TEXAS INSTRUMENTS

Improving Man's Effectiveness Through Electronics

Model 990 Computer

Link Editor Reference Manual

MANUAL NO. 949617-9701
ORIGINAL ISSUE 15 AUGUST 1977
REVISED 15 DECEMBER 1977
INCLUDES,
CHANGE 1 . . . . . 15 MARCH 1978

Digital Systems Division

LIST OF EFFECTIVE PAGES

INSERT LATEST CHANGED PAGES DESTROY SUPERSEDED PAGES

Note: The portion of the text affected by the changes is indicated by a vertical bar in the outer margins of the page.

Model 990 Computer Link Editor Reference Manual (949617-9701)

Original Issue . . . . .15 August 1977
Revised . . . . .15 December 1977 (ECN 419810)
Change 1 . . . . .15 March 1978 (ECN 419844)

Total number of pages in this publication is 148 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover	1	3-8 - 3-44	1	C-1 - C-2	1
Effective Pages	1	4-1 - 4-13	0	Appendix D Div	0
iii	1	4-14	1	D-1 - D-2	0
iv	0	4-15 - 4-16	0	Appendix E Div	1
v-viii	1	4-17 - 4-24	1	E-1 - E-2	1
1-1 - 1-2	0	5-1 - 5-4	0	Appendix F Div	1
2-1	0	6-1 - 6-4	0	F-1 - F-2	1
2-2	1	7-1 - 7-2	0	Alphabetical Index Div	0
2-3 - 2-7	0	7-3	1	Index 1 - Index 6	0
2-8 - 2-12	1	7-4 - 7-6	0	User's Response	1
3-1	1	Appendix A Div	0	Business Reply	0
3-2	0	A-1 - A-2	0	Cover Blank	0
3-3	1	Appendix B Div	0	Cover	0
3-4 - 3-7	0	B-1 - B-6	0		
		Appendix C Div	0		

PREFACE

The Link Editor provides the user of a Texas Instruments DX10 or TX990 Operating System with the means to combine separately generated object modules to form a single linked output.

This manual describes the use of the Link Editor and consists of seven sections and six appendices, as follows:

I. Introduction — Provides an introduction to the Link Editor.

II. Structure and Function — Describes the structure of Link Edit processing and describes overlays and libraries.

III. Link Editor Commands — Provides descriptions of all the Link Editor Commands and also provides examples of command uses.

IV. Link Editor Examples — Provides examples of Link Editor runs of differing structure.

V. Link Editor Use on DX10 — Describes the operation of the Link Editor on a DX10 system.

VI. Link Editor Use on TXDS — Describes the use of the Link Editor or a TXDS system.

VII. Error Reporting — Describes the errors reported by the Link Editor.

The six appendixes are as follows:

A—Overlay Manager

B—TXDS Linking Loader

C—Command Syntax

D—TXDS Overlay Loader Routine

E—Link Editor Condition Codes Under DX10

F—Object Record Format and Tags

The following documents contain information relative to the Link Editor:

Title	Part Number
Model 990 Computer DX10 Operating System *Volumes 1 through 6*	946250-9701 to 946250-9706
Model 990 Computer TX990 Operating System Programmer's Guide (Release 2)	946259-9701
Model 990 Comuter Terminal Executive Development System (TXDS) Programmer's Guide	946258-9701

Title	Part Number
Model 990 Computer TMS 9900 Microprocessor Assembly Language Programmer's Guide	943441-9701
Model 990 Computer DX20 Operating System FORTRAN Programmer's Guide	946260-9701
Model 990 Computer DX10 Operating System COBOL Programmer's Guide	946266-9701
Model 990 Computer EX990 Operating System Programmer's Guide	949618-9701

TABLE OF CONTENTS

Paragraph Title Page

SECTION I. INTRODUCTION

SECTION II. STRUCTURE AND FUNCTION

SECTION III. LINK EDITOR COMMANDS

TABLE OF CONTENTS

Paragraph | Title | Page

SECTION IV. LINK EDITOR EXAMPLES

SECTION V. LINK EDITOR USE ON DX10

SECTION VI. LINK EDITOR USE ON TXDS

SECTION VII. ERROR REPORTING

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

1.1 INTRODUCTION

The Link Editor provides the user of the Texas Instruments TXDS or DX10 operating systems with the means to combine separately generated object modules to form a single linked output. The Link Editor accepts modules that have been generated by the assembler, the COBOL compiler, the FORTRAN compiler, or that have been produced by a previous partial link.

