physical storage techniques can also be changed without causing application programs to be rewritten.

2. Controlled Redundancy

Data items will be stored only once except where there are technical or economic reasons for redundant storage. Different users who perceive the same data differently can employ them in different ways. In a time-critical processing situation, a trade-off between minimizing redundancy and maximizing processing time can be accepted.

3. Integrity Control

Integrity refers to the ability of a DBMS to ensure that the database contain only accurate data and protect the database from hardware, software and operational failure. Examples of database integrity support are record locks, recovery/restart, and security. In a multiuser environment DBMS's usually use record locks to control concurrent record updates. Recovery/restart requires saving of before and after update record images to some device. When necessary they can restore the before image of the record to a logical point and restart the application without destroying the integrity of the data. This is a complex process and usually is implemented in a mainframe environment. Backup and restore still is the most often used integrity control measure in a microcomputer environment.

4. Ease of Use

Complexity is hidden from the users by the DBMS. Users can gain access to data in a simple fashion. A query, non-procedural or report generation language should permit some end users to bypass the application programming step.

5. Security and availability

With proper security unauthorized access to the data will be prevented. The same data may be restricted in limited ways to different users.

Data is quickly available to users at almost all times when they are needed. A multiuser DBMS allows the same copy of the database to be shared among multiple online users and batch programs.

B. COMPONENTS OF A MICRO DBMS

As with dBASE III PLUS, most micro DBMSs provide the programmers with the following application-building tools:

1. A data definition command language that allows users to define databases with just a few commands. Database restructuring can also be done in a similar way with minimal user involvement.

2. An online full screen data display facility allows users to add, modify and display data in the data base sequentially without programming.

3. Sorting and indexing are convenient tools to arrange records in a specific order with one command. Sorting or indexing can be performed on multiple fields.

4. A menu-driven utility allows users to accomplish most database management operations by selecting appropriate menu and submenu options. Novice programmers can use this tool until they are more familiar with the software. Once they have gained some expertise with the process, they can use commands that allow them to specify their requirements.

5. A full screen text editor that allows programmers to code and edit the program source code.

6. A data manipulation language that gives the progammer a more advanded and efficient way to build an application. dBASE III PLUS command programs can access the database fields defined earlier with data definition commands, without further defining it within the programs. Once dBASE files are opened, they can be used for input and output.

7. A query facility that provides quick online display of the requested information that meets a set of conditions the user defines without programming. A menu-driven assistant utility can be used to create a query file which stores the filter conditions and can be invoked later.

8. A report generator that allows users to customize their printer or screen reports using the ASSIST

menu-driven utility. A similar label generator is also available.

9. A screen generator that allows easy creation of a customized data entry screen. Each screen field is tied to a data field of a database record. More complex screen input/output functions can be implemented in a program using screen I/O related commands.

C. SPECIAL FEATURES OF dBASE III PLUS COMMAND LANGUAGE

Among all the components mentioned above, the data manipulation language is the selected focus for this paper. It is this embedded command language that makes dBASE III PLUS a powerful data management tool. Thus it is worthwhile to take a closer look at how dBASE III PLUS is different from the non-database high level procedural language in a microcomputer environment.

dBASE III PLUS can operate in two modes: direct command mode and programming mode. In direct command mode the programmer issues a command at dBASE's "dot prompt". If the syntax is correct dBASE immediately performs the command and displays the results on the screen. With the direct commands available in dBASE III PLUS the user can exploit all the database management facilities dBASE has to offer.

For those whose needs are more complex, dBASE III PLUS also provides a complete programming language. To code or edit a dBASE program the programmer can access the dBASE text editor via the MODIFY COMMAND statement. To execute a program only requires one to issue a DO command with the program name. A program can be executed in either the batch mode or online interactive mode. The output listing can be directed to a printer or a screen.

All but a few of the dBASE direct commands are designed for practical use within a program as well as from the dot prompt. In addition to the vocabulary of direct commands, dBASE includes a set of instructions designed specifically to define the logic and structure of a program. These instructions provide the essential feature of a traditional high level language, making dBASE far more than just a command-driven database manager.

Figure 1 compares the general programming features between dBASE III PLUS and some other high level languages. The following sections present some of the important dBASE III PLUS features.

1. Variables

A variable is simply a name that represents a certain data value. Programs typically need storage space for specific data items that are required during program execution. In a dBASE program the major data structure usually con-

	BASIC	C	COBOL	dBASE	FORTRAN	PL/1
1. Math Capabilities	4	4	2	3	5	4
2. Character Handling	5	5	4	5	2	4
3. Data Structures	3	5	5	5	3	5
4. Control Structures	3	5	3	3	3	4
(IF-THEN-ELSE, CASE, RECURSION, CONTROLLED LOOP)						
5. Console Input/Output	5	4	2	5	4	4
6. File Input/Output	4	4	5	5	4	4
7. Subroutine Interface	2	3	2	4	5	4
8. Low-level Operation	3	5	2	2	2	3
9. User Friendliness	5	3	3	5	3	4
(English-like, Ease of Learning, Ease of Coding, Ease of Debugging, Ease of Maintaining, Self-documentation)						

Rating Scale: 5 = Excellent 1 = Poor

Figure 1: High Level Languages Comparison[5,6]

sists of open databases with which the program is working. However, other intermediate data items may also come into play and the program sets aside memory space for such items through the creation of variables. The type is determined when data is stored in the variable. dBASE III PLUS variable is loosely-typed. There is no need to declare variable type before they are used. dBASE uses the STORE command or "=" to assign a value to a variable. However, a program can also store a value for a variable from the screen via input commands such as INPUT, ACCEPT, @... GET. To gain access to the data item the program simply refers to the name of the variable in which the value is stored.

A variable in a dBASE program is a name assigned to a memory location that can be used to hold a data element, not a record. Most high level languages allow the programmer to store related information in temporary storage as a record so it can be retrieved and handled as a record.

2. Input and Output

dBASE has the input/output commands to receive information from the keyboard; and to send messages and information to the display terminal or printer.

a. The print commands ? and ?? are simple ways to send lines of text to the screen or printer.

b. The @... Say command presents formatted data at a specific location on the screen. To switch output to the printer no program change is required. The "SET DEVICE TO PRINTER" command can be issued at the dot prompt before printing.

c. The INPUT, ACCEPT, @... GET, READ, and WAIT commands accept information from the keyboard in a variety of ways.

3. Control Structures

A control structure defines alternative courses of action in a program. The choice of which course to take depends upon the value - TRUE or FALSE - of a conditional expression.

dBASE III PLUS supports the three most common control structures found in other high level languages: IF-THEN-ELSE, DO CASE, and DO WHILE. Nested loops are allowed. Two special loop control related commands are LOOP and EXIT. The LOOP command transfers execution to the beginning of the DO WHILE ... ENDDO structure, and the EXIT command aborts the looping process while execution continues with the command line following the ENDDO.

4. Modular Programming

The dBASE language encourages modularized, top-down approach programming. The GOTO command in dBASE is strictly a file operation command, not a program logic transfer command.

Each program module ends with a RETURN command which transfers excution back to the main program. The DO command combined with the program name will call and transfer control to that program. The RETURN command in the called program returns control to the line following the DO command in the calling program.

Data elements created in lower level modules are not automatically passed to higher level modules. The PUBLIC command can be used to declare that variables created in lower level modules, be shared by higher level modules. Variables can also be designated as PRIVATE so that the variables are recognized only within the module that creates them. Unlike variables, database records are considered public by every module in the program structure.

5. Debugging Commands

Very few programs perform perfectly during the first execution attempt. The process of locating and correcting the sources of program errors is called debugging. The dBASE program provides commands such as SET TALK and SET ECHO to help with this critical stage of program development. With "SET TALK ON" the dBASE III PLUS interpreter will display all the interactive messages on the screen. If some interactive messages are undesirable, users can use "SET ECHO OFF". This causes each command line to be displayed as it is executed. This will help users to locate a program error in a specific command line.

6. Database Management Functions

dBASE III PLUS database files are usually created in the dot prompt command mode. Once a file is created a data entry screen is available to load the file. Users can then add/modify/delete data as in the command mode. However, when routine massive updates are necessary, a set of dBASE III PLUS programs are usually written to perform the task. In the dBASE III PLUS program the user issues a "USE" command to open a file. The user can open multiple files if desired. dBASE III PLUS will keep track of the record currency for all files opened. The user then uses the "SELECT" command to move from one file to another. To move from one record to another within the same file, the user can issue commands in the program such as "GOTO 5", which means go to record 5. Other commands include: "GO TOP" - go to the top of the file; "GO BOTTOM" - go to the bottom of the file; "SKIP 2" - move the record pointer forward twice; and "SKIP -2" - move the record pointer backward twice. The "LOCATE" command sequentially searches the active database file for a record that satisfies a specified condition, while the "FIND" command searches for the first data record in an indexed file with a specified search key.

Data can be displayed, added, modified, or deleted once the desired record location is made current. The updated information can be obtained within the program from the screen or an updated file. The "DELETE" command does not delete records from the file, it only marks the records in an active database file with a deletion symbol(*). Records with a deletion symbol can be removed physically by the "PACK" command or can be recoverd by the "RECALL" command. Other file manangement functions which can be performed within the program are: 1) add data records from one data base file to the end of another file with an "APPEND FROM" command; 2) copy, rename, or erase a file; 3) create a new file by merging specified data records from two open files with the "JOIN" command; 4) rearrange data records in one or more key fields in ascending or descending order with a "SORT" command; and 5) create a key file in which all records are ordered according to the contents of the specified key field with an "INDEX" command.

IV. PROGRAMMING WITH dBASE III PLUS

A. CSECT INTERACTION PROBLEM DESCRIPTION

Microcomputer software vendors are constantly improving their products by eliminating bugs, adding user requested functions, and fully utilizing the most current microprocessor technology breakthroughs. All these improvements require program updates. A piece of successful comprehensive software involves tens or even hundreds of programs and subroutines. Changes made to a given program may affect the program it calls or the program that calls it. Changes made within a program may also affect the flow of control caused by JUMP instructions within the program. An automated program hierarchy report system was implemented in PL/1[7] on the microcomputer to provide complete information for all affected programs or subroutines. With this information, software maintenance programmers can start their job quicker with less errors. To explore the capability of a typical micro DBMS command language, dBASE III PLUS was chosen for its popularity to implement the same task.

Some software packages are implemented with assembly language because of its better utilization of storage and fast processing speed. When implementing a program hierarchy report system for an assembly-written software, the control section should be the object of analysis. A control section (CSECT) is a part of an assembly program specified by the programmer to be a relocatable unit. All elements of are to be loaded into adjoining virtual storage locations. A CSECT can be referred to by any other CSECT or separated assembled modules. For example, in an assembly language written software when changes are made to a CSECT called by 10 other CSECTs, these 10 CSECTs need to be examined to verify the necessity for modification. To find out how many other CSECTs will be affected by changes made to a single CSECT, one must answer the following questions:

1. What other CSECTs are called by this CSECT?
2. What other CSECTs call this CSECT?
3. What other CSECTs are jumped to by this CSECT?
4. What other CSECTs jump to this CSECT?

To answer questions 1 and 3 one must to examine all the CALL and JUMP instructions within a particular CSECT to determine what the targeted CSECTs are. To answer questions 2 and 4 one must examine all the CALL and JUMP instructions in other CSECTs to check if any of the target CSECTs match the CSECT that is to be updated. This process does not involve complicated decision making but is rather repetitive. It is a perfect microcomputer programming task which can help reserve the programmer's energy for more creative work. Besides, the computer can do the job much faster and more efficiently.

The Intel 8085A assembler instruction set is assumed to be used in the assembly programs analyzed here. The task can be implemented in two stages. First from the assembly output listing organize the information into meaningful data structures, so they can be used in the second stage. For each CSECT:

a. What are the beginning and ending addresses for this CSECT?

b. What are the labels within this CSECT? What are the label addresses?

c. What are the exit points within this CSECT? Do they exit to other CSECTs via JUMP or CALL instructions? What are the exit addresses?

In the second stage the CSECT Interaction Hierarchy analysis programs use the files built in the first stage to examine every exit point in each CSECT. If an exit in CSECT A has an exit type "CALL" and the targeted CSECT B can be found, an output record is created to show that CSECT A calls CSECT B. Also another output record is created to indicate that CSECT B is called by CSECT A.

If an exit in CSECT A jumps to a label within CSECT B, an output record is created showing that CSECT A jumps to CSECT B and another output record is built to show that CSECT B is jumped to from CSECT A. If an exit label cannot be found among all the CSECTs and all label names have been processed, this exit is flagged as "unresolvable".

The CSECT Interaction Hierarchy Report should contain the following information for each individual CSECT: 1) list all the CSECTs it calls; 2) all the CSECTs it is called by;

3) all the CSECTs it jumps to; 4) and all the CSECTs from which it is jumped. The unresolved exits should also be indicated.

B. INPUT/OUTPUT

In this paper it is assumed that the first stage has already been implemented. Three dBASE III PLUS input files were created with the structures shown in Figure 2.

dBASE III PLUS CSECT Interaction Hierarchy programs listed in Appendix A create an output data base file called "OUTPUT" to hold all information required for generating the CSECT Interaction Hierarchy Report(See Appendix D). The OUTPUT file structure is shown in Figure 3.

A sample of the CSECT Hierarchy Report is shown in Figure 4. Complete input and output file structure and data can be found in Appendix B and C respectively.

C. dBASE III METHODOLOGY

The hierarchy of CSECT interaction is constructed from the three dBASE III PLUS input files: CSECT, EXIT, and LABEL files. For each CSECT in the CSECT file, it is determined whether it is part of a linked CSECT group. Each linked group of CSECTs is assigned a number. If the CSECT is part of a linked group, the link field in the CSECT record is set to the assigned number. If the CSECT is not linked the link field is set to zero. The link group numbers created in the CSECT file are copied to the corresponding records

CSECT file: Provide CSECT information for all CSECTs.

	Field Name	Type	Width	Description
1	CSECTNO	Numeric	3	Csect Number
2	CSECTNAME	Character	8	Csect Name
3	GEGNADDRS	Numeric	4	Csect Beginning Address
4	ENDADDRS	Numeric	4	Csect Ending Address
5	CSECTLINK	Numeric	3	Csect Link Number

EXIT file: Provide exit information for all exits.

	Field Name	Type	Width	Description
1	ECSECTNO	Numeric	3	Csect Number
2	ECSECTNAME	Character	8	Csect Name
3	EXITNAME	Numeric	8	Csect Exit Names
4	EXITADRS	Numeric	4	Csect Exit Address
5	EXITYPE	Character	1	Exit Type (1 - Call, 2 - Jump)
6	EXITLINK	Numeric	3	Csect Link Number

LABEL file: Provide label information for all labels.

	Field Name	Type	Width	Description
1	LCSECTNO	Numeric	3	Csect Number
2	LCSECTNAME	Character	8	Csect Name
3	LABELNAME	Character	8	Csect Label Name
4	LABELADRS	Numeric	4	Csect Label Address
5	LABELINK	Numeric	3	Csect Link Number

Figure 2: dBASE Hierarchy Application Input Files

OUTPUT file: Provide information to build Csect Hierarchy listing.

	Field Name	Type	Width	Description
1	OCSECTNO	Numeric	3	Csect Number
2	ORECNO	Numeric	3	Output Record Number
3	OCSECT1	Character	8	Csect Name
4	OEXITYPE	Numeric	1	Relations Between OCSECT1 & OCSECT2 (1 - Call, 2 - Called by. 3 - Jump to, 4 - Jumped to by)
5	OCSECT2	Character	8	Target Csect Name
6	UNRESOLVE	Character	1	'Y' When Exit Address not found

Figure 3: dBASE Hierarchy Application Output File

```
DATASET: TSS2525.CSECT.DATA

                    CSECT HIERARCHY

IAOEPARM
--------

CSECT IAOEPARM DOES NOT CALL ANY CSECT
CSECT IAOEPARM IS NOT CALLED BY ANY CSECT
CSECT IAOEPARM DOES NOT JUMP TO ANY CSECT
CSECT IAOEPARM IS NOT JUMPED TO BY ANY CSECT

ICOEICOT
--------

CSECT ICOEICOT DOES NOT CALL ANY CSECT
CSECT ICOEICOT IS NOT CALLED BY ANY CSECT
CSECT ICOEICOT DOES NOT JUMP TO ANY CSECT
CSECT ICOEICOT IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
--------

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT
--------

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

IAOEAOFF
--------

CSECT IAOEAOFF DOES NOT CALL ANY CSECT
CSECT IAOEAOFF IS NOT CALLED BY ANY CSECT
CSECT IAOEAOFF DOES NOT JUMP TO ANY CSECT
CSECT IAOEAOFF IS NOT JUMPED TO BY ANY CSECT
```

Figure 4: Example of CSECT Hierarchy Report

in the LABEL and EXIT file to avoid the need for cross-referencing two tables. This allows minimizing of extra I/Os.