The major function of the Link Editor is the resolution of external definitions and references in each of the individual unlinked or partially linked object modules. The Link Editor also provides for the design and use of overlays, which allow the user to design memory efficient programs. The references to these overlays are resolved by the Link Editor.

The information in this document pertains to both TXDS and DX10 systems. However, some features of the Link Editor are not available on both systems. These differences are noted within the command descriptions, and are also detailed in Sections 5 and 6.

The following conventions are used throughout this document to define command syntax:

- Angle brackets (< >) specify a parameter that is entered by the user
- Brackets ([ ]) specify an optional entry
- Braces ({ }) indicate that a choice of the specified options must be made
- Underline (___) defines a default value
- <acnm> indicates a pathname or access name.

Some of the commands require the entry of a DX10 pathname. For disc files, the pathname is a concatenation of the volume name (default is the system disc), the directory levels (excluding the volume catalog) leading to the file, and the file name itself. Components of the pathname are separated by periods. Examples of the pathname are as follows:

VOLONE.AGENCY.RECORDS

MYDISC.MYDIRECT.MYDIRCTA.MYFILE

VOLTWO.JOE

DS01.USERA.PAYROLL

.USERA.PAYROLL Same as preceding. Default is system disc.

Device pathnames in DX10 consist of the device abbreviation concatenated with the device number. Examples are:

LP01	Line printer one
ST02	Station (VDT or hard copy) two
DS02	Disc unit two
CS01	Cassette unit one

In addition to the pathname, DX10 files can be specified by use of a synonym. Synonyms are values defined to be equal to a pathname by use of the DX10 System Command Interpreter command Assign Synonym, which is defined in the *DX10 Operating System Production Operation Reference Manual*, manual number 946250-9702. Synonyms are local to the station from which they are defined.

File pathnames in TXDS are specified in the following manner:

<dev>:<file>/<ext>

Where <dev> is the four character device name, such as DSC, DSC2, etc., <file> is the seven character name of the desired file; and <ext> is a three character file qualifier that may be used to describe the contents of a file. The defaults are defined in Section VI. Examples include:

DSC2:AGENCY/ONE

DSC:JOE

Device pathnames in TXDS are specified as in the following:

LP	Line printer
LOG	Logging device

SECTION II

STRUCTURE AND FUNCTION

2.1 LINK EDITOR FILE STRUCTURE

Figure 2-1 illustrates the relationships between the files accessed by the Link Editor and the output of the Link Editor. The Control file, which may be on disc, diskette, tape, cards, or cassette, contains the command stream that controls the linking process. In addition, the Control file may contain some or all of the object modules that are to be linked. Object modules not included in the Control file may be on disc, tape, cassette, cards, or diskette. Libraries, such as the FORTRAN runtime library, contain object modules to be included in the Link Edit. The output of the Link Editor is a linked object module that is written to a data file or a memory image that is wirtten to a program file. There are three output formats, as described in the FORMAT command in Section III.

2.2 PROCEDURES AND TASKS

Link Editor commands allow the user to link separately installed procedure and task segments. Procedures can be shared among several tasks by linking the task sections and the procedures using the Link Editor.

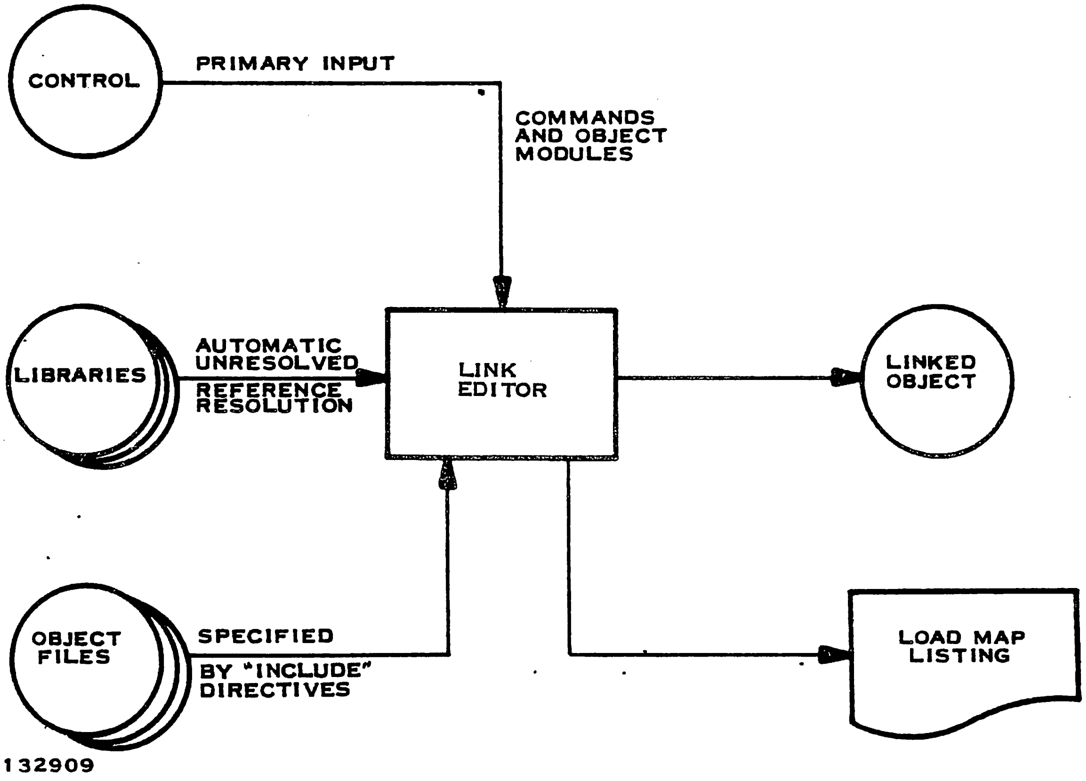

CONTROL
PRIMARY INPUT
COMMANDS AND OBJECT MODULES
LIBRARIES
AUTOMATIC UNRESOLVED REFERENCE RESOLUTION
LINK EDITOR
LINKED OBJECT
OBJECT FILES
SPECIFIED BY "INCLUDE" DIRECTIVES
LOAD MAP LISTING
(A)132909

Figure 2-1. Link Editor Files

Tasks are identified to the Link Editor by the TASK command, and separately installed procedures are identified by the PROCEDURE command. Descriptions of the commands are found in Section III, and examples of the use of these commands may be found in Section IV.

A task is defined as being either a complete program, containing both variable data and executable code, or as the variable data portion of a program. Tasks are not shareable. Procedures contain the program segment, PSEG, which may be all of a program, but is usually only the executable code and read only data. A procedure may be shared if it is reentrant, linked by use of the PROCEDURE command, and separately installed in the system. Refer to the *DX10 Operating System Application Programmer's Guide,* manual number 946250-9703 or the *Model 990 Computer TX990 Operating System Programmer's Guide* (*Release 2*), manual number 946259-9701 for complete information on the structures of tasks and procedures.

2.3 OVERLAYS

When memory space is at a premium, the user may find it advantageous to use overlays in the programs. Programs that do not use overlays are loaded in their entirety into the system and remain in memory until execution completes. Programs that use overlays conserve memory space since each overlay is resident in memory only when it is called. The total memory space required by the program is that required to hold the root portion of the program and the longest overlay path.

In the subsequent discussion of overlays, the following definitions apply:

- Phase - The smallest functional unit that can be loaded as a logical entity during execution. The linked object output contains one object module for each phase.

- Root Phase - The main or memory resident phase (level 0).

- Level - The point at which a new phase begins, identified by a level number in the overlay structure. Phases having the same level number and parent phase are loaded at the same location and are mutually exclusive (i.e., they cannot be memory resident at the same time).