The basic program algorithm consists of the following steps:

1. Starting with the first CSECT in the CSECT file, the linked field is checked to determine whether the CSECT is part of a linked group. A CSECT is part of a linked group if its link number is not zero.

2. If the CSECT is part of a linked group and the exit label is not blank, then:

   a. The exit label is compared to the names of the other CSECTs in the same linked group.

   b. If the exit label is not found in 2.a., the exit label is compared to the names of the CSECTs not in the linked group.

   c. If the exit label is not found in 2.b., the exit label is compared to the labels within the other CSECTs in the same linked group.

   d. If the exit label is not found in 2.c., the exit label is compared to the labels within CSECTs which are not part of the linked group.

e. If the exit label is not found in 2.d., the exit label is not a label that has been processed and it is called "unresolvable".

3. If the CSECT is not part of a linked group (unlinked), then:

   a. the exit label is compared to the names of the other CSECTs.

   b. If the exit label is not found in 3.a., the exit label is compared to the labels in the other CSECTs.

   c. If the exit label is not found in 3.b., the exit label is not a label that has been processed and it is called "unresolvable".

4. Repeat steps 2 and 3 for the rest of the exit points in the same CSECT.

5. Repeat steps 2 to 4 for the rest of the CSECTs.

During the processing of steps 1 to 5 above, a CSECT hierarchy output file is created. The output file is sorted on OCSECT_NO, OEXIT_TYPE, and ORECNO. The sorted output file is then processed to produce the printout of the CSECT hierarchy which consists of an interaction table for each CSECT processed.

D. PL/1 SOLUTION VS. dBASE III SOLUTION

1. Methodologies

The dBASE III programs build the CSECT hierarchy into a single dBASE III file called OUTPUT. Each record has two CSECT names. The relation between the two CSECTs is represented by a single digit number. A "1" means the first CSECT calls the second CSECT. A "2" means the first CSECT is called by the second CSECT. A "3" means the first CSECT jumps to the second CSECT. A "4" means the first CSECT is jumped to from second CSECT.

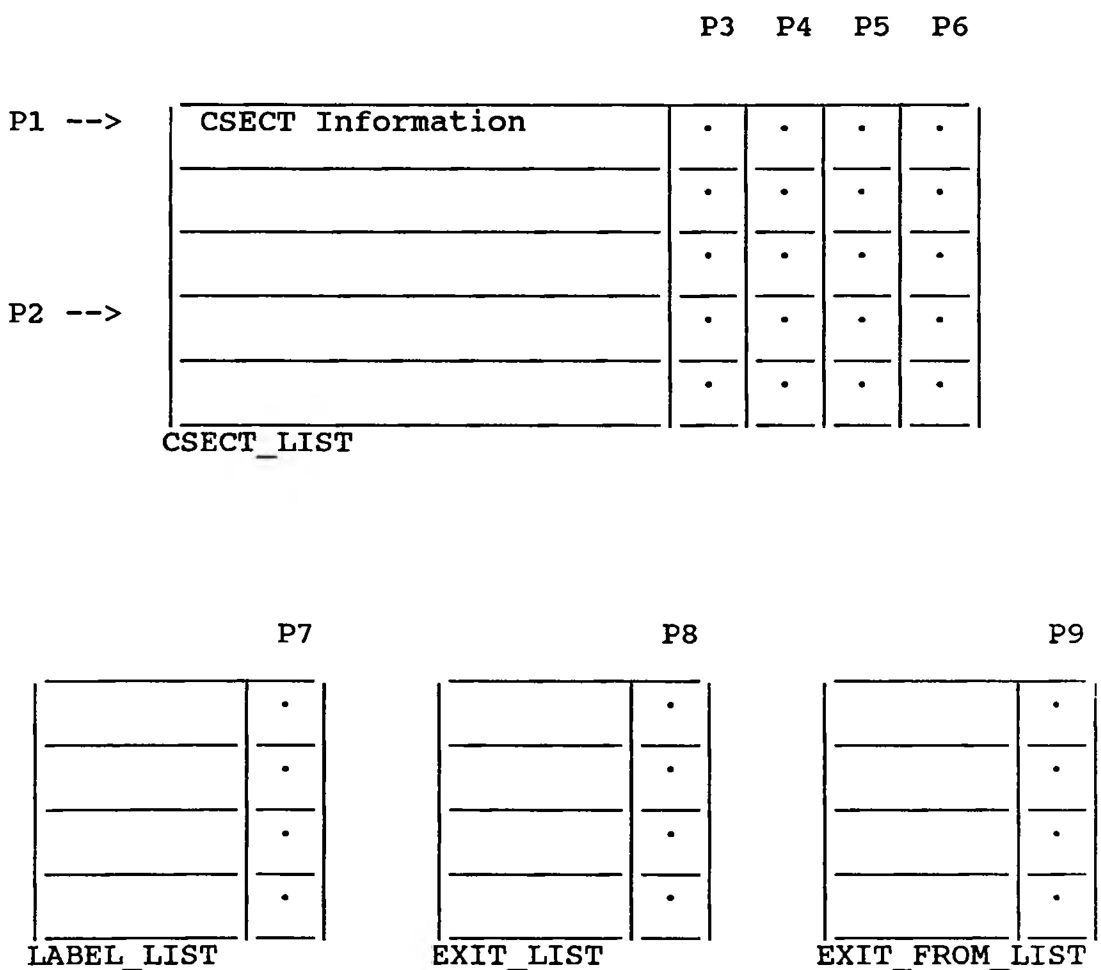
The OUTPUT file is then sorted on the first CSECT's number and the relation flag so the print programs can process the sorted OUTPUT file sequentially and produce the report in the requested format [Figure 4]. The PL/1 program[7] handles the problem in a more complex way. It first builds a circular CSECT list which contains all the CSECTs to be processed. For every CSECT in the CSECT linked list it then builds two other linked lists - EXIT_LIST and EXIT_FROM_LIST. The EXIT_LIST contains all the CSECTs that are called or jumped to by this CSECT. The EXIT_FROM_LIST contains all the CSECTs that call or jump to this CSECT. The pointers to these two lists are saved in the CSECT circular list.

When all CSECTs are processed the print subroutines process the CSECT circular list from top to bottom. For each CSECT to be printed, the two associated linked lists have to be processed twice. The PL/1 subroutines examine the EXIT_LIST to print all the CSECTs it calls and then it examines the EXIT_FROM_LIST to print all the CSECTs that call this CSECT. Then these two linked lists are re-examined to print all the CSECTs this CSECT jumps to and all the CSECTs that jump to it.

The dBASE III methodology is more straightforward. It can be divided into two parts. The first part builds the hierarchy into a file. The second part is to print the hierarchy from that file. Since dBASE III is coded in small modules and does not require compiling, the programs in the second part can be re-executed to reproduce the report without rebuilding the output files. Alternatively they can be easily modified to produce different reports based on the same file.

2. Data Structures

Figure 5 shows the linked list data structure adopted by the PL/1 program. Two types of pointers must be maintained by the programmer in this case. The first type is the "next record" pointer. Every record in the linked list must carry a next record pointer in order to allow walking through the list. Since the next CSECT pointer is unknown until the next record is created, the current CSECT pointer has to be saved. When the next CSECT record is created and the pointer allocated, the saved pointer is used to store this next CSECT pointer in the previous CSECT.



'.': Pointer
P1 : Pointer to the linked circular CSECT_LIST
P2 : Pointer to the current CSECT
P3 : Pointer to the linked LABEL_LIST
P4 : Pointer to the linked EXIT_LIST
P5 : Pointer to the linked EXIT_FROM_LIST
P6 thru P9: Pointers to the next entry in various linked lists

Figure 5: Data Structure Used in PL/I programs

In the circular CSECT list other than next CSECT pointers, each CSECT record has three other pointers. Each of these pointers points to a different link list. Pointer EXIT_HEAD_PTR points to EXIT_LIST which contains all the exits in this CSECT and the corresponding exit-to CSECT names. Pointer LABEL_HEAD_PTR points to LBEL_LIST which contains label information for all the labels in this CSECT. Pointer EXIT_FROM_HEAD points to a list of other CSECTs which either CALLS or JUMPS TO this CSECT.

Very often multiple link lists are built or accessed concurrently. Just trying to keep track of each pointer's function is a difficult task. This cumbersome pointer maintenance often inteferes with the logical thought process needed for problem resolution.

To determine whether an exit label of a particular CSECT matches a label in other CSECTs, the exit label address must be checked to see if it falls within the beginning and ending addresses of a CSECT. If it does then the next task is to find the matching label in that CSECT.

To perform the same task in dBASE III PLUS programs, LABEL_NAME in the LABEL file is examined. If a match is found, the LCSECT_ NAME on the same record gives the CSECT name of the matching label. Figure 6 shows the PL/1 program's complex label checking process.

```
LINKCHECK: PROC;

NEXT = CURRENT_CSECT -> NEXT_CSECT;
/*  DETERMIN  IF  THE EXIT ADDRESS IS GREATER THAN  THE  */
/*  BEGINNING ADDRESS OF A CSECT AND LESS THAN OR EQUAL  */
/* TO THE ENDING ADDRESS OF A CSECT.                     */
DO WHILE(NEXT ->= CURRENT_CSECT);
  IF NEXT -> LINK = CURRENT_CSECT ->LINK THEN
     IF (EXIT_POINT -> EXIT_ADR > NEXT -> BEG_CSECT_ADR) &
        (EXIT_POINT -> EXIT_ADR <= NEXT -> END_CSECT_ADR)
     THEN
       DO;
         CALL LABEL_CHECK;
         RETURN;
       END;
     ELSE
       ;
   ELSE
     ;
  NEXT = NEXT -> NEXT_CSECT;
END;


LABEL_CHECK: PROC;

/* DETERMINE IF EXIT LABEL MATCHES A LABEL IN THE CSECT */
/* PREVIOUSLY FOUND.                                    */

DO WHILE (LABEL_PTR ->= NULL);
  IF  LABEL_PTR -> LBEL_LIST.LBEL_NAME =
     EXIT_POINT -> EXIT_LIST.EXIT_LBEL THEN
     DO;
       EXIT_POINT-> EXIT_LIST.CSECT_EXITED_TO =
         NEXT -> CSECT_LIST.CSECT_NAME;
       CALL UPDATE_DATA;
       RETURN;
     END;
   ELSE
     LABEL_PTR = LABEL_PTR -> LBEL_LIST.NEXT_LBEL;
END;
EXIT_POINT -> EXIT_LIST.CSECT_EXITED_TO =
  NEXT -> CSECT_LIST.CSECT_NAME;
```

Figure 6. Label Checking Logic in PL/I Programs

3. Record Handling

In the PL/1 program input sequential files are read into storage and are built into linear linked list data structures. Each item in the list has a pointer used to access the next item in the list.

If CSECT_LIST is the name of a linked list, CURRENT_ CSECT is the external pointer that points to the list and NEXT_CSECT is the internal pointer that points to the next record in the list. The syntax for updating the external pointer in order to point to the next CSECT in the list structure in the PL/1 program is:

```
CURRENT_CSECT = CURRENT_CSECT -> CSECT_LIST.NEXT_CSECT.
```

When there is a need to skip a record, the syntax will repeat as follows:

```
CURRENT_CSECT = CURRENT_CSECT -> CSECT_LIST.NEXT_CSECT
CURRENT_CSECT = CURRENT_CSECT -> CSECT_LIST.NEXT_CSECT
```

dBASE III PLUS command language has an integrated DBMS. dBASE III PLUS keeps track of records for the users. In order to get to the next record the programmer simply codes "SKIP 1" or "GOTO NEXT". To skip one record and get to the third record simply code "SKIP 2".

Another powerful record handling feature is the LOCATE command. The LOCATE command will search an entire file from top to bottom until the selection criteria specified in the command is met or the end of the file is reached. The programmer does not need to code the loop control structure or set up a counter to handle the repetitive reading of the records. This makes the program source code shorter in length and much easier to maintain. The LOCATE command is an example of the nonprocedural language capability of dBASE III PLUS. The dBASE III PLUS programmer can use this command to tell the computer which records he wants instead of giving detailed instructions for the process. Figure 7 shows examples of searching a CSECT within the same linked CSECT group with PL/1 and dBASE III PLUS. It is obvious that the dBASE syntax is more English-like and user friendly.

4. Variables

In PL/1 variables are strongly-typed. Each variable must be declared as a certain type and length before it can be used. dBASE variables are loosely-typed. There is no need to declare a variable. The variable's type is determined by the value stored in it. The variable types in dBASE are oriented toward data processing business applications and are: character, numeric, date, memo, and logical.

5. File Definition

All dBASE III files are defined outside the program. The file definition and creation is independent of the program. The CREATE command with an acceptable file name brings up the field definition screen for defining the specification of each data field, such as its name, type and width. In the PL/1 program both the input and output files have to be defined.

```
/* DETERMINE IF THE EXIT LABEL MATCHES A NAME OF A CSECT */

/* IN THE SAME LINKED GROUP                              */
```

dBASE III PLUS:

```
     LOCATE FOR CSECTNAME =  TEXITNAME .AND. CSECTLINK =
        TEXITLINK .AND. CSECTNO <> TCSECTNO

     IF .NOT. EOF()
          DO OUTPUT
     ENDIF
```

PL/I:

```
     DO WHILE (NEXT ->= CURRENT_CSECT);
       IF CURRENT_CSECT -> LINK = NEXT -> LINK THEN
         IF EXIT_POINT -> EXIT_LBEL = NEXT ->
            CSECT_LIST.CSECT_NAME THEN
            DO;
              EXIT_POINT -> CSECT_EXITED_TO = '        ';
              CALL UPDATE_DATA;
              RETURN;
            END;
         ELSE
           ;
       ELSE
         ;
       NEXT = NEXT -> NEXT_CSECT;
     END;
     NEXT = CURRENT_CSECT -> NEXT_CSECT;
```

Figure 7.   Example of Powerful dBASE Command Language

6. Sorting and Indexing

In a linked list structure if the CSECTs must be stored in a certain sequence, it is the programmer's responsibility to plan ahead and implement the record insertion logic along with the necessary sorting criteria into the program.

In a dBASE environment sorting can be added to the program logic by inserting a SORT command. The SORT command does not change the record sequence in the original file. It creates an output file to hold the resequenced data. Sorting can also be done while in the command mode by keying the same SORT command at the dot prompt on the screen. This is very helpful for testing multiple sorted fields.

In the OUTPUT file created by dBASE CSECT interact programs, the data item OCSECTNO identifies the source CSECT. OCSECT1 is the name of the source CSECT. OCSECT2 is the name of the targeted CSECT. OEXITYPE is the exit type. If OEXITYPE = 1, it means OCSECT1 calls OCSECT2; if OEXITYPE = 1 and UNRESOLVE = "Y", then it means OCSECT1 calls an unresolvable OCSECT2. OEXITYPE = 2 means OCSECT1 is called by OCSECT2; OEXITYPE = 3 means OCSECT1 jumps to OCSECT2; and OEXITYPE = 4 means OCSECT1 is jumped to by OCSECT2.