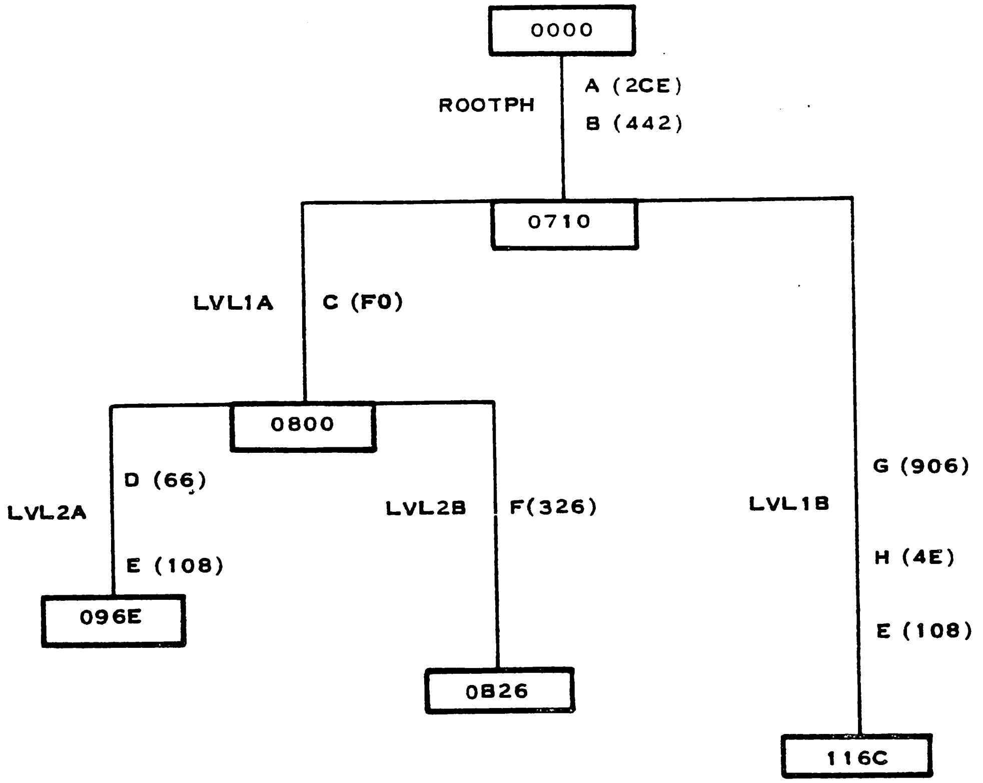
- Path - A series of phases starting with the root phase and including a successive, higher numbered phase at each level.

2.3.1 STRUCTURE CONSIDERATIONS. The structure of an overlayed program is dependent upon the relationships between the phases in the program. Phases that do not have to be in memory at the same time can overlay each other. These phases are considered to be independent in that they do not reference each other either directly or indirectly. Independent phases can be assigned the same load address and are loaded only when referenced.

When a specific phase is called, all phases in its path must be in memory, i.e., all phases between the called phase and the root phase must be in memory. A reference from one phase to another phase that is further from the root phase (in the same path) is a downward reference. A reference to a phase closer to the root phase is an upward reference. When automatic overlay loading is selected, a downward reference causes an overlay to be read into memory, if it is not currently resident, but an upward reference does not cause an overlay to be loaded. If automatic overlay loading is not used, the user must ensure that the overlay is in memory before referencing it.

Under DX10, the Load Overlay Supervisor call is used; under TXDS an overlay loader subroutine (see Appendix D) is available.

Figure 2-2 is an example of an overlayed program structure. The program consists of five phases at three levels, linked from nine object modules (eight unique modules, one used in two phases). The root phase, ROOTPH, begins at level 0. There are two phases, LVL1A and LVL1B, at level 1. These phases are mutually exclusive. Level 2 also has two phases, LVL2A and LVL2B, which are mutually exclusive. Three paths exist in this structure with the first path containing ROOTPH, LVL1A, and LVL2A; the second path contains ROOTPH, LVL1A, and LVL2B; and the third path contains ROOTPH and LVL1B.



NOTE: SYMBOLS TO THE LEFT OF THE VERTICAL LINES OF THE TREE ARE PHASE NAMES; SYMBOLS TO THE RIGHT OF THE VERTICAL LINES ARE MODULE NAMES. VALUES IN PARENTHESES ARE MODULE LENGTHS IN BYTES (HEXADECIMAL). HEXADECIMAL ADDRESSES ARE SHOWN AT THE TOP OF THE TREE AND AT THE BOTTOM OF EACH VERTICAL LINE.