These output records are created for every exit in each CSECT in a sequential manner. An example of an unsorted file is shown in Figure 8. Sorting on OCSECTNO and OEXITYPE will group all OUTPUT records for each CSECT together in the

```
 1  1IAOEPARM1
 1  2IAOEPARM3
 2  3ICOEICOT1
 2  4ICOEICOT3
 3  5IEVEADDR1
 3  6IEVEADDR3
 4  7IKBEKBDT1
 4  8IKBEKBDT3
 5  9IAOEAOFF1
 5 10IAOEAOFF3
 6 11ICCEPARM1
 6 12ICCEPARM3
 7 13IEVEADDR1
 7 14IEVEADDR3
 8 15IIOEAREA1
 8 16IIOEAREA3
 9 17ICCECLMP1ITEEABRT
26 18ITEEABRT2ICCECLMP
 9 19ICCECLMP1IWTEWAITY
10 20IEXEPARM1
10 21IEXEPARM3
11 22IEVEADDR1
11 23IEVEADDR3
12 24IIOEAREA1
12 25IIOEAREA3
13 26IEXEEXER1ITEEABRT
26 27ITEEABRT2IEXEEXER
13 28IEXEEXER1IWTEWAITY
13 29IEXEEXER1ITEEABRT
26 30ITEEABRT2IEXEEXER
13 31IEXEEXER1IWTEWAITY
14 32IITEPARM1
14 33IITEPARM3
15 34IEVEADDR1
15 35IEVEADDR3
16 36IIOEAREA1
16 37IIOEAREA3
17 38IITEINIT1
17 39IITEINIT3
19 40IEVEADDR1
19 41IEVEADDR3
20 42IIOEAREA1
20 43IIOEAREA3
21 44IKBEKBDT1
21 45IKBEKBDT3
22 46IMDEMAIN1IITEINIT
17 47IITEINIT2IMDEMAIN
22 48IMDEMAIN1IDMEDISPY
22 49IMDEMAIN1IWTEWAITY
```

Figure 8: OUTPUT File Not Sorted on ORECNO

call, called by, jump to, and jumped to by sequence which is required on the printout. However if a CSECT has multiple OUTPUT records for a particular exit type, i.e. one csect calls five other csects, the sorted order for these call exits does not necessarily conform to the original exit sequence. One way to preserve the original exit sequence is to add a field called "ORECNO". This is the sequence of the output records in the order in which they are created. The first OUTPUT record will have a value of one, the next will have a value of two, etc,. Then sorting on OCSECTNO, OEXITYPE and ORECNO will satisfy the printout request completely.

7. File Restructuring

The OUTPUT file in this application did not have the field ORECNO when it was created. It was discovered later that this field was necessary to produce the hierarchy report in the original exit sequence. The MODIFY STRUCTURE command provides a very convenient way to change the file structure while preserving all the data in the restructured file. Once the command is issued in the command mode, the file structure screen is displayed. The user can then modify the structure online. No further action is required from the user. This convenient feature shortens the application implementation time. Contrarily, a file restructuring in a PL/1 application always requires program modification and file conversion by the programmer or user.

8. Execution Speed

dBASE III PLUS is a relational DBMS. This means each file it creates is a table or sequential file and is ideal for processing sequential data. In order to access a record directly dBASE III PLUS uses a binary search technique to build and access an index file. The index file only contains the sorted ascending indexed fields and the pointers to the corresponding records in the database file. It is usually faster to sort the smaller index file than the database file itself. However in this paper to build the 304 OUTPUT records from 75 CSECT records, 174 EXIT records, and 288 LABEL records, it takes about 30 minutes execution time on a 10 Mhz turbo IBM PC-XT compatible system.

9. User Friendly Language

The dBASE programming language is very English-like. Its high level syntax is very similar to those languages used by business application programmers such as COBOL or BASIC. dBASE III PLUS is easy to understand, easy to read, and easy to code. The same set of user friendly commands used in the command mode for quick inquiry can also be used in the programming mode for more complex data processing.

10. Lines of Coding

Some dBASE III PLUS features simplify and shorten the program coding. For example, in dBASE III PLUS there is no need to declare a variable before accessing it. The statement "MOVE 1 TO X" declares variable "X" as a numeric variable and initializes it with a value of "1". The difficult problem of pointer maintenance in PL/1 is handled by dBASE III PLUS, not the user. All the files are defined outside the program code, and once defined it can be used in any program without redefining the files within the program. Without coding file and variable definitions in the program, the dBASE III PLUS programs for the CSECT Hierarchy Report uses 350 lines of source code, while the PL/1 solution uses 580 lines.

11. Modular Programming

The dBASE solution is coded in 12 different programs using between 6 and 60 lines of code. The storage restriction of 4K in program size and lack of COBOL paragraph or PL/1 procedure counterparts force programmers to use a modular programming technique. With this technique as each module is designed, the programmer can test it for syntax and logic errors before linking the modules together to form a complete system. It is also easier to reorganize the program modules when necessary. Reorganizing a dBASE III PLUS application usually involves modifying only some of the program modules and is often a simple task.

E. Summary of Advantages and Disadvantages

In this particular application all the functional requirements are successfully implemented using dBASE III PLUS without complex file structures and programming logic. In many cases dBASE III PLUS offers more advantages than PL/1.

1. Advantages of the dBASE III PLUS Solution
   - variable declaration is not required
   - data can be prepared on line
   - files can be sorted on multiple fields with a single command online
   - files can be displayed with a single command
   - files can be redesigned and restructured with a few commands
   - on line inquiry is possible with simple commands
   - on line debugging is possible
   - testing can be isolated to a module level
   - functional changes can be done at a module level
   - a CSECT Interaction Hierarchy Report can be displayed online with minimum changes
   - a CSECT Hierarchy Report can be regenerated without reconstructing the CSECT hierarchy

2. Advantages of the PL/1 Solution
   - compiled object code offers a faster execution time
   - the program can be run on a mainframe with minimum changes
   - better utilization of storage because storage is addressable at bit level

3. Disadvantages of the dBASE III PLUS solution
   - slow execution time with the interpreted dBASE III PLUS programs

4. Disadvantages of PL/1 solution

- the link list structure is hard to follow
- slow program development and testing.

V. CONCLUSIONS

A comparison of PL/1 and dBASE III PLUS solutions for the CSECT Hierarchy problem has been presented. While each language has its advantages and disadvantages, dBASE III PLUS is a better tool for this particular application because of its convenient features such as: integrated data base management function, data manipulation command language, mutiple field indexing and sorting, query capability, and menu-driven data definition.

There is no universal language that is best for all applications. The reason is that every programming language is designed with specific interests in mind. As data processing applications are often divided into two major categories - business oriented and science oriented - programming languages are often implemented to meet the requirements of only the requisite category.

PL/1 is equipped with features that are required and suited for scientific applications. These features include float data type, recursion, and arithmetic built-in functions. Unliked PL/1 dBASE III PLUS is designed for business applications. Extremely complex applications have been programmed with dBASE III plus and are available on the market. The Application Junction catalog published by Ashton-Tate provides a sampling of over 700 dBASE programs that cover a wide variety of applications[8].

Gary Elfring [9] suggests that the actual process of selecting a language should be broken into 3 major steps as shown in Figure 9. The first step is to characterize the application for which the language is being selected. Next, one must identify the features that a language should have in order to implement the previously described application. Finally, some practical consideration such as the availability, performance, and compatibility shoud be taken into account. Figure 9 provides a list of questions which should be answered before the selection decision is made. Both dBASE III PLUS and PL/1 are reasonable choices for the CSECT Hierarchy Report system according to the aspects presented in the Figure 9.

While both dBASE III PLUS and PL/1 can equally satisfy the functionalities required by the selected application, the user friendliness features become an important language selection factor. dBASE III PLUS command language's user friendliness features in areas such as training, coding, testing, maintenance and simplified file structures makes it a better choice than the non-DBMS procedural PL/1 for this CSECT project.

Step 1. Identify the Application

- What is the type or class of application?
- What level of language is needed?
- Is it too big to be expressed as one module?
- Is it too big to be fully understood by one programmer?

Step 2. Idendify Language Features

- What audience was the language designed for?
- What class of problems was the language designed to resolve?
- Can the syntax be understood?
- Is it terse or verbose?
- Is it consistent?
- What data types are supported?
- How are data types treated?
- Does the language support structured programming?
- Are exceptions possible?
- Is portability needed?
- How portable is the language?
- How is I/O handled?
- Is access to other programming languages needed?
- Is stand-alone product support required?
- Is real-time control needed?

Step 3. Practical Considerations

- How available is the language?
- How popular is the language?
- How does a user learn the language?
- What is the source of this information?
- What are the characteristics of the compiler?
- Is the code produced quick, compact, and predictable?
- What kind of software libraries are availble?

Figure 9. Choosing a Programming Language[9]

BIBLIOGRAPHY

1. "Corporate Bestsellers," Software News, August 1985, 32.

2. Chorafas, Dimitris N., Fourth and Fifth Generation Programming Languages, Vol. 1 McGraw-Hill Book Company, 1986, 53-54.

3. "Data Structure," Encyclopedia of Computer Science, Van Nostrand Reinhold Company, 1st Edition, 1976, 433-436.

4. Martin, James, Computer Database Organization, 2nd Edition, Prentice-Hall, NJ., 33-46.

5. Mandell, Steven L., Computers and Data Processing - Concepts and Applications, 3rd Edition, West Publishing Company, 1985, 243.

6. Taylor, Charles F., The Master Handbook of High-Level Microcomputer Language, TAB Books Inc., 1984, 351.

7. Kullman, Annette and Zobrist, George, "Program Hierarchy for Microcode", Proceedings of 28th Midwest Circuit and Systems Symposium, Louisville, Kentucky, August 1985.

8. Layman, Don. "All Aboard at Application Junction," PC Magazine, February 7, 1984, 144.

9. Elfring, Gary, "Choosing a Programming Language", Byte, June 1985, Vol. 10(No. 6), 235.

VITA

Karen Yingling Tam was born on October 16, 1955 in Taipei, Taiwan. She received her secondary education in Taichung, Taiwan. In May 1977, she graduated from Soochow University in Taipei, Taiwan with a B. A. in Social Work. During the next two years, she worked as a social worker to counsel teenagers with emotional problems.

In 1979, she came to the United States and began her data processing education at Washington University, St. Louis, Missouri. She received her B.S. in Systems and Data Processing from Washington University in 1982.

In 1982, while working at Concordia Publishing House as an application programmer, she began her graduate study in Computer Science at the University of Missouri-Rolla. One and half years later she was transferred to the Technical Support group and promoted to System Programmer.

Karen Tam has been married to Edwin Tam since 1984. In April 1985, the Tam family moved to Akron, Ohio. Karen is currently working as System Programmer at American Seaway Foods, Inc., Cleveland, Ohio.