(A)132910A

Figure 2-2. Overlay Structure Example

Assuming that each object module is contained in a directory with the pathname DS01.OBJECT, then the following is the control file to define the structure in figure 2-2.

```
LIBRARY  DS01.OBJECT

PHASE  0,ROOTPH

INCLUDE  (A),(B)

PHASE  1,LVL1A

INCLUDE  (C)

PHASE  2,LVL2A

INCLUDE  (D),(E)

PHASE  2,LVL2B

INCLUDE  (F)

PHASE  1,LVL1B

INCLUDE  (G),(H),(E)

END
```

Note that the order of definition is top-to-bottom and left-to-right within the structure as drawn in figure 2-2.

Figure 2-2 also serves as an example for determining the memory requirements of an overlay program. The length of each path is as follows:

ROOTPH+LVL1A+LVL2A=$096E_{16}$ bytes

ROOTPH+LVL1A+LVL2B=$0B26_{16}$ bytes

ROOTPH+LVL1B   =$116C_{16}$ bytes

Since the Link Editor determines the memory requirements by adding the requirements for the root phase and the longest overlay path, this program would require $116C_{16}$ bytes of memory. If this program did not use overlays, it would require $15E8_{16}$ bytes of memory. Therefore, use of an overlay structure saves $047C_{16}$ bytes of memory.

Careful planning of overlays is essential to achieving the maximum benefit from an overlay structure. The following considerations apply to overlay structure planning:

- Amount of memory available. The Link Editor allocates space for the longest path including phase 0.

- The frequency of use of each phase. Calling an overlay requires time, and execution time can be significantly increased if an overlay is called frequently. Code that is frequently used should be included in the main program (task or procedure), rather than in an overlay.

- The interaction between phases. A phase that transfers control to or accesses another phase must be in the same path as that phase.

- Transfers of control within a path or between paths. Flow of control does not necessarily follow a tree structure (as shown in figure 2-2).

2.3.2 AUTOMATIC OVERLAY LOADING. When the Link Editor is used to produce overlayed programs, the user can specify (by use of the LOAD command) that the Link Editor include an Automatic Overlay Manager in the linked output. The overlay manager performs automatic overlay loading during execution of the overlayed program. The LOAD command is only applicable when the IMAGE format is selected.

When the automatic overlay loading feature is used, the program must call overlays by using the Branch and Load Workspace Pointer (BLWP) instruction with the operand specifying the symbolic name of a subroutine within the overlay that is to be accessed. These references can be only one level down from the current level; i.e., you cannot go from level 1 to level 3. Note that register or indexed addressing cannot be used. For example, the call must be written:

```
BLWP   @name
```

where 'name' is the symbol associated with the transfer vector (Workspace Pointer/Program Counter) to the subroutine. Calls of the form:

```
BLWP   *5 Register Notation-INVALID
```

or:

```
BLWP   @NAME(2) Indexed Addressing-INVALID
```

are NOT VALID when using the overlay manager. The overlay manager replaces the BLWP command with a BLWP into the overlay manager and indirect addressing or indexed addressing will not work correctly.

An example of correct usage in assembly language is:

```
IDT        'SUBY'

REF        SUBZ
 .
 .
 .
BLWP       @SUBZ
 .
 .
 .
END
```

The symbol referenced (REFed), SUBZ, must be defined (DEFed) in the appropriate module as in the following:

```
IDT        'SUBX'
 .
 .
 .
DEF        SUBZ
 .
 .
 .
SUBZ DATA  WPZ,ENTRYZ
 .
 .
 .
END
```

Under DX10, the overlay manager program is included in the system subroutine library, pathname DSO1.S$SYSLIB. When the overlay manager is to be included in the linked output, the user must provide a LIBRARY command in the Link Editor control file to define this subroutine library. Under TXDS, no LIBRARY command is used. The overlay manager resides on the diskette containing the Link Editor. The structure of the overlay manager is discussed in detail in Appendix A.

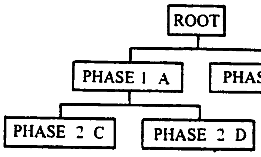
2.3.3 USER LOADED OVERLAYS. If the user does not use the automatic overlay loading capability, overlays must be loaded by issuing a Load Overlay Supervisor call (DX10 only), or by calling the overlay loader subroutine (TXDS). For a complete description of the Load Overlay supervisor call, refer to the *DX10 Operating System Application Programmer's Guide*, manual number 946250-9703. The overlay loader subroutine for TXDS is described in Appendix D.