APPENDIX A

dBASE CSECT INTERACTION PROGRAMS

```
*
*   PROGRAM  : INTERACT (MAININLINE)      CALLS:LNKCSECT
*                                                LNKOTHER
*
*   FUNCTION : THIS PROGRAM CONSTRUCTS A LIST OF CSECT
*       INTERACTIONS.
*
SET DEFA TO B
DO LNKCSECT
DO LNKOTHER
CLOSE DATABASES
SELECT 1
USE EXIT
DO WHILE .NOT. EOF ()
   STORE ECSECTNO   TO TECSECTNO
   STORE ECSECTNAME TO TCSECTI
   STORE EXITNAME   TO TEXITNAME
   STORE EXITADRS   TO TEXITADRS
   IF EXITYPE = 'C'
      STORE 1 TO TEXITYPE
   ELSE
      IF EXITYPE = 'J'
         STORE 3 TO TEXITYPE
      ENDIF
   ENDIF
   STORE EXITLINK TO TEXITLINK
   STORE 'N' TO TNOEXIT
   STORE 'N' TO TUNRESLV
   IF EXITNAME = '        '
      STORE 'Y' TO TNOEXIT
      DO OUTPUT
      SELECT 1
      SKIP
      LOOP
   ELSE
      STORE 'N' TO TUNRESLV
      IF EXITLINK = 0
         DO UNLINKED
      ELSE
         DO LINKED
     ENDIF
   ENDIF
   SELECT 1
   SKIP
ENDDO
CLOSE DATABASES
DO PRINTOUT
```

```
*
*  PROGGRAM  : LNKCSECT                CALLED BY:INTERACT
*
*  FUNCTION : THIS PROGRAM ASSIGNS A NUMBER TO CSCETS
*             BELONG TO THE SAME LINKED GROUP.
*
STORE 1 TO TCOUNTER
STORE 1 TO TLINKNUM
SELECT 1
USE CSECT
REPL CSECTLINK WITH TLINKNUM
STORE ENDADRS TO TENDADRS
DO WHILE .NOT. EOF()
   SKIP
   IF EOF()
      EXIT
   ELSE
      IF BEGNADRS >= TENDADRS
        REPL CSECTLINK WITH TLINKNUM
        STORE TCOUNTER + 1 TO TCOUNTER
      ELSE
        IF TCOUNTER = 1
           SKIP -1
           REPL CSECTLINK WITH 0
           SKIP
           REPL CSECTLINK WITH TLINKNUM
        ELSE
           STORE TLINKNUM + 1 TO TLINKNUM
           REPL CSECTLINK WITH TLINKNUM
        ENDIF
        STORE 1 TO TCOUNTER
      ENDIF
      STORE ENDADRS TO TENDADRS
   ENDIF
ENDDO
RETURN
```

```
*
*   PROGRAM   : LNKOTHER               CALLED BY: INTERACT
*
*   FUNCTION : THIS PROGRAM COPIES THE.LINKED.GROUP.
*              NUMBER  ESTABLISHED IN CSECT FILE TO.
*              EXIT FILE.
SELECT 1
USE CSECT
STORE CSECTNO   TO TCSECTNO
STORE CSECTLINK TO TCSECTLINK
DO WHILE .NOT. EOF()
   SELECT 2
   USE EXIT
   LOCATE FOR ECSECTNO = TCSECTNO
   REPL EXITLINK  WITH TCSECTLINK
   SKIP
   DO WHILE ECSECTNO = TCSECTNO
      REPL EXITLINK WITH TCSECTLINK
      SKIP
   ENDDO
   SELECT 3
   USE LABEL
   LOCATE FOR LCSECTNO = TCSECTNO
   REPL LABELINK WITH TCSECTLINK
   SKIP
   DO WHILE LCSECTNO = TCSECTNO
      REPL LABELINK WITH TCSECTLINK
      SKIP
   ENDDO
   SELECT I
   SKIP
   STORE CSECTNO   TO TCSECTNO
   STORE CSECTLINK TO TCSECTLINK
ENDDO
RETURN
```

```
*
*  PROGRAM  : LINKED                          CALLED BY: INTERACT
*                                             CALLS    : OUTPUT
*
*  FUNCTION : THIS PROGRAM PROCESSES THE CSECTS WHICH
*             BELONG TO A LINKED GROUP.
*
SELECT 2
USE CSECT
LOCATE FOR CSECTNAME = TEXITNAME .AND. CSECTLINK =
  TEXITLINK .AND. CSECTNO <> TECSECTNO .AND. TEXITADRS
  > BEGNADRS .AND. TEXITADRS <= ENDADRS
IF .NOT. EOF()
   STORE CSECTNO    TO TCSECTNO
   STORE CSECTNAME  TO TCSECT2
   USE OUTPUT
   DO OUTPUT
ELSE
   LOCATE FOR CSECTNAME = TEXITNAME .AND. CSECTNO <>
   TECSECTNO .AND. TEXITADRS > BEGNADRS .AND. TEXITADRS
   <= ENDADRS
   IF .NOT. EOF()
      STORE CSECTNO   TO TCSECTNO
      STORE CSECTNAME TO TCSECT2
      DO OUTPUT
   ELSE
     USE LABEL
      LOCATE FOR LABELNAME = TEXITNAME .AND. LABELINK.=
      TEXITLINK .AND. LCSECTNO <> TECSECTNO .AND.
      TEXITADRS = LABELADRS
      IF .NOT. EOF()
         STORE LCSECTNO   TO TCSECTNO
         STORE LCSECTNAME TO TCSECT2
         DO OUTPUT
      ELSE
        LOCATE FOR LABELNAME = TEXITNAME .AND. LCSECTNO
        <>TECSECTNO .AND. TEXITADRS = LABELADRS
        IF .NOT. EOF()
           STORE LCSECTNO   TO TCSECTNO
           STORE LCSECTNAME TO TCSECT2
           DO OUTPUT
        ELSE
           STORE 'Y' TO TUNRESLV
           DO OUTPUT
        ENDIF
     ENDIF
   ENDIF
ENDIF
RETURN
```

```
*
*  PROGRAM : UNLINKED                      CALLED BY: INTERACT
*                                          CALLS    : OUTPUT
*
*   FUNCTION : THIS PROGRAM PROCESSES THE CSECTS WHICH
*              DON'T BELONG TO ANY LINKED GROUP.
*
SELECT 2
USE CSECT
LOCATE FOR CSECTNAME = TEXITNAME .AND. CSECTNO <>
    TECSECTNO .AND. TEXITADRS > BEGNADRS .AND.
    TEXITADRS <= ENDADRS
IF .NOT. EOF()
   STORE CSECTNAME TO TCSECT2
   DO OUTPUT
ELSE
   USE LABEL
   LOCATE FOR LABELNAME = TEXITNAME .AND. LCSECTNO <>
      TECSECTNO .AND. TEXITADRS = LABELADRS
   IF .NOT. EOF()
      STORE LCSECNAME TO TCSECT2
      DO OUTPUT
   ELSE
      STORE 'Y' TO TUNRESLV
   ENDIF
ENDIF
RETURN
```

```
*
*   PROGRAM   : OUTPUT                     CALLED BY: INTEFACT
*                                                     UNLINKED
*                                                     LINKED
*
*   FUNCTION : THIS PROGRAM BUILDS OUTPUT FILE
*
SELECT 2
USE OUTPUT
APPEND BLANK
IF TUNRESLV = 'Y'
   REPL OCSECTNO  WITH TECSECTNO
   REPL ORECNO    WITH RECNO()
   REPL OCSECTI   WITH TCSECT1
   REPL OEXITYPE  WITH 1
   REPL OCSECT2   WITH TEXITNAME
   REPL UNRESOLVE WITH 'Y'
ELSE
   IF TNOEXIT = 'Y'
      REPL OCSECTNO WITH TECSECTNO
      REPL ORECNO   WITH RECNO()
      REPL OCSECTI  WITH TCSECT1
      REPL OEXITYPE WITH 1
      APPEND BLANK
      REPL OCSECTNO WITH TECSECTNO
      REPL ORECNO   WITH RECNO()
      REPL OCSECTI  WITH TCSECTI
      REPL OEXITYPE WITH 3
   ELSE
      REPL OCSECTNO WITH TECSECTNO
      REPL ORECNO   WITH RECNO()
      REPL OCSECT1  WITH TCSECT1
      REPL OEXITYPE WITH TEXITYPE
      REPL OCSECT2  WITH TCSECT2
      APPEND BLANK
      REPL OCSECTNO WITH TCSECTNO
      REPL ORECNO   WITH RECNO()
      REPL OCSECT1  WITH TCSECT2
      REPL OCSECT2  WITH TCSECT1
      IF TEXITYPE = 1
         REPL OEXITYPE WITH 2
      ELSE
         IF EXITTYPE = 3
            REPL OEXITYPE WITH 4
         ENDIF
      ENDIF
   ENDIF
ENDIF
RETURN
```

```
*
*   PROGRAM  : PRINTOUT                          CALLED BY: INTERACT
*                                                CALLS    : BEGNCHCK
*                                                           ENDCHCK
*                                                           GAPCHCK
*                                                           PRINTIT
*
*   FUNCTION:
*
*     1. SORT OUPUT FILE ON CSECT EBER AND EXIT TYPE
*     2. PRINT OUTPUT HEADING
*     3. IF THIS IS A NEW CSECT,
*        A) CALL 'ENDCHCK' TO PRINT APPROPRIATE INTERAC-
*           TION INFORMATION BEFORE PROCESSING CURRENT
*           ENTRY;
*        B) CALL 'BEGNCHCK' TO PRINT NEW CSECT HEADING
*           AND TO CHECK IF EXIT TYPE  STARTS  WITH 1.
*           IF  NOT,   PRINT  APPROPRIATE  INTERACTION
*           INFORNATION BEFORE PROCESSING CURRENT ENTRY;
*           ELSE CALL 'PRINTIT' TO PROCESS CURRENT ENTRY.
*     4. IF THIS IS NOT A NEW  CSECT, CALL 'GAPCHCK' TO
*        CHECK IF THERE IS A GAP BETWEEN PREVIOUS ENTRY
*        EXIT TYPE AND  CURRENT ENTRY  EXIT TYPE. IF  SO,
*        PRINT APPROPRIATE INFOPNATION BEFORE PROCESSING
*        CURRENT ENTRY;  ELSE CALL 'PRINTIT' TO PROCESS
*        CURRENT ENTRY.
*
SET DEVICE TO PRINT
SET TALK OFF
STORE 5 TO TLINENUM
@ TLINENUM,15 SAY "DATASET: TSS2525.CSECT.DATA"
STORE TLINENUM+2 TO TLINENUM
@ TLINENUM,28 SAY "CSECT HIEARARCHY"
STORE TLINENUM+3 TO TLINENUM
USE OUTPUT
SORT ON OCSECTNO,OEXITYPE,ORECNO TO SORTOUT
USE SORTOUT
STORE 0 TO TEXITYPE
DO BEGNCHCK
DO PRINTIT
STORE OCSECTNO TO TCSECTNO
STORE OCSECT1  TO TCSECT1
STORE OEXITYPE TO TEXITYPE
STORE OCSECT2  TO TCSECT2
SKIP
DO WHILE .NOT. EOF()
   IF OCSECTNO <> TCSECTNO
      DO ENDCHCK
      DO BEGNCHCK
   ELSE
      DO GAPCHCK
   ENDIF
   DO PRINTIT
   STORE OCSECTNO TO TCSECTNO
```

```
    STORE OCSECT1  TO TCSECT1
    STORE OEXITYPE TO TEXITYPE
    STORE OCSECT2  TO TCSECT2
    SKIP
ENDDO
RETURN
```

```
*
*   PROGRAM : BEGNCHCK                     CALLED BY: PRINTOUT
*                                          CALLS    : PRNTCHCK
*   FUNCTION:
*
*     1. PRINT CSECT HEADING.
*     2. IF CSECT EXIT TYPE STARTS WITH 2  THEN  THIS
*        CSECT DOES NOT CALL ANY CSECT.
*     3. IF CSECT EXIT TYPE STARTS WITH  3 THEN THIS
*        CSECT DOES NOT CALL ANY CSECT AND IS NOT CALLED
*        BY ANY CSECTS.
*     4. IF CSECT EXIT TYPE STARTS WITH 4 THEN THIS CSECT
*        DOES NOT CALL ANY CSECT, IS NOT  CALLED BY ANY
*        CSECT, AND DOES NOT JUMP TO ANY  CSECT.
*
@ TLINENUM,14  SAY OCSECT1
STORE TLINENUM+1 TO TLINENUM
@ TLINENUM,14  SAY "_______"
STORE TLINENUM+1 TO TLINENUM
DO PRNTCHCK
IF OEXITYPE >= 2
   @ TLINENUM,14 SAY "CSECT"
   @ TLINENUM,20 SAY OCSECT1
   @ TLINENUM,29 SAY "DOES NOT CALL ANY CSECT"
   DO PRNTCHCK
ENDIF
IF OEXITYPE >= 3
   @ TLINENUM,14 SAY "OCSECT"
   @ TLINENUM,20 SAY OCSECT1
   @ TLINENUM,29 SAY "IS NOT CALLED BY ANY CSECT"
   DO PRNTCHCK
ENDIF
IF OEXITYPE = 4
   @ TLINENUM,14 SAY "CSECT"
   @ TLINENUM,20 SAY OCSECT1
   @ TLINENUM,29 SAY "DOES NOT JUMPED TO ANDY CSECT"
   DO PRNTCHCK
ENDIF
RETURN
```

```
*
*   PROGRAM : ENDCHCK                      CALLED BY: PRINTOUT
*                                          CALLS    : PRNTCKCK
*
*   FUNCTION:
*
*      1. IF CSECT EXIT TYPE ENDS WITH 1 THEN THIS CSECT
*         IS NOT CALLED BY ANY CSECT, DOES NOT JUMP TO
*         ANY CSECT, AND IS NOT JED TO BY ANY CSECT.
*      2. IF CSECT EXIT TYPE ENDS WITH 2 THEN THIS CSECT
*         DOES NOT JUMP TO ANY CSECT AND IS NOT JUMPED
*         TO BY ANY CSECT.
*      3. IF CSECT EXIT TYPE ENDS WITH 3 THEN THIS CSECT
*         IS NO JUMPED TO BY ANY CSECT.
*
IF TEXITYPE < 2
   @ TLINENUM,14 SAY "CSECT"
   @ TLINENUM,20 SAY TCSECT1
   @ TLINENUM,29 SAY "IS NOT CALLED BY ANY CSECT"
   DO PPNTCHCK
ENDIF
IF TEXITYPE < 3
   @ TLINENUM,14 SAY "CSECT"
   @ TLINENUM,20 SAY TCSECT1
   @ TLINENUM,29 SAY "DOES NOT JUMP TO ANY CSECT"
   DO PRNTCHCK
ENDIF
IF TEXITYPE < 4
   @ TLINENUM,14 SAY "CSECT"
   @ TLINENUM,20 SAY TCSECT1
   @ TLINENUM,29 SAY "IS NOT JUMPED TO BY ANY CSECT"
   STORE TLINENUM TO TLINENUM
   DO PRNTCHCK
ENDIF
RETURN
```

```
*
*   PROGRAM : GAPCHCK                    CALLED BY: PRINTOUT
*                                        CALLS    : PRNTCKCK
*
*   FUNCTION:
*
*      1.  IF PREVIOUS ENTRY EXIT TYPE IS 1, AND
*             A. IF CURRENT ENTRY EXIT TYPE IS 3, THEN
*                THIS CSECT IS NOT CALLED BY ANY CSECT;
*             B. IF CURRENT ENTRY EXIT TYPE IS 4, THEN
*                THIS CSECT IS NOT CALLED BY ANY CSECT
*                AND DOES NOT JUMP TO ANY CSECT.
*      2.  IF PREVIOUS ENTRY EXIT TYPE IS 2 AND CURRENT
*          ENTRY EXIT TYPE IS 4 THEN THIS CSECT IS NOT
*          JUMPED TO BY ANY CSECT.
*
IF TEXITYPE = 1
   IF OEXITYPE >= 3
      @ TLINENUM,14 SAY "CSECT"
      @ TLINENUM,20 SAY OCSECT1
      @ TLINENUM,29 SAY "IS NOT CALLED BY ANY CSECT"
      DO PRNTCHCK
   ENDIF
   IF OEXITYPE = 4
      @ TLINENUM,14 SAY "CSECT"
      @ TLINENUM,20 SAY OCSECT1
      @ TLINENUM,29 SAY "DOES NOT JUMP TO ANY CSECT"
      DO PRNTCHCK
   ENDIF
ELSE
   IF TEXITYPE = 2 .AND. OEXITYPE = 4
      @ TLINENUM,14 SAY "CSECT"
      @ TLINENUM,20 SAY OCSECT1
      @ TLINENUM,29 SAY "DOES NOT JUMP TO ANY CSECT"
      DO PRNTCHCK
   ENDIF
ENDIF
RETURN
```

```
*
*  PROGRAM : PRNTCHCK                 CALLED BY: BEGNCHCK
*                                                GAPCHCK
*                                                ENDCHCK
*                                                PRINT IT
*
*  FUNCTION: IF NEW PAGE, PRINTS PAGE HEADING.
*
STORE TLINENUM+1 TO TLINENUM
IF TLINENUM >= 60
   EJECT
   @ 5,15 SAY "DATASET: TSS2525.CSECT.DATA"
   STORE 8 TO TLINENUM
ENDIF
RETURN
```

```
*
*   PROGRAM : PRINTIT                    CALLED BY: PRINTOUT
*                                        CALLS    : PRNTCHCK
*
*   FUNCTION:
*
*     1. IF CURRENT OUTPUT ENTRY EXIT TYPE IS 1, AND
*           IF OCSECT2 IS EMPTY THEN OCSECT1 DOES NOT
*              CALL ANY CSECT;
*           IF OCSECT2 IS NOT EMPTY THEN OCSECT1 CALLS
*              OCSECT2.
*     2. IF CURRENT OUTPUT ENTRY EXIT TYPE IS 2 THEN
*           OCSECT1 IS CALLED BY OCSECT2.
*     3. IF CURRENT OUTPUT ENTRY EXIT TYPE IS 3, AND
*           IF OCSECT2 IS EMPTY THEN OCSECT1 DOES NOT
*              JUMP TO ANY CSECT;
*           IF OCSECT2 IS NOT EMPTY THEN OCSECT1 JUMPS
*              TO OCSECT2.
*     4. IF CURRENT OUTPUT ENTRY EXIT TYPE IS 4 THEN
*           THIS OCSECT1 IS JUMPED TO FROM OCSECT2.
*
IF OEXITYPE = 1
   IF OCSECT2 = "        "
      @ TLINENUM,14 SAY "CSECT"
      @ TLINENUM,20 SAY OCSECT1
      @ TLINENUM,29 SAY "DOES NOT CALL ANY CSECT"
   ELSE
      IF OEXITYPE <> TEXITYPE
         @ TLINENUM,14 SAY "CSECT"
         @ TLINENUM,20 SAY OCSECT1
         @ TLINENUM,29 SAY "CALLS "
      ENDIF
      IF UNRESOLVE = 'Y'
         @ TLINENUM,35 SAY "UNRESOLVED LABEL"
         @ TLINENUM,52 SAY OCSECT2
      ELSE
         @ TLINENUM,35 SAY "CSECT "
         @ TLINENUM,41 SAY OCSECT2
      ENDIF
   ENDIF
ELSE
   IF OEXITYPE = 2
      IF OEXITYPE <> TEXITYPE
         @ TLINENUM,14 SAY "CSECT"
         @ TLINENUM,20 SAY OCSECT1
         @ TLINENUM,29 SAY "IS CALLED BY "
      ENDIF
      @ TLINENUM,42  SAY "CSECT "
      @ TLINENUM,48  SAY OCSECT2
   ELSE
      IF OEXITYPE = 3
         IF OCSECT2 = "        "
            @ TLINENUM,14 SAY "CSECT"
            @ TLINENUM,20 SAY OCSECT1
```

```
            @ TLINENUM,29 SAY "DOES NOT JUMP TO ANY CSECT"
         ELSE
            IF OEXITYPE <> TEXITYPE
               @ TLINENUM,14 SAY "CSECT"
               @ TLINENUM,20 SAY OCSECT1
               @ TLINENUM,29 SAY "JUMPS TO"
            ENDIF
            @ TLINENUM,38 SAY "CSECT"
            @ TLINENUM,44 SAY OCSECT2
         ENDIF
      ELSE
         IF OEXITYPE <> TEXITYPE
            @ TLINENUM,14 SAY "CSECT"
            @ TLINENUM,20 SAY OCSECT1
            @ TLINENUM,29 SAY "IS JUMPED TO BY"
         ENDIF
         @ TLINENUM,45 SAY "CSECT"
         @ TLINENUM,51 SAY OCSECT2
      ENDIF
   ENDIF
ENDIF
DO PRNTCHCK
RETURN
```

APPENDIX B

dBASE INPUT/OUPUT FILE STRUCTURES

Structure for database: CSECT.dbf
Number of data records: 75

Field	Field Name	Type	Width
1	CSECTNO	Numeric	3
2	CSECTNAME	Character	8
3	BEGNADRS	Numeric	4
4	ENDADRS	Numeric	4
5	CSECTLINK	Numeric	3
** Total **			23

Structure for datbase: EXIT.dbf
Number of data records: 176

Field	Field Name	Type	Width
1	ECSECTNO	Numeric	3
2	ECSECTNAME	Character	8
3	EXITNAME	Character	8
4	EXITADRS	Numeric	4
5	EXITYPE	Character	1
6	EXITLINK	Numeric	3
** Total **			28

Structure for database: LABEL.dbf
Number of data records: 288

Field	Field Name	Type	Width
1	LCSECTNO	Numeric	3
2	LCSECTNAME	Character	8
3	LABELNAME	Character	8
4	LABELADRS	Numeric	4
5	LABELINK	Numeric	3
** Total **			27

Structure for database: OUTPUT.dbf
Number of data records: 304

Field	Field Name	Type	Width
1	OCSECTNO	Numeric	3
2	ORECNO	Numeric	3
3	OCSECT1	Character	8
4	OEIXTYPE	Numeric	1
5	OCSECT2	Character	8
6	UNRESOLVE	Character	1
** Total **			25

Structure for database: SORTOUT.dbf
Number of data records: 304

Field	Field Name	Type	Width
1	OCSECTNO	Numeric	3
2	ORECNO	Numeric	3
3	OCSECT1	Character	8
4	OEXITYPE	Numeric	1
5	OCSECT2	Character	8
6	UNRESOLVE	Character	1
** Total **			25

APPENDIX C

dBASE INPUT/OUTPUT DATA

<u>CSECT</u> <u>FILE</u>

```
 1IAOEPARM   0   0  1
 2ICOEICOT  12  12  1
 3IEVEADDR 303 303  1
 4IKBEKBDT 303 303  1
 5IAOEAOFF16061791  1
 6ICCEPARM   0   0  2
 7IEVEADDR   0   0  2
 8IIOEAREA   0   0  2
 9ICCECLMP   0 158  2
10IEXEPARM   0   0  3
11IEVEADDR   0   0  3
12IIOEAREA   0   0  3
13IEXEEXER   0 156  3
14IITEPARM   0   0  4
15IEVEADDR   0   0  4
16IIOEAREA   0   0  4
17IITEINIT   0  56  4
18IMDEPARM   0   0  5
19IEVEADDR 446 446  5
20IOEAREA  446 446  5
21IKBEKBDT 446 446  5
22IMDEMAIN17492206  5
23ITEEPARM   0   0  6
24IEVEADDR 287 287  6
25IIOEAREA 287 287  6
26ITEEABRT 287 417  6
27IBOEPARM   0   0  7
28ICOEICOT   2   2  7
29IEVEADDR 293 293  7
30IIOEAREA 293 293  7
31IBOEBCOT 293 579  7
32IBTEPARM   0   0  8
33IEVEADDR 745 745  8
34IIOEAREA 745 745  8
35IKBEKBDT 745 745  8
36IBTEBLDT20482426  8
37EBTSTAND24272489  8
38EBTDIEVA24902552  8
39EBTUPADS25532601  8
40EBTUPADD26022650  8
41IPTEPARM   0   0  9
42IEVEADDR  10  10  9
43IKBEKBDT  10  10  9
44IPTEPROC13131446  9
45EPTNORMP14471538  9
46EPTLOCKP15391709  9
47EPTSHIFT17101877  9
```

```
48EPTSPACE18782017   9
49IRKEPARM    0    0 10
50IEVEADDR   37   37 10
51IRKERKBT   37  174 10
52ISEEPARM    0    0 11
53IEVEADDR    1    1 11
54IKBEKBDT    1    1 11
55ODRRDMFR13041406 11
56ISGEPARM    0    0 12
57IEVEADDR    9    9 12
58IKBEKBDT    9    9 12
59ISGESNDG13121422 12
60ITKEPARM    0    0 13
61IEVEADDR    5    5 13
62IIOEAREA    5    5 13
63IKBEKBDT    5    5 13
64ITKETEST13082265 13
65ETKUPADS22662314 13
66ETLREAD[23152515 13
67IUCEPARM    0    0 14
68IEVEADDR    0    0 14
69IIOEAREA    0    0 14
70IUCEUNCL    0  150 14
71IAOEPARM    0    0 15
72ICOEICOT   12   12 15
73IEVEADDR  303  303 15
74IKBEKBDT  303  303 15
75IAOEAOFF16061880 15
```

EXIT FILE

```
 1IAOEPARM           0   1
 2ICOEICOT           0   1
 3IEVEADDR           0   1
 4IKBEKBDT           0   1
 5IAOEAOFF           0   1
 6ICCEPARM           0   2
 7IEVEADDR           0   2
 8IIOEAREA           0   2
 9ICCECLMPITEEABRT   0C  2
 9ICCECLMPIWTEWAIT8102C  2
10IEXEPARM           0   3
11IEVEADDR           0   3
12IIOEAREA           0   3
13IEXEEXERITEEABRT   0C  3
13IEXEEXERIWTEWAIT8102C  3
13IEXEEXERITEEABRT   0C  3
13IEXEEXERIWTEWAIT8102C  3
14IITEPARM           0   4
15IEVEADDR           0   4
16IIOEAREA           0   4
17IITEINIT           0   4
19IEVEADDR           0   5
20IIOEAREA           0   5
21IKBEKBDT           0   5
22IMDEMAINIITEINIT   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIWTEWAIT8102C  5
22IMDEMAINIWTEWAIT8102C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIBOEBCOT   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIRKERKBT   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIBTEBLDT   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINICCECLMP   0C  5
22IMDEMAINIUCEUNCL   0C  5
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIWTEWAIT8102C  5
22IMDEMAINIWTEWAIT8102C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIEXEEXER   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINITKETEST   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIAOEAOFF   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIPTEPROC   0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIUCEUNCL   0C  5
22IMDEMAINIDMEDISP8099C  5
```

```
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINISGESNDG    0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINISEESNDE    0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINISGESNDG    0C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIUCEUNCL    0C  5
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIUCEUNCL    0C  5
22IMDEMAINIWSEWRIT8156C  5
22IMDEMAINIDMEDISP8099C  5
22IMDEMAINIWTEWAIT8102C  5
23ITEEPARM            0   6
24IEVEADDR            0   6
25IIOEAREA            0   6
26ITEEABRTIDMEDISP8099C  6
27IBOEPARM            0   7
28ICOEICOT            0   7
29IEVEADDR            0   7
30IIOEAREA            0   7
31IBOEBCOTIWTEWAIT8102C  7
31IBOEBCOTIWTEWAIT8102C  7
31IBOEBCOTITEEABRT    0C  7
31IBOEBCOTITEEABRT    0C  7
31IBOEBCOTIWTEWAIT8102C  7
32IBTEPARM            0   8
33IEVEADDR            0   8
34IIOEAREA            0   8
35IKBEKBDT            0   8
36IBTEBLDTEBTSTAND2427C  8
36IBTEBLDTEBTDIEVA2490C  8
36IBTEBLDTEBTSTAND2427C  8
36IBTEBLDTEBTUPADS2553C  8
36IBTEBLDTEBTUPADD2602C  8
36IBTEBLDTEBTDIEVA2490C  8
36IBTEBLDTEBTUPADS2553C  8
36IBTEBLDTEBTSTAND2427C  8
36IBTEBLDTITEEABRT    0C  8
37EBTSTAND            0   8
38EBTDIEVA            0   8
39EBTUPADS            0   8
40EBTUPADD            0   8
41IPTEPARM            0   9
42IEVEADDR            0   9
43IKBEKBDT            0   9
44IPTEPROCEPTNORMP1447C  9
44IPTEPROCEPTLOCKP1539C  9
```

```
44IPTEPROCEPTSPACE1878C  9
44IPTEPROCEPTSHIFT1710C  9
45EPTNORMP           0   9
46EPTLOCKP           0   9
47EPTSHIFT           0   9
48EPTSPACE           0   9
49IRKEPARM           0  10
50IEVEADDR           0  10
51IRKERKBTIWSEWRIT8156C 10
51IRKERKBTIDMEDISP8099C 10
51IRKERKBTIWSEWRIT8156C 10
52ISEEPARM           0  11
53IEVEADDR           0  11
54IKBEKBDT           0  11
55ISEESNDEIWSEWRIT8156C 11
55ISEESNDEIWSEWRIT8156C 11
56ISGEPARM           0  12
57IEVEADDR           0  12
58IKBEKBDT           0  12
59ISGESNDGIWSEWRIT8156C 12
59ISGESNDGIWSEWRIT8156C 12
59ISGESNDGIWSEWRIT8156C 12
60ITKEPARM           0  13
61IEVEADDR           0  13
62IIOEAREA           0  13
63IKBEKBDT           0  13
64ITKETESTITEEABRT   0C 13
64ITKETESTIWTEWAIT8102C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
64ITKETESTITEEABRT   0C 13
64ITKETESTETKREADP2315C 13
64ITKETESTETKUPADS2266C 13
65ETKUPADS           0  13
66ETKREADPIWTEWAIT8102C 13
66ETKREADPITEEABRT   0C 13
66ETKREADPIWTEWAIT8102C 13
67IUCEPARM           0  14
```

67IUCEPARM	0	14
68IEVEADDR	0	14
69IIOEAREA	0	14
70IUCEUNCLITEEABRT	0C	14
70IUCEUNCLITEEABRT	0C	14
71IAOEPARM	0	15
72ICOEICOT	0	15
73IEVEADDR	0	15
74IKBEKBDT	0	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBOEBCOT	0C	15
75IAOEAOFFIBTEBLDT	0C	15
75IAOEAOFFIBTEBLDT	0C	15

LABEL FILE

```
 1IAOEPARMIAOEPARM    0  1
 2ICOEICOTICOEICOT   12  1
 3IEVEADDRIEVEADDR  303  1
 4IKBEKBDTIKBEKBDT  303  1
 5IAOEAOFFIAOEAOFF1606  1
 5IAOEAOFFLBTBWHL11642  1
 5IAOEAOFFLSTELSE51696  1
 5IAOEAOFFLSTELSE71710  1
 5IAOEAOFFLSTENDF81717  1
 5IAOEAOFFLBTADDUM1737  1
 5IAOEAOFFLBTENIF91774  1
 5IAOEAOFFLBTEWHL11787  1
 6ICCEPARMICCEPARM    0  2
 7IEVEADDRIEVEADDR    0  2
 8IIOEAREAIIOEAREA    0  2
 9ICCECLMPICCECLMP    0  2
 9ICCECLMP@@DL0009   45  2
 9ICCECLMP@@EN0010   63  2
 9ICCECLMP@@DL0028  123  2
 9ICCECLMP@@EN0029  141  2
 9ICCECLMP@@EL0029  143  2
10IEXEPARMIEXEPARM    0  3
11IEVEADDRIEVEADDR    0  3
12IIOEAREAIIOEAREA    0  3
13IEXEEXERIEXEEXER    0  3
13IEXEEXER@@DL0006   22  3
13IEXEEXER@@DL0010   47  3
13IEXEEXER@@EN0011   65  3
13IEXEEXER@@DL0026  109  3
13IEXEEXER@@EN0027  127  3
14IITEPARMIITEPARM    0  4
15IEVEADDRIEVEADDR    0  4
16IIOEAREAIIOEAREA    0  4
17IITEINITIITEINIT    0  4
18IMDEPARMIMDEPARM    0  5
19IEVEADDRIEVEADDR  446  5
20IIOEAREAIIOEAREA  446  5
21IKBEKBDTIKBEKBDT  446  5
22IMDEMAINIMDEMAIN1749  5
22IMDEMAIN@@DL00521780  5
22IMDEMAIN@@EN00531795  5
22IMDEMAIN@@DL00641800  5
22IMDEMAIN@@DL00651800  5
22IMDEMAIN@@EN00661817  5
22IMDEMAIN@@EN00781887  5
22IMDEMAIN@@EN00971964  5
22IMDEMAIN@@00104 2021  5
22IMDEMAIN@@EN01112049  5
22IMDEMAIN@@EL01112063  5
22IMDEMAIN@@EN01182085  5
22IMDEMAIN@@EL01182094  5
```

```
22IMDEMAIN@@EN01032097  5
22IMDEMAIN@@EN01262130  5
22IMDEMAIN@@EL01032134  5
22IMDEMAIN@@EN00922137  5
22IMDEMAIN@@EL00922154  5
22IMDEMAIN@@EN00852157  5
22IMDEMAIN@@EL00852180  5
22IMDEMAINLNEWKBRD2180  5
22IMDEMAIN@@DL01392186  5
22IMDEMAIN@@EN01402201  5
23ITEEPARMITEEPARM   0  6
24IEVEADDRIEVEADDR 287  6
25IIOEAREAIIOEAREA 287  6
26ITEEABRTITEEABRT 287  6
26ITEEABRTL02      356  6
26ITEEABRTL03      362  6
26ITEEABRTL08      368  6
26ITEEABRTL28      374  6
26ITEEABRTL29      380  6
26ITEEABRTL0A      386  6
26ITEEABRTL2A      392  6
26ITEEABRTL44      398  6
26ITEEABRTLERROR   404  6
26ITEEABRTLENDCASE 407  6
26ITEEABRT@@DL0040 416  6
27IBOEPARMIBOEPARM   0  7
28ICOEICOTICOEICOT   2  7
29IEVEADDRIEVEADDR 293  7
30IIOEAREAIIOEAREA 293  7
31IBOEBCOTIBOEBCOT 293  7
31IBOEBCOT@@DL0009 339  7
32IBTEPARMIBTEPARM   0  8
33IEVEADDRIEVEADDR 745  8
34IIOEAREAIIOEAREA 745  8
35IKBEKBDTIKBEKBDT 745  8
36IBTEBLDTIBTEBLDT2048  8
36IBTEBLDT@@EN00172089  8
36IBTEBLDTLBTBWHL32146  8
36IBTEBLDTLBTEWHL32161  8
36IBTEBLDT@@EN00232164  8
36IBTEBLDTLBTBWHL12205  8
36IBTEBLDTLBTTHEN22248  8
36IBTEBLDTLBTELSE22254  8
36IBTEBLDTLBTTHEN32291  8
36IBTEBLDTLBTELSE32297  8
36IBTEBLDTLBTELSE42319  8
36IBTEBLDTLBTENIF92325  8
36IBTEBLDTLBTEWHL12328  8
36IBTEBLDTLBTBWHL22328  8
36IBTEBLDTLBTEWHL22343  8
36IBTEBLDT@@EL00232343  8
36IBTEBLDTSETUP   2351  8
36IBTEBLDT@@DL00322380  8
36IBTEBLDT@@EN00332398  8
```

```
37EBTSTANDEBTSTAND2427  8
37EBTSTANDLSTELSE52448  8
37EBTSTANDLSTELSE72461  8
37EBTSTANDLSTENDF82467  8
37EBTSTANDLSTREPT12474  8
38EBTDIEVAEBTDIEVA2490  8
38EBTDIEVALDIELSE52511  8
38EBTDIEVALDIELSE72524  8
38EBTDIEVALDIENDF82530  8
38EBTDIEVALDIREPT12537  8
39EBTUPADSEBTUPADS2553  8
39EBTUPADSLUPELSE52573  8
39EBTUPADSLUPELSE72585  8
39EBTUPADSLUPENDF82590  8
40EBTUPADDEBTUPADD2602  8
40EBTUPADDLUPELSE42622  8
40EBTUPADDLUPELSE62634  8
40EBTUPADDLUPENDF92639  8
41IPTEPARMIPTEPARM   0  9
42IEVEADDRIEVEADDR  10  9
43IKBEKBDTIKBEKBDT  10  9
44IPTEPROCIPTEPROC1313  9
44IPTEPROCLPTWHIL01355  9
44IPTEPROCLPTELSE51384  9
44IPTEPROCLPTELSE71399  9
44IPTEPROC@@EN00051418  9
44IPTEPROC@@EL00051421  9
44IPTEPROCLPTENDF81421  9
44IPTEPROCLPTENDW91442  9
45EPTNORMPEPTNORMP1447  9
45EPTNORMP$$0015  1478  9
45EPTNORMP$$0022  1503  9
45EPTNORMP@@EN00201513  9
45EPTNORMP@@EL00201524  9
45EPTNORMP@@EN00131527  9
45EPTNORMP@@EL00131538  9
46EPTLOCKPEPTLOCKP1539  9
46EPTLOCKP$$0033  1570  9
46EPTLOCKP$$0040  1599  9
46EPTLOCKP$$0047  1624  9
46EPTLOCKP@@EN00451631  9
46EPTLOCKP@@EN00381634  9
46EPTLOCKP$$0056  1654  9
46EPTLOCKP$$0063  1682  9
46EPTLOCKP@@EN00611689  9
46EPTLOCKP@@EN00541689  9
46EPTLOCKP@@EL00381689  9
46EPTLOCKP@@EN00311689  9
46EPTLOCKPLPTLNDF81709  9
47EPTSHIFTEPTSHIFT1710  9
47EPTSHIFT$$0074  1741  9
47EPTSHIFT$$0081  1770  9
47EPTSHIFT$$0088  1796  9
47EPTSHIFT$$0095  1825  9
```

```
47EPTSHIFT$$0102  1850  9
47EPTSHIFT@@EN01001857  9
47EPTSHIFT@@EN00931857  9
47EPTSHIFT@@EN00861857  9
47EPTSHIFT@@EN00791857  9
47EPTSHIFT@@EN00721857  9
47EPTSHIFTLPTSNDF81877  9
48EPTSPACEEPTSPACE1878  9
48EPTSPACE$$0114  1909  9
48EPTSPACE$$0121  1935  9
48EPTSPACE$$0128  1964  9
48EPTSPACE$$0135  1990  9
48EPTSPACE@@EN01331997  9
48EPTSPACE@@EN01261997  9
48EPTSPACE@@EN01191997  9
48EPTSPACE@@EN01121997  9
48EPTSPACELPTSNDF92017  9
49IRKEPARMIRKEPARM    0 10
50IEVEADDRIEVEADDR   37 10
51IRKERKBTIRKERKBT   37 10
51IRKERKBT@@DL0007   65 10
51IRKERKBT@@DL0013   78 10
51IRKERKBT@@DL0015   96 10
51IRKERKBT@@DT0013  106 10
51IRKERKBT$$0023    130 10
52ISEEPARMISEEPARM    0 11
53IEVEADDRIEVEADDR    1 11
54IKBEKBDTIKBEKBDT    1 11
55ISEESNDEISEESNDE1304 11
55ISEESNDE@@DL00051332 11
55ISEESNDE@@EN00061341 11
55ISEESNDE@@EN00111354 11
55ISEESNDE@@EN00171364 11
55ISEESNDE@@EL00171366 11
55ISEESNDE@@EL00111366 11
55ISEESNDE@@EN00251385 11
55ISEESNDE@@EL00251386 11
55ISSESNDE@@EL00061389 11
56ISGEPARMISGEPARM    0 12
57IEVEADDRIEVEADDR    9 12
58IKBEKBDTIKBEKBDT    9 12
59ISGESNDGISGESNDG1312 12
59ISGESNDG@@DL00071365 12
59ISGESNDG@@DT00071393 12
60ITKEPARMITKEPARM    0 13
61IEVEADDRIEVEADDR    5 13
62IIOEAREAIIOEAREA    5 13
63IKBEKBDTIKBEKBDT    5 13
64ITKETESTITKETEST1308 13
64ITKETEST@@DL00101363 13
64ITKETEST@@EN00111381 13
64ITKETESTLTKBWHL11405 13
64ITKETESTLTKEWHL11433 13
64ITKETEST@@DL00261458 13
```

```
64ITKETEST@@EN00271476 13
64ITKETEST@@DL00431520 13
64ITKETEST@@EN00441538 13
64ITKETESTLTKBWHL21573 13
64ITKETESTLTKIFTH11597 13
64ITKETESTLTKIFEN11610 13
64ITKETESTLTKEWHL21616 13
64ITKETEST@@DL00631647 13
64ITKETEST@@EN00641665 13
64ITKETEST@@DL00781681 13
64ITKETESTLTKBWHL31706 13
64ITKETESTLTKIFTH21731 13
64ITKETESTLTKIFEN21744 13
64ITKETESTLTKEWHL31750 13
64ITKETEST@@DL00891783 13
64ITKETEST@@EN00901801 13
64ITKETEST@@DL01041817 13
64ITKETEST@@DL01141856 13
64ITKETEST@@EN01151874 13
64ITKETEST@@DL01311918 13
64ITKETEST@@EN01321936 13
64ITKETESTLTKBWHL41971 13
64ITKETESTLTKIFEL32012 13
64ITKETESTLTKIFEN32025 13
64ITKETESTLTKEWHL42031 13
64ITKETEST@@DL01502056 13
64ITKETEST@@EN01512074 13
64ITKETEST@@DL01622081 13
64ITKETESTLTKBWHL52106 13
64ITKETESTLTKIFEN42144 13
64ITKETESTLTKEWHL52150 13
64ITKETEST@@DL01722175 13
64ITKETEST@@EN01732193 13
64ITKETESTLTKBWHL62217 13
64ITKETESTLTKIFEN52251 13
64ITKETESTLTKEWHL62257 13
65ETKUPADSETKUPADS2266 13
65ETKUPADSLUPELSE52286 13
65ETKUPADSLUPELSE72298 13
65ETKUPADSLUPENDF82303 13
66ETKREADPETKREADP2315 13
66ETKREADP@@DL01872338 13
66ETKREADP@@DT01872338 13
66ETKREADPLTKREP012376 13
66ETKREADP@@EN01962394 13
66ETKREADPLTKIFEL12421 13
66ETKREADPLTKENIF12427 13
66ETKREADPLTKEMULT2469 13
66ETKREADPLTKMULT02473 13
66ETKREADPLTKMULT12485 13
66ETKREADPLTKDONE92490 13
66ETKREADPLTKENIF32500 13
66ETKREADPLTKENIF62525 13
67IUCEPARMIUCEPARM    0 14
```

```
68IEVEADDRIEVEADR     0 14
69IIOEAREAIIOEAREA    0 14
70IUCEUNCLIUCEUNCL    0 14
70IUCEUNCL@@DL0011   57 14
70IUCEUNCL@@EN0012   75 14
70IUCEUNCL@@DL0027  111 14
70IUCEUNCL@@EN0028  129 14
71IAOEPARMIAOEPARM    0 15
72ICOEICOTICOEICO2   12 15
73IEVEADDRIEVEADDR  303 15
74IKBEKBDTIKBEKBDT  303 15
75IAOEAOFFIAOEAOFF1606 15
75IAOEAOFFLBTBWHL11642 15
75IAOEAOOFRIGTSPOT1669 15
75IAOEAOFFBRANCHPT1672 15
75IAOEAOFFLSTELSE51749 15
75IAOEAOFFLSTELSE71799 15
75IAOEAOFFLSTENDF81806 15
75IAOEAOFFLBTADDUM1826 15
75IAOEAOFFLBTENIF91863 15
75IAOEAOFFLBTEWHL11876 15
```