NOTE

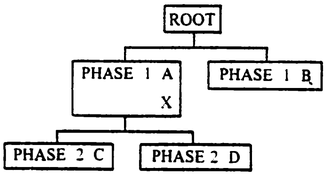
> Do not use the Automatic Overlay Manager and user loaded overlays in the same program. Unpredictable results may occur.

2.3.4 PROMOTION OF MODULES. In an overlayed program, any object modules not specifically included (by use of the INCLUDE command), but which are brought in as a result of a library search for external reference resolution (by automatic symbol resolution, SEARCH command, or the FIND command) can be promoted to a higher level (closer to the root) in the overlay structure. Such a module is promoted when it is referenced in two or more mutually exclusive phases. The module is always promoted to the lowest level phase (farthest from the root phase) that is common to the paths of all overlays that reference the module.

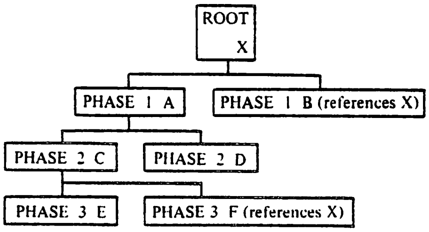
For example, assume a structure consisting of a root phase, two phases (A and B) at level one, and two phases (C and D) at level two under A. The structure is as shown in the following:



Modules C and D both reference module X. Module X is included by symbol resolution and is placed in Phase One, module A. The structure is then as follows:



The following structure shows a referenced module, X, that is promoted to the root phase. The structure consists of a root phase, two phases (A and B) at level one, two phases (C and D) at level two under A, and two phases (E and F) at level three under C. Modules B and F both reference module X, which is included by symbol resolution and promoted to the lowest common phase, which is the root phase. The structure is as follows:



2.4 IMAGE FORMAT

The IMAGE format, selected by use of the FORMAT command, allows the Link Editor to install linked output memory images directly to a specified DX10 or TXDS program file or a DX10 image file. This feature allows the user to bypass the actual installation of tasks, procedures and overlays. Linked output programs can replace existing programs or they can be added to the file. When the IMAGE format is selected and the overlays, tasks, and procedures are installed on a program file, the identifiers (IDs) of these overlays, tasks, and procedures are automatically assigned by the system. The ID assigned appears on the Load Map for the appropriate procedure, task, or phase. In order to load an overlay using a Load Overlay SVC, reference the overlay by name in the calling program, as shown:

```
REF <ovly name>

DATA <ovly name>
```

The Link Editor resolves the reference and stores the assigned overlay ID as the DATA statement operand. The ID may then be used in the supervisor call block.

NOTE

If the task name matches the overlay name, the task ID is stored in the DATA statement.

2.5 LINK EDITOR LIBRARIES

The Link Editor supports two types of library file structures, random libraries and sequential libraries. Random libraries are supported only on DX10 systems and the LIBRARY and SEARCH commands apply to random libraries only. Sequential libraries are supported on both TXDS and DX10 systems, with the FIND command applying to sequential libraries only.

2.5.1 SYMBOL RESOLUTION WITH LIBRARIES. One of the foremost advantages of libraries is the capability of the Link Editor to perform symbol resolution. Two types of resolution are available under DX10, automatic resolution at the end of the Link Edit and resolution at a user defined point in the Link Edit. TXDS only allows resolution at a user defined point in the Link Edit.

Automatic symbol resolution occurs at the end of the Link Edit (when the END command is detected). Random libraries defined by the LIBRARY command are searched in the same order they are defined to resolve any unresolved references (REFs). Any additional unresolved references created by modules included to satisfy references are also resolved automatically.

When an external reference (REF) is resolved by searching a library, the Link Editor searches for a file with the same name as that referenced by the module. For example, if a REF is to 'ABC', the Link Editor searches the defined libraries for a file named 'ABC'.