<u>OUTPUT</u> <u>FILE</u>

```
 1   1IAOEPARM1
 1   2IAOEPARM3
 2   3ICOEICOT1
 2   4ICOEICOT3
 3   5IEVEADDR1
 3   6IEVEADDR3
 4   7IKBEKBDT1
 4   8IKBEKBDT3
 5   9IAOEAOFF1
 5  10IAOEAOFF3
 6  11ICCEPARM1
 6  12ICCEPARM3
 7  13IEVEADDR1
 7  14IEVEADDR3
 8  15IIOEAREA1
 8  16IIOEAREA3
 9  17ICCECLMP1ITEEABRT
26  18ITEEABRT2ICCECLMP
 9  19ICCECLMP1IWTEWAITY
10  20IEXEPARM1
10  21IEXEPARM3
11  22IEVEADDR1
11  23IEVEADDR3
12  24IIOEAREA1
12  25IIOEAREA3
13  26IEXEEXER1ITEEABRT
26  27ITEEABRT2IEXEEXER
13  28IEXEEXER1IWTEWAITY
13  29IEXEEXER1ITEEABRT
26  30ITEEABRT2IEXEEXER
13  31IEXEEXER1IWTEWAITY
14  32IITEPARM1
14  33IITEPARM3
15  34IEVEADDR1
15  35IEVEADDR3
16  36IIOEAREA1
16  37IIOEAREA3
17  38IITEINIT1
17  39IITEINIT3
19  40IEVEADDR1
19  41IEVEADDR3
20  42IIOEAREA1
20  43IIOEAREA3
21  44IKBEKBDT1
21  45IKBEKBDT3
22  46IMDEMAIN1IITEINIT
17  47IITEINIT2IMDEMAIN
22  48IMDEMAIN1IDMEDISPY
22  49IMDEMAIN1IWTEWAITY
22  50IMDEMAIN1IWTEWAITY
22  51IMDEMAIN1IDMEDISPY
```

```
22 52IMDEMAIN1IBOEBCOT
31 53IBOEBCOT2IMDEMAIN
22 54IMDEMAIN1IDMEDISPY
22 55IMDEMAIN1IRKERKBT
51 56IRKERKBT2IMDEMAIN
22 57IMDEMAIN1IDMEDISPY
22 58IMDEMAIN1IBTEBLDT
36 59IBTEBLDT2IMDEMAIN
22 60IMDEMAIN1IDMEDISPY
22 61IMDEMAIN1ICCECLMP
 9 62ICCECLMP2IMDEMAIN
22 63IMDEMAIN1IUCEUNCL
70 64IUCEUNCL2IMDEMAIN
22 65IMDEMAIN1IWSEWRITY
22 66IMDEMAIN1IWTEWAITY
22 67IMDEMAIN1IWTEWAITY
22 68IMDEMAIN1IDMEDISPY
22 69IMDEMAIN1IEXEEXER
13 70IEXEEXER2IMDEMAIN
22 71IMDEMAIN1IDMEDISPY
22 72IMDEMAIN1ITKETEST
64 73ITKETEST2IMDEMAIN
22 74IMDEMAIN1IDMEDISPY
22 75IMDEMAIN1IAOEAOFF
 5 76IAOEAOFF2IMDEMAIN
22 77IMDEMAIN1IDMEDISPY
22 78IMDEMAIN1IPTEPROC
44 79IPTEPROC2IMDEMAIN
22 80IMDEMAIN1IDMEDISPY
22 81IMDEMAIN1IUCEUNCL
70 82IUCEUNCL2IMDEMAIN
22 83IMDEMAIN1IDMEDISPY
22 84IMDEMAIN1IWSEWRITY
22 85IMDEMAIN1IDMEDISPY
22 86IMDEMAIN1IWSEWRITY
22 87IMDEMAIN1IDMEDISPY
22 88IMDEMAIN1ISGESNDG
59 89ISGESNDG2IMDEMAIN
22 90IMDEMAIN1IDMEDISPY
22 91IMDEMAIN1ISEESNDE
55 92ISEESNDE2IMDEMAIN
22 93IMDEMAIN1IDMEDISPY
22 94IMDEMAIN1IWSEWRITY
22 95IMDEMAIN1IDMEDISPY
22 96IMDEMAIN1ISGESNDG
59 97ISGESNDG2IMDEMAIN
22 98IMDEMAIN1IDMEDISPY
22 99IMDEMAIN1IUCEUNCL
70100IUCEUNCL2IMDEMAIN
22101IMDEMAIN1IWSEWRITY
22102IMDEMAIN1IDMEDISPY
22103IMDEMAIN1IUCEUNCL
70104IUCEUNCL2IMDEMAIN
22105IMDEMAIN1IWSEWRITY
```

```
22106IMDEMAIN1IDMEDISPY
22107IMDEMAIN1IWTEWAITY
23108ITEEPARM1
23109ITEEPARM3
24110IEVEADDR1
24111IEVEADDR3
25112IIOEAREA1
25113IIOEAREA3
26114ITEEABRT1IDMEDISPY
27115IBOEPARM1
27116IBOEPARM3
28117ICOEICOT1
28118ICOEICOT3
29119IEVEADDR1
29120IEVEADDR3
30121IIOEAREA1
30122IIOEAREA3
31123IBOEBCOT1IWTEWAITY
31124IBOEBCOT1IWTEWAITY
31125IBOEBCOT1ITEEABRT
26126ITEEABRT2IBOEBCOT
31127IBOEBCOT1ITEEABRT
26128ITEEABRT2IBOEBCOT
31129IBOEBCOT1IWTEWAITY
32130IBTEPARM1
32131IBTEPARM3
33132IEVEADDR1
33133IEVEADDR3
34134IIOEAREA1
34135IIOEAREA3
35136IKBEKBDT1
35137IKBEKBDT3
36138IBTEBLDT1EBTSTAND
37139EBTSTAND2IBTEBLDT
36140IBTEBLDT1EBTDIEVA
38141EBTDIEVA2IBTEBLDT
36142IBTEBLDT1EBTSTAND
37143EBTSTAND2IBTEBLDT
36144IBTEBLDT1EBTUPADS
39145EBTUPADS2IBTEBLDT
36146IBTEBLDT1EBTUPADD
40147EBTUPADD2IBTEBLDT
36148IBTEBLDT1EBTDIEVA
38149EBTDIEVA2IBTEBLDT
36150IBTEBLDT1EBTUPADS
39151EBTUPADS2IBTEBLDT
36152IBTEBLDT1EBTSTAND
37153EBTSTAND2IBTEBLDT
36154IBTEBLDT1ITEEABRT
26155ITEEABRT2IBTEBLDT
37156EBTSTAND1
37157EBTSTAND3
38158EBTDIEVA1
38159EBTDIEVA3
```

39160EBTUPADS1
39161EBTUPADS3
40162EBTUPADD1
40163EBTUPADD3
41164IPTEPARM1
41165IPTEPARM3
42166IEVEADDR1
42167IEVEADDR3
43168IKBEKBDT1
43169IKBEKBDT3
44170IPTEPROC1EPTNORMP
45171EPTNORMP2IPTEPROC
44172IPTEPROC1EPTLOCKP
46173EPTLOCKP2IPTEPROC
44174IPTEPROC1EPTSPACE
48175EPTSPACE2IPTEPROC
44176IPTEPROC1EPTSHIFT
47177EPTSHIFT2IPTEPROC
45178EPTNORMP1
45179EPTNORMP3
46180EPTLOCKP1
46181EPTLOCKP3
47182EPTSHIFT1
47183EPTSHIFT3
48184EPTSPACE1
48185EPTSPACE3
49186IRKEPARM1
49187IRKEPARM3
50188IEVEADDR1
50189IEVEADDR3
51190IRKERKBT1IWSEWRITY
51191IRKERKBT1IDMEDISPY
51192IRKERKBT1IWSEWRITY
52193ISEEPARM1
52194ISEEPARM3
53195IEVEADDR1
53196IEVEADDR3
54197IKBEKBDT1
54198IKBEKBDT3
55199ISEESNDE1IWSEWRITY
55200ISEESNDE1IWSEWRITY
56201ISGEPARM1
56202ISGEPARM3
57203IEVEADDR1
57204IEVEADDR3
58205IKBEKBDT1
58206IKBEKBDT3
59207ISGESNDG1IWSEWRITY
59208ISGESNDG1IWSEWRITY
59209ISGESNDG1IWSEWRITY
60210ITKEPARM1
60211ITKEPARM3
61212IEVEADDR1
61213IEVEADDR3

62214IIOEAREA1
62215IIOEAREA3
63216IKBEKBDT1
63217IKBEKBDT3
64218ITKETEST1ITEEABRT
26219ITEEABRT2ITKETEST
64220ITKETEST1IWTEWAITY
64221ITKETEST1ETKREADP
66222ETKREADP2ITKETEST
64223ITKETEST1ETKUPADS
65224ETKUPADS2ITKETEST
64225ITKETEST1ITEEABRT
26226ITEEABRT2ITKETEST
64227ITKETEST1ITEEABRT
26228ITEEABRT2ITKETEST
64229ITKETEST1ETKREADP
66230ETKREADP2ITKETEST
64231ITKETEST1ETKUPADS
65232ETKUPADS2ITKETEST
64233ITKETEST1ITEEABRT
26234ITEEABRT2ITKETEST
64235ITKETEST1ETKREADP
66236ETKREADP2ITKETEST
64237ITKETEST1ETKUPADS
65238ETKUPADS2ITKETEST
64239ITKETEST1ITEEABRT
26240ITEEABRT2ITKETEST
64241ITKETEST1ITEEABRT
26242ITEEABRT2ITKETEST
64243ITKETEST1ITEEABRT
26244ITEEABRT2ITKETEST
64245ITKETEST1ETKREADP
66246ETKREADP2ITKETEST
64247ITKETEST1ETKREADP
66248ETKREADP2ITKETEST
64249ITKETEST1ETKUPADS
65250ETKUPADS2ITKETEST
64251ITKETEST1ITEEABRT
26252ITEEABRT2ITKETEST
64253ITKETEST1ETKREADP
66254ETKREADP2ITKETEST
64255ITKETEST1ETKUPADS
65256ETKUPADS2ITKETEST
64257ITKETEST1ITEEABRT
26258ITEEABRT2ITKETEST
64259ITKETEST1ETKREADP
66260ETKREADP2ITKETEST
64261ITKETEST1ETKUPADS
65262ETKUPADS2ITKETEST
65263ETKUPADS1
65264ETKUPADS3
66265ETKREADP1IWTEWAITY
66266ETKREADP1ITEEABRT
26267ITEEABRT2ETKREADP

```
66268ETKREADP1IWTEWAITY
67269IUCEPARM1
67270IUCEPARM3
67271IUCEPARM1
67272IUCEPARM3
68273IEVEADDR1
68274IEVEADDR3
69275IIOEAREA1
69276IIOEAREA3
70277IUCEUNCL1ITEEABRT
26278ITEEABRT2IUCEUNCL
70279IUCEUNCL1ITEEABRT
26280ITEEABRT2IUCEUNCL
71281IAOEPARM1
71282IAOEPARM3
72283ICOEICOT1
72284ICOEICOT3
73285IEVEADDR1
73286IEVEADDR3
74287IKBEKBDT1
74288IKBEKBDT3
75289IAOEAOFF1IBOEBCOT
31290IBOEBCOT2IAOEAOFF
75291IAOEAOFF1IBOEBCOT
31292IBOEBCOT2IAOEAOFF
75293IAOEAOFF1IBOEBCOT
31294IBOEBCOT2IAOEAOFF
75295IAOEAOFF1IBOEBCOT
31296IBOEBCOT2IAOEAOFF
75297IAOEAOFF1IBOEBCOT
31298IBOEBCOT2IAOEAOFF
75299IAOEAOFF1IBOEBCOT
31300IBOEBCOT2IAOEAOFF
75301IAOEAOFF1IBTEBLDT
36302IBTEBLDT2IAOEAOFF
75303IAOEAOFF1IBTEBLDT
36304IBTEBLDT2IAOEAOFF
```

<u>SORTOUT</u> <u>FILE</u>

```
 1   1IAOEPARM1
 1   2IAOEPARM3
 2   3ICOEICOT1
 2   4ICOEICOT3
 3   5IEVEADDR1
 3   6IEVEADDR3
 4   7IKBEKBDT1
 4   8IKBEKBDT3
 5   9IAOEAOFF1
 5  76IAOEAOFF2IMDEMAIN
 5  10IAOEAOFF3
 6  11ICCEPARM1
 6  12ICCEPARM3
 7  13IEVEADDR1
 7  14IEVEADDR3
 8  15IIOEAREA1
 8  16IIOEAREA3
 9  17ICCECLMP1ITEEABRT
 9  19ICCECLMP1IWTEWAITY
 9  62ICCECLMP2IMDEMAIN
10  20IEXEPARM1
10  21IEXEPARM3
11  22IEVEADDR1
11  23IEVEADDR3
12  24IIOEAREA1
12  25IIOEAREA3
13  26IEXEEXER1ITEEABRT
13  28IEXEEXER1IWTEWAITY
13  29IEXEEXER1ITEEABRT
13  31IEXEEXER1IWTEWAITY
13  70IEXEEXER2IMDEMAIN
14  32IITEPARM1
14  33IITEPARM3
15  34IEVEADDR1
15  35IEVEADDR3
16  36IIOEAREA1
16  37IIOEAREA3
17  38IITEINIT1
17  47IITEINIT2IMDEMAIN
17  39IITEINIT3
19  40IEVEADDR1
19  41IEVEADDR3
20  42IIOEAREA1
20  43IIOEAREA3
21  44IKBEKBDT1
21  45IKBEKBDT3
22  46IMDEMAIN1IITEINIT
22  48IMDEMAIN1IDMEDISPY
22  49IMDEMAIN1IWTEWAITY
22  50IMDEMAIN1IWTEWAITY
22  51IMDEMAIN1IDMEDISPY
```

```
22 52IMDEMAIN1IBOEBCOT
22 54IMDEMAIN1IDMEDISPY
22 55IMDEMAIN1IRKERKBT
22 57IMDEMAIN1IDMEDISPY
22 58IMDEMAIN1IBTEBLDT
22 60IMDEMAIN1IDMEDISPY
22 61IMDEMAIN1ICCECLMP
22 63IMDEMAIN1IUCEUNCL
22 65IMDEMAIN1IWSEWRITY
22 66IMDEMAIN1IWTEWAITY
22 67IMDEMAIN1IWTEWAITY
22 68IMDEMAIN1IDMEDISPY
22 69IMDEMAIN1IEXEEXER
22 71IMDEMAIN1IDMEDISPY
22 72IMDEMAIN1ITKETEST
22 74IMDEMAIN1IDMEDISPY
22 75IMDEMAIN1IAOEAOFF
22 77IMDEMAIN1IDMEDISPY
22 78IMDEMAIN1IPTEPROC
22 80IMDEMAIN1IDMEDISPY
22 81IMDEMAIN1IUCEUNCL
22 83IMDEMAIN1IDMEDISPY
22 84IMDEMAIN1IWSEWRITY
22 85IMDEMAIN1IDMEDISPY
22 86IMDEMAIN1IWSEWRITY
22 87IMDEMAIN1IDMEDISPY
22 88IMDEMAIN1ISGESNDG
22 90IMDEMAIN1IDMEDISPY
22 91IMDEMAIN1ISEESNDE
22 93IMDEMAIN1IDMEDISPY
22 94IMDEMAIN1IWSEWRITY
22 95IMDEMAIN1IDMEDISPY
22 96IMDEMAIN1ISGESNDG
22 98IMDEMAIN1IDMEDISPY
22 99IMDEMAIN1IUCEUNCL
22101IMDEMAIN1IWSEWRITY
22102IMDEMAIN1IDMEDISPY
22103IMDEMAIN1IUCEUNCL
22105IMDEMAIN1IWSEWRITY
22106IMDEMAIN1IDMEDISPY
22107IMDEMAIN1IWTEWAITY
23108ITEEPARM1
23109ITEEPARM3
24110IEVEADDR1
24111IEVEADDR3
25112IIOEAREA1
25113IIOEAREA3
26114ITEEABRT1IDMEDISPY
26 18ITEEABRT2ICCECLMP
26 27ITEEABRT2IEXEEXER
26 30ITEEABRT2IEXEEXER
26126ITEEABRT2IBOEBCOT
26128ITEEABRT2IBOEBCOT
26155ITEEABRT2IBTEBLDT
```

```
26219ITEEABRT2ITKETEST
26226ITEEABRT2ITKETEST
26228ITEEABRT2ITKETEST
26234ITEEABRT2ITKETEST
26240ITEEABRT2ITKETEST
26242ITEEABRT2ITKETEST
26244ITEEABRT2ITKETEST
26252ITEEABRT2ITKETEST
26258ITEEABRT2ITKETEST
26267ITEEABRT2ETKREADP
26278ITEEABRT2IUCEUNCL
26280ITEEABRT2IUCEUNCL
27115IBOEPARM1
27116IBOEPARM3
28117ICOEICOT1
28118ICOEICOT3
29119IEVEADDR1
29120IEVEADDR3
30121IIOEAREA1
30122IIOEAREA3
31123IBOEBCOT1IWTEWAITY
31124IBOEBCOT1IWTEWAITY
31125IBOEBCOT1ITEEABRT
31127IBOEBCOT1ITEEABRT
31129IBOEBCOT1IWTEWAITY
31 53IBOEBCOT2IMDEMAIN
31290IBOEBCOT2IAOEAOFF
31292IBOEBCOT2IAOEAOFF
31294IBOEBCOT2IAOEAOFF
31296IBOEBCOT2IAOEAOFF
31298IBOEBCOT2IAOEAOFF
31300IBOEBCOT2IAOEAOFF
32130IBTEPARM1
32131IBTEPARM3
33132IEVEADDR1
33133IEVEADDR3
34134IIOEAREA1
34135IIOEAREA3
35136IKBEKBDT1
35137IKBEKBDT3
36138IBTEBLDT1EBTSTAND
36140IBTEBLDT1EBTDIEVA
36142IBTEBLDT1EBTSTAND
36144IBTEBLDT1EBTUPADS
36146IBTEBLDT1EBTUPADD
36148IBTEBLDT1EBTDIEVA
36150IBTEBLDT1EBTUPADS
36152IBTEBLDT1EBTSTAND
36154IBTEBLDT1ITEEABRT
36 59IBTEBLDT2IMDEMAIN
36302IBTEBLDT2IAOEAOFF
36304IBTEBLDT2IAOEAOFF
37156EBTSTAND1
37139EBTSTAND2IBTEBLDT
```

```
37143EBTSTAND2IBTEBLDT
37153EBTSTAND2IBTEBLDT
37157EBTSTAND3
38158EBTDIEVA1
38141EBTDIEVA2IBTEBLDT
38149EBTDIEVA2IBTEBLDT
38159EBTDIEVA3
39160EBTUPADS1
39145EBTUPADS2IBTEBLDT
39151EBTUPADS2IBTEBLDT
39161EBTUPADS3
40162EBTUPADD1
40147EBTUPADD2IBTEBLDT
40163EBTUPADD3
41164IPTEPARM1
41165IPTEPARM3
42166IEVEADDR1
42167IEVEADDR3
43168IKBEKBDT1
43169IKBEKBDT3
44170IPTEPROC1EPTNORMP
44172IPTEPROC1EPTLOCKP
44174IPTEPROC1EPTSPACE
44176IPTEPROC1EPTSHIFT
44 79IPTEPROC2IMDEMAIN
45178EPTNORMP1
45171EPTNORMP2IPTEPROC
45179EPTNORMP3
46180EPTLOCKP1
46173EPTLOCKP2IPTEPROC
46181EPTLOCKP3
47182EPTSHIFT1
47177EPTSHIFT2IPTEPROC
47183EPTSHIFT3
48184EPTSPACE1
48175EPTSPACE2IPTEPROC
48185EPTSPACE3
49186IRKEPARM1
49187IRKEPARM3
50188IEVEADDR1
50189IEVEADDR3
51190IRKERKBT1IWSEWRITY
51191IRKERKBT1IDMEDISPY
51192IRKERKBT1IWSEWRITY
51 56IRKERKBT2IMDEMAIN
52193ISEEPARM1
52194ISEEPARM3
53195IEVEADDR1
53196IEVEADDR3
54197IKBEKBDT1
54198IKBEKBDT3
55199ISEESNDE1IWSEWRITY
55200ISEESNDE1IWSEWRITY
55 92ISEESNDE2IMDEMAIN
```

```
56201ISGEPARM1
56202ISGEPARM3
57203IEVEADDR1
57204IEVEADDR3
58205IKBEKBDT1
58206IKBEKBDT3
59207ISGESNDG1IWSEWRITY
59208ISGESNDG1IWSEWRITY
59209ISGESNDG1IWSEWRITY
59 89ISGESNDG2IMDEMAIN
59 97ISGESNDG2IMDEMAIN
60210ITKEPARM1
60211ITKEPARM3
61212IEVEADDR1
61213IEVEADDR3
62214IIOEAREA1
62215IIOEAREA3
63216IKBEKBDT1
63217IKBEKBDT3
64218ITKETEST1ITEEABRT
64220ITKETEST1IWTEWAITY
64221ITKETEST1ETKREADP
64223ITKETEST1ETKUPADS
64225ITKETEST1ITEEABRT
64227ITKETEST1ITEEABRT
64229ITKETEST1ETKREADP
64231ITKETEST1ETKUPADS
64233ITKETEST1ITEEABRT
64235ITKETEST1ETKREADP
64237ITKETEST1ETKUPADS
64239ITKETEST1ITEEABRT
64241ITKETEST1ITEEABRT
64243ITKETEST1ITEEABRT
64245ITKETEST1ETKREADP
64247ITKETEST1ETKREADP
64249ITKETEST1ETKUPADS
64251ITKETEST1ITEEABRT
64253ITKETEST1ETKREADP
64255ITKETEST1ETKUPADS
64257ITKETEST1ITEEABRT
64259ITKETEST1ETKREADP
64261ITKETEST1ETKUPADS
64 73ITKETEST2IMDEMAIN
65263ETKUPADS1
65224ETKUPADS2ITKETEST
65232ETKUPADS2ITKETEST
65238ETKUPADS2ITKETEST
65250ETKUPADS2ITKETEST
65256ETKUPADS2ITKETEST
65262ETKUPADS2ITKETEST
65264ETKUPADS3
66265ETKREADP1IWTEWAITY
66266ETKREADP1ITEEABRT
66268ETKREADP1IWTEWAITY
```

```
66222ETKREADP2ITKETEST
66230ETKREADP2ITKETEST
66236ETKREADP2ITKETEST
66246ETKREADP2ITKETEST
66248ETKREADP2ITKETEST
66254ETKREADP2ITKETEST
66260ETKREADP2ITKETEST
67269IUCEPARM1
67271IUCEPARM1
67270IUCEPARM3
67272IUCEPARM3
68273IEVEADDR1
68274IEVEADDR3
69275IIOEAREA1
69276IIOEAREA3
70277IUCEUNCL1ITEEABRT
70279IUCEUNCL1ITEEABRT
70 64IUCEUNCL2IMDEMAIN
70 82IUCEUNCL2IMDEMAIN
70100IUCEUNCL2IMDEMAIN
70104IUCEUNCL2IMDEMAIN
71281IAOEPARM1
71282IAOEPARM3
72283ICOEICOT1
72284ICOEICOT3
73285IEVEADDR1
73286IEVEADDR3
74287IKBEKBDT1
74288IKBEKBDT3
75289IAOEAOFF1IBOEBCOT
75291IAOEAOFF1IBOEBCOT
75293IAOEAOFF1IBOEBCOT
75295IAOEAOFF1IBOEBCOT
75297IAOEAOFF1IBOEBCOT
75299IAOEAOFF1IBOEBCOT
75301IAOEAOFF1IBTEBLDT
75303IAOEAOFF1IBTEBLDT
```

APPENDIX D

dBASE OUTPUT REPORT

DATASET: TSS2525.CSECT.DATA

CSECT HIEARARCHY

IAOEPARM

CSECT IAOEPARM DOES NOT CALL ANY CSECT
CSECT IAOEPARM IS NOT CALLED BY ANY CSECT
CSECT IAOEPARM DOES NOT JUMP TO ANY CSECT
CSECT IAOEPARM IS NOT JUMPED TO BY ANY CSECT

ICOEICOT

CSECT ICOEICOT DOES NOT CALL ANY CSECT
CSECT ICOEICOT IS NOT CALLED BY ANY CSECT
CSECT ICOEICOT DOES NOT JUMP TO ANY CSECT
CSECT ICOEICOT IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

IAOEAOFF

CSECT IAOEAOFF DOES NOT CALL ANY CSECT
CSECT IAOEAOFF IS CALLED BY CSECT IMDEMAIN
CSECT IAOEAOFF DOES NOT JUMP TO ANY CSECT
CSECT IAOEAOFF IS NOT JUMPED TO BY ANY CSECT

ICCEPARM

CSECT ICCEPARM DOES NOT CALL ANY CSECT

DATASET: TSS2525.CSECT.DATA

CSECT ICCEPARM IS NOT CALLED BY ANY CSECT
CSECT ICCEPARM DOES NOT JUMP TO ANY CSECT
CSECT ICCEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

ICCECLMP

CSECT ICCECLMP CALLS CSECT ITEEABRT
UNRESOLVED LABEL IWTEWAIT
CSECT ICCECLMP IS CALLED BY CSECT IMDEMAIN
CSECT ICCECLMP DOES NOT JUMP TO ANY CSECT
CSECT ICCECLMP IS NOT JUMPED TO BY ANY CSECT

IEXEPARM

CSECT IEXEPARM DOES NOT CALL ANY CSECT
CSECT IEXEPARM IS NOT CALLED BY ANY CSECT
CSECT IEXEPARM DOES NOT JUMP TO ANY CSECT
CSECT IEXEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT

DATASET: TSS2525.CSECT.DATA

CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

IEXEEXER

```
CSECT IEXEEXER CALLS CSECT ITEEABRT
                     UNRESOLVED LABEL IWTEWAIT
                     CSECT ITEEABRT
                     UNRESOLVED LABEL IWTEWAIT
CSECT IEXEEXER IS CALLED BY CSECT IMDEMAIN
CSECT IEXEEXER DOES NOT JUMP TO ANY CSECT
CSECT IEXEEXER IS NOT JUMPED TO BY ANY CSECT
```

IITEPARM

CSECT IITEPARM DOES NOT CALL ANY CSECT
CSECT IITEPARM IS NOT CALLED BY ANY CSECT
CSECT IITEPARM DOES NOT JUMP TO ANY CSECT
CSECT IITEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

IITEINIT

CSECT IITEINIT DOES NOT CALL ANY CSECT
CSECT IITEINIT IS CALLED BY CSECT IMDEMAIN
CSECT IITEINIT DOES NOT JUMP TO ANY CSECT
CSECT IITEINIT IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT