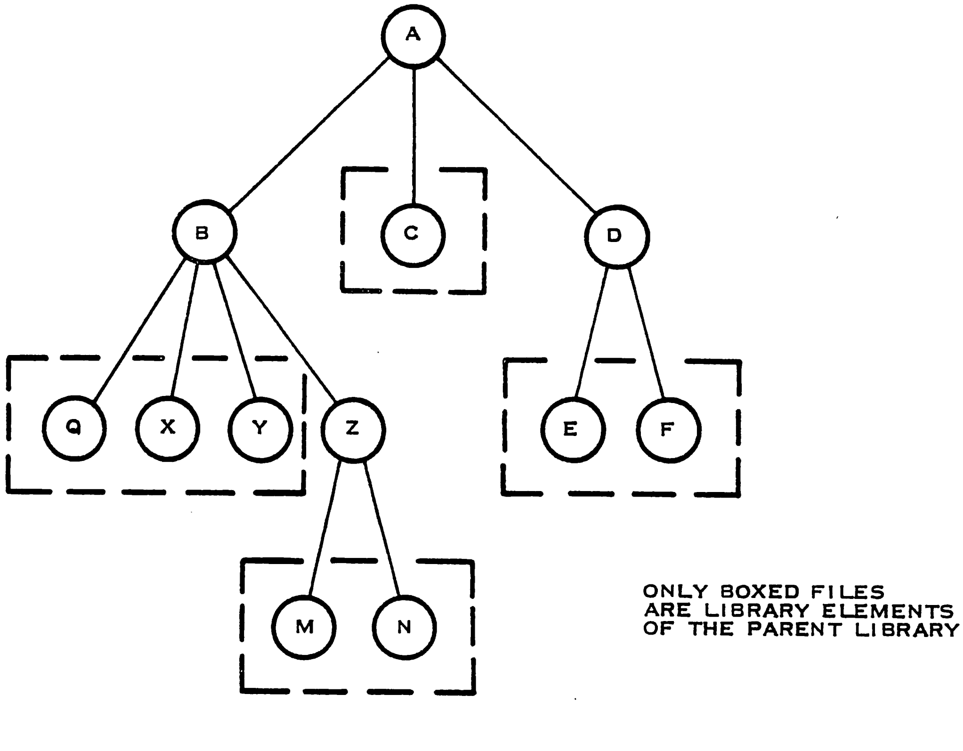


Figure 2-3. Random Library Structure

By using the SEARCH (DX10 only) or FIND (DX10 or TXDS) commands, the user can define the point within the Link Edit that symbol resolution is to occur. The SEARCH command applies to DX10 random libraries only, whereas the FIND command applies to TXDS and DX10 sequential libraries. Use of these commands allow the user to resolve previously unresolved references at a specific point within the Link Edit. The resolution occurs at the point within the link edit that the SEARCH or FIND command is processed.

2.5.2 RANDOM LIBRARIES - DX10 ONLY. A random library is a set of data files specified in a directory. Figure 2-3 is an example of the structure of a random library. A random library is conceptual in that its members are files defined in a directory.

In figure 2-3, A, B, D and Z are directories, with Q, X, Y, M, N, C, E, and F being data files. Each directory is a node, with the highest level (A in figure 2-3) being the root node. The root node is referred to in DX10 as the Volume Catalog (VCATALOG), and is assigned a symbolic name when the disc volume is installed or initialized. The VCATALOG contains pointers for each directory (node) at the next level and pointers to any files in the level immediately below the VCATALOG. In figure 2-3, pointers are contained in the VCATALOG for File C and directories B and D.

The next level of directories consists of directories B and D, with B containing the pointers to files Q, X, and Y, and directory D containing pointers to Files E and F. Directory B also contains a pointer to directory Z, and directory Z contains pointers to files M and N.

Random libraries are defined to the Link Editor by use of the LIBRARY command, with the command specifying only the pathname of a directory. For example, the following command is used to define the library (B) that contains the pointers to files Q, X, and Y:

```
LIBRARY  A.B
```

The library (Z) that contains pointers to files M and N is defined as follows:

```
LIBRARY  A.B.Z
```

The library (D) that contains pointers to files E and F is defined as follows:

```
LIBRARY  A.D.
```

The library (A) that contains the pointers to file C is defined as follows:

```
LIBRARY  A
```

2.5.3 SECONDARY ENTRY POINTS AND ALIASES - DX10 ONLY. Modules in a random library can have more than one entry point. However, these secondary entry points must be defined to the system as aliases if automatic symbol resolution is being used since they are not contained in the directory. If the module is specifically included (by use of the INCLUDE command), alias definition is not required. For example, a module name CALC has two entry points, CALC and CALC2. The module is located on disc two and is defined in directory MODS, which is defined in directory PROGS. The Link Editor statement to define CALC as a file in the library uses the library pathname: DSO2.PROGS.MODS. Note that CALC2 is not defined in the MODS directory. Therefore, the user must assign CALC2 as an alias to define it as being equal to the file name CALC. In DX10, this is accomplished by use of the Add Alias (AA) command. The Add Alias command is defined in the *DX10 Operating System Production Operations Reference Manual,* manual number 946250-9702. The format of the command is as follows:

```
AA
ADD ALIAS TO PATHNAME

   PATHNAME: <acnm>

   ALIAS PATHNAME: <acnm>
```

where:

<acnm> is the pathname to the file.