DATASET: TSS2525.CSECT.DATA

```
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA
________

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT
________

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

IMDEMAIN
________

CSECT IMDEMAIN CALLS CSECT IITEINIT
                     UNRESOLVED LABEL IDMEDISP
                     UNRESOLVED LABEL IWTEWAIT
                     UNRESOLVED LABEL IWTEWAIT
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IBOEBCOT
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IRKERKBT
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IBTEBLDT
                     UNRESOLVED LABEL IDMEDISP
                     CSECT ICCECLMP
                     CSECT IUCEUNCL
                     UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IWTEWAIT
                     UNRESOLVED LABEL IWTEWAIT
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IEXEEXER
                     UNRESOLVED LABEL IDMEDISP
                     CSECT ITKETEST
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IAOEAOFF
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IPTEPROC
                     UNRESOLVED LABEL IDMEDISP
                     CSECT IUCEUNCL
                     UNRESOLVED LABEL IDMEDISP
                     UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IDMEDISP
                     UNRESOLVED LABEL IWSEWRIT
```

DATASET: TSS2525.CSECT.DATA

```
                    UNRESOLVED LABEL IDMEDISP
                    CSECT ISGESNDG
                    UNRESOLVED LABEL IDMEDISP
                    CSECT ISEESNDE
                    UNRESOLVED LABEL IDMEDISP
                    UNRESOLVED LABEL IWSEWRIT
                    UNRESOLVED LABEL IDMEDISP
                    CSECT ISGESNDG
                    UNRESOLVED LABEL IDMEDISP
                    CSECT IUCEUNCL
                    UNRESOLVED LABEL IWSEWRIT
                    UNRESOLVED LABEL IDMEDISP
                    CSECT IUCEUNCL
                    UNRESOLVED LABEL IWSEWRIT
                    UNRESOLVED LABEL IDMEDISP
                    UNRESOLVED LABEL IWTEWAIT
CSECT IMDEMAIN IS NOT CALLED BY ANY CSECT
CSECT IMDEMAIN DOES NOT JUMP TO ANY CSECT
CSECT IMDEMAIN IS NOT JUMPED TO BY ANY CSECT

ITEEPARM
________

CSECT ITEEPARM DOES NOT CALL ANY CSECT
CSECT ITEEPARM IS NOT CALLED BY ANY CSECT
CSECT ITEEPARM DOES NOT JUMP TO ANY CSECT
CSECT ITEEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA
________

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

ITEEABRT
________

CSECT ITEEABRT CALLS UNRESOLVED LABEL IDMEDISP
CSECT ITEEABRT IS CALLED BY CSECT ICCECLMP
                            CSECT IEXEEXER
                            CSECT IEXEEXER
```

DATASET: TSS2525.CSECT.DATA

```
                                        CSECT IBOEBCOT
                                        CSECT IBOEBCOT
                                        CSECT IBTEBLDT
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ITKETEST
                                        CSECT ETKREADP
                                        CSECT IUCEUNCL
                                        CSECT IUCEUNCL
CSECT ITEEABRT DOES NOT JUMP TO ANY CSECT
CSECT ITEEABRT IS NOT JUMPED TO BY ANY CSECT
```

IBOEPARM

CSECT IBOEPARM DOES NOT CALL ANY CSECT
CSECT IBOEPARM IS NOT CALLED BY ANY CSECT
CSECT IBOEPARM DOES NOT JUMP TO ANY CSECT
CSECT IBOEPARM IS NOT JUMPED TO BY ANY CSECT

ICOEICOT

CSECT ICOEICOT DOES NOT CALL ANY CSECT
CSECT ICOEICOT IS NOT CALLED BY ANY CSECT
CSECT ICOEICOT DOES NOT JUMP TO ANY CSECT
CSECT ICOEICOT IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

IBOEBCOT

DATASET: TSS2525.CSECT.DATA

```
CSECT IBOEBCOT CALLS UNRESOLVED LABEL IWTEWAIT
                     UNRESOLVED LABEL IWTEWAIT
                     CSECT ITEEABRT
                     CSECT ITEEABRT
                     UNRESOLVED LABEL IWTEWAIT
CSECT IBOEBCOT IS CALLED BY CSECT IMDEMAIN
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
CSECT IBOEBCOT DOES NOT JUMP TO ANY CSECT
CSECT IBOEBCOT IS NOT JUMPED TO BY ANY CSECT
```

IBTEPARM

```
CSECT IBTEPARM DOES NOT CALL ANY CSECT
CSECT IBTEPARM IS NOT CALLED BY ANY CSECT
CSECT IBTEPARM DOES NOT JUMP TO ANY CSECT
CSECT IBTEPARM IS NOT JUMPED TO BY ANY CSECT
```

IEVEADDR

```
CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT
```

IIOEAREA

```
CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT
```

IKBEKBDT

```
CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT
```

IBTEBLDT

```
CSECT IBTEBLDT CALLS CSECT EBTSTAND
```

DATASET: TSS2525.CSECT.DATA

```
                          CSECT EBTDIEVA
                          CSECT EBTSTAND
                          CSECT EBTUPADS
                          CSECT EBTUPADD
                          CSECT EBTDIEVA
                          CSECT EBTUPADS
                          CSECT EBTSTAND
                          CSECT ITEEABRT
CSECT IBTEBLDT IS CALLED BY CSECT IMDEMAIN
                            CSECT IAOEAOFF
                            CSECT IAOEAOFF
CSECT IBTEBLDT DOES NOT JUMP TO ANY CSECT
CSECT IBTEBLDT IS NOT JUMPED TO BY ANY CSECT

EBTSTAND
________

CSECT EBTSTAND DOES NOT CALL ANY CSECT
CSECT EBTSTAND IS CALLED BY CSECT IBTEBLDT
                            CSECT IBTEBLDT
                            CSECT IBTEBLDT
CSECT EBTSTAND DOES NOT JUMP TO ANY CSECT
CSECT EBTSTAND IS NOT JUMPED TO BY ANY CSECT

EBTDIEVA
________

CSECT EBTDIEVA DOES NOT CALL ANY CSECT
CSECT EBTDIEVA IS CALLED BY CSECT IBTEBLDT
                            CSECT IBTEBLDT
CSECT EBTDIEVA DOES NOT JUMP TO ANY CSECT
CSECT EBTDIEVA IS NOT JUMPED TO BY ANY CSECT

EBTUPADS
________

CSECT EBTUPADS DOES NOT CALL ANY CSECT
CSECT EBTUPADS IS CALLED BY CSECT IBTEBLDT
                            CSECT IBTEBLDT
CSECT EBTUPADS DOES NOT JUMP TO ANY CSECT
CSECT EBTUPADS IS NOT JUMPED TO BY ANY CSECT

EBTUPADD
________

CSECT EBTUPADD DOES NOT CALL ANY CSECT
CSECT EBTUPADD IS CALLED BY CSECT IBTEBLDT
CSECT EBTUPADD DOES NOT JUMP TO ANY CSECT
CSECT EBTUPADD IS NOT JUMPED TO BY ANY CSECT

IPTEPARM
________
```

DATASET: TSS2525.CSECT.DATA

CSECT IPTEPARM DOES NOT CALL ANY CSECT
CSECT IPTEPARM IS NOT CALLED BY ANY CSECT
CSECT IPTEPARM DOES NOT JUMP TO ANY CSECT
CSECT IPTEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

IPTEPROC

```
CSECT IPTEPROC CALLS CSECT EPTNORMP
                     CSECT EPTLOCKP
                     CSECT EPTSPACE
                     CSECT EPTSHIFT
CSECT IPTEPROC IS CALLED BY CSECT IMDEMAIN
CSECT IPTEPROC DOES NOT JUMP TO ANY CSECT
CSECT IPTEPROC IS NOT JUMPED TO BY ANY CSECT
```

EPTNORMP

CSECT EPTNORMP DOES NOT CALL ANY CSECT
CSECT EPTNORMP IS CALLED BY CSECT IPTEPROC
CSECT EPTNORMP DOES NOT JUMP TO ANY CSECT
CSECT EPTNORMP IS NOT JUMPED TO BY ANY CSECT

EPTLOCKP

CSECT EPTLOCKP DOES NOT CALL ANY CSECT
CSECT EPTLOCKP IS CALLED BY CSECT IPTEPROC
CSECT EPTLOCKP DOES NOT JUMP TO ANY CSECT
CSECT EPTLOCKP IS NOT JUMPED TO BY ANY CSECT

EPTSHIFT

DATASET: TSS2525.CSECT.DATA

```
CSECT EPTSHIFT DOES NOT CALL ANY CSECT
CSECT EPTSHIFT IS CALLED BY CSECT IPTEPROC
CSECT EPTSHIFT DOES NOT JUMP TO ANY CSECT
CSECT EPTSHIFT IS NOT JUMPED TO BY ANY CSECT

EPTSPACE
________

CSECT EPTSPACE DOES NOT CALL ANY CSECT
CSECT EPTSPACE IS CALLED BY CSECT IPTEPROC
CSECT EPTSPACE DOES NOT JUMP TO ANY CSECT
CSECT EPTSPACE IS NOT JUMPED TO BY ANY CSECT

IRKEPARM
________

CSECT IRKEPARM DOES NOT CALL ANY CSECT
CSECT IRKEPARM IS NOT CALLED BY ANY CSECT
CSECT IRKEPARM DOES NOT JUMP TO ANY CSECT
CSECT IRKEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IRKERKBT
________

CSECT IRKERKBT CALLS UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IDMEDISP
                     UNRESOLVED LABEL IWSEWRIT
CSECT IRKERKBT IS CALLED BY CSECT IMDEMAIN
CSECT IRKERKBT DOES NOT JUMP TO ANY CSECT
CSECT IRKERKBT IS NOT JUMPED TO BY ANY CSECT

ISEEPARM
________

CSECT ISEEPARM DOES NOT CALL ANY CSECT
CSECT ISEEPARM IS NOT CALLED BY ANY CSECT
CSECT ISEEPARM DOES NOT JUMP TO ANY CSECT
CSECT ISEEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________

CSECT IEVEADDR DOES NOT CALL ANY CSECT
```

DATASET: TSS2525.CSECT.DATA

```
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT
________

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

ISEESNDE
________

CSECT ISEESNDE CALLS UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IWSEWRIT
CSECT ISEESNDE IS CALLED BY CSECT IMDEMAIN
CSECT ISEESNDE DOES NOT JUMP TO ANY CSECT
CSECT ISEESNDE IS NOT JUMPED TO BY ANY CSECT

ISGEPARM
________

CSECT ISGEPARM DOES NOT CALL ANY CSECT
CSECT ISGEPARM IS NOT CALLED BY ANY CSECT
CSECT ISGEPARM DOES NOT JUMP TO ANY CSECT
CSECT ISGEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT
________

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

ISGESNDG
________

CSECT ISGESNDG CALLS UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IWSEWRIT
                     UNRESOLVED LABEL IWSEWRIT
```

DATASET: TSS2525.CSECT.DATA

```
CSECT ISGESNDG IS CALLED BY CSECT IMDEMAIN
                            CSECT IMDEMAIN
CSECT ISGESNDG DOES NOT JUMP TO ANY CSECT
CSECT ISGESNDG IS NOT JUMPED TO BY ANY CSECT
```

ITKEPARM

```
CSECT ITKEPARM DOES NOT CALL ANY CSECT
CSECT ITKEPARM IS NOT CALLED BY ANY CSECT
CSECT ITKEPARM DOES NOT JUMP TO ANY CSECT
CSECT ITKEPARM IS NOT JUMPED TO BY ANY CSECT
```

IEVEADDR

```
CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT
```

IIOEAREA

```
CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT
```

IKBEKBDT

```
CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT
```

ITKETEST

```
CSECT ITKETEST CALLS CSECT ITEEABRT
                     UNRESOLVED LABEL IWTEWAIT
                     CSECT ETKREADP
                     CSECT ETKUPADS
                     CSECT ITEEABRT
                     CSECT ITEEABRT
                     CSECT ETKREADP
                     CSECT ETKUPADS
                     CSECT ITEEABRT
                     CSECT ETKREADP
                     CSECT ETKUPADS
```

DATASET: TSS2525.CSECT.DATA

```
                     CSECT ITEEABRT
                     CSECT ITEEABRT
                     CSECT ITEEABRT
                     CSECT ETKREADP
                     CSECT ETKREADP
                     CSECT ETKUPADS
                     CSECT ITEEABRT
                     CSECT ETKREADP
                     CSECT ETKUPADS
                     CSECT ITEEABRT
                     CSECT ETKREADP
                     CSECT ETKUPADS
CSECT ITKETEST IS CALLED BY CSECT IMDEMAIN
CSECT ITKETEST DOES NOT JUMP TO ANY CSECT
CSECT ITKETEST IS NOT JUMPED TO BY ANY CSECT
```

ETKUPADS

____

```
CSECT ETKUPADS DOES NOT CALL ANY CSECT
CSECT ETKUPADS IS CALLED BY CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
CSECT ETKUPADS DOES NOT JUMP TO ANY CSECT
CSECT ETKUPADS IS NOT JUMPED TO BY ANY CSECT
```

ETKREADP

____

```
CSECT ETKREADP CALLS UNRESOLVED LABEL IWTEWAIT
                     CSECT ITEEABRT
                     UNRESOLVED LABEL IWTEWAIT
CSECT ETKREADP IS CALLED BY CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
                            CSECT ITKETEST
CSECT ETKREADP DOES NOT JUMP TO ANY CSECT
CSECT ETKREADP IS NOT JUMPED TO BY ANY CSECT
```

IUCEPARM

____

```
CSECT IUCEPARM DOES NOT CALL ANY CSECT
CSECT IUCEPARM DOES NOT CALL ANY CSECT
CSECT IUCEPARM IS NOT CALLED BY ANY CSECT
```

DATASET: TSS2525.CSECT.DATA

```
CSECT IUCEPARM DOES NOT JUMP TO ANY CSECT
CSECT IUCEPARM DOES NOT JUMP TO ANY CSECT
CSECT IUCEPARM IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IIOEAREA
________

CSECT IIOEAREA DOES NOT CALL ANY CSECT
CSECT IIOEAREA IS NOT CALLED BY ANY CSECT
CSECT IIOEAREA DOES NOT JUMP TO ANY CSECT
CSECT IIOEAREA IS NOT JUMPED TO BY ANY CSECT

IUCEUNCL
________

CSECT IUCEUNCL CALLS CSECT ITEEABRT
                     CSECT ITEEABRT
CSECT IUCEUNCL IS CALLED BY CSECT IMDEMAIN
                            CSECT IMDEMAIN
                            CSECT IMDEMAIN
                            CSECT IMDEMAIN
CSECT IUCEUNCL DOES NOT JUMP TO ANY CSECT
CSECT IUCEUNCL IS NOT JUMPED TO BY ANY CSECT

IAOEPARM
________

CSECT IAOEPARM DOES NOT CALL ANY CSECT
CSECT IAOEPARM IS NOT CALLED BY ANY CSECT
CSECT IAOEPARM DOES NOT JUMP TO ANY CSECT
CSECT IAOEPARM IS NOT JUMPED TO BY ANY CSECT

ICOEICOT
________

CSECT ICOEICOT DOES NOT CALL ANY CSECT
CSECT ICOEICOT IS NOT CALLED BY ANY CSECT
CSECT ICOEICOT DOES NOT JUMP TO ANY CSECT
CSECT ICOEICOT IS NOT JUMPED TO BY ANY CSECT

IEVEADDR
________
```

DATASET: TSS2525.CSECT.DATA

CSECT IEVEADDR DOES NOT CALL ANY CSECT
CSECT IEVEADDR IS NOT CALLED BY ANY CSECT
CSECT IEVEADDR DOES NOT JUMP TO ANY CSECT
CSECT IEVEADDR IS NOT JUMPED TO BY ANY CSECT

IKBEKBDT

CSECT IKBEKBDT DOES NOT CALL ANY CSECT
CSECT IKBEKBDT IS NOT CALLED BY ANY CSECT
CSECT IKBEKBDT DOES NOT JUMP TO ANY CSECT
CSECT IKBEKBDT IS NOT JUMPED TO BY ANY CSECT

IAOEAOFF

```
CSECT IAOEAOFF CALLS CSECT IBOEBCOT
                     CSECT IBOEBCOT
                     CSECT IBOEBCOT
                     CSECT IBOEBCOT
                     CSECT IBOEBCOT
                     CSECT IBOEBCOT
                     CSECT IBTEBLDT
                     CSECT IBTEBLDT
```