Entry points CALC and CALC2 are equated as follows:

```
AA

   PATHNAME: DSO2.PROGS.MODS.CALC

   ALIAS PATHNAME: DSO2.PROGS.MODS.CALC2
```

This command causes the pathname for CALC2 to be equated to the pathname for CALC. Any REF to CALC2 can now be resolved, since CALC2 is now in the directory.

2.5.4 SEQUENTIAL LIBRARIES. Sequential libraries are available to users of both the DX10 and TXDS operating systems. A sequential library is a sequential file that contains two or more object modules that were generated by a partial Link Edit.

On both DX10 and TXDS, the FIND command is used to access a sequential library. For example, assume that a sequential library in a DX10 system has the access name .MODS.OBJ. The FIND command would then appear in the control file as follows:

```
FIND   .MODS.OBJ
```

In TXDS, the FIND command would appear as follows for a sequential library:

```
FIND   DSC:MODS/OBJ
```

where DSC defines a diskette unit, MODS specifies the file, and OBJ specifies the file extension.

Sequential libraries contain one or more concatenated files which were output by partial link edits. The outputs of the partial link edits are concatenated into a sequential file by use of the DX10 CC (Copy Concatenate) System Command Interpreter command, the AF (Append File) command, or the TXDS Copy Concatenate Utility program.

Since the output of a partial Link Edit can consist of several modules, the user should note that when that output is contained in a sequential library and is referenced in a subsequent Link Edit, all modules are included.

Sequential libraries are searched in a different manner from random libraries. When an external reference (REF) is to be resolved using a sequential library, the Link Editor examines each external definition (DEF) in each module of the library in an attempt to find a corresponding value.

2.6 SEGMENTATION OF LINKED OUTPUT

The Link Editor provides for reorganization of the linked output into program segments, data segments, and common segments. The Model 990 Computer Assemblers (SDSMAC and TXMIRA) provide a set of directives to define these segments, with PSEG defining program segments, DSEG defining data segments, and CSEG defining common segments. If these directives are not used, the entire object module is tagged as a program segment. The FORTRAN and COBOL compilers automatically output tagged object that defines these respective segments.

SECTION III

LINK EDITOR COMMANDS

3.1 INTRODUCTION

This section contains the descriptions and syntax for the commands accepted by the Link Editor. Appendix C contains an alphabetic list of the commands and their syntax. Except where noted in the command descriptions, the commands apply to both TXDS and DX10. When using these commands, either the entire command or only the first four characters may be specified. Any comments used are separated from the command by a semicolon (;) delimiter.

3.2 MODULES AND LIBRARIES

The commands described in the following paragraphs are those that define the object modules that are to be included in the linked object output module. Also described are those commands that identify random and sequential libraries and those that specify when library searches are to be performed.

3.2.1 INCLUDE COMMAND. The INCLUDE command defines modules or files of modules that are to be included in a phase. The command is required in each phase of a task or procedure, and more than one INCLUDE command may be used, as needed. However, no INCLUDE is required after a TASK command if the procedure defined contains data segments (DSEG) or common segments (CSEG). These segments are included in the task by the Link Editor. The syntax for the command is as follows:

```
*INCLUDE  <acnm>[, <acnm>][, . . .]
```

When the Control Stream is read from a card reader and the INCLUDE command has no operand, the module or modules follow the INCLUDE command in the control file, and each module is terminated with a record having a colon (:) in the first character position, and the last module is followed by an end-of-file record (/*). When used, the <acnm> parameter (or pathname) specifies a file or device that contains one or more of the modules for the phase. If the name is enclosed in parentheses (DX10 only), the Link Editor searches all defined random libraries in the order they are defined to locate a file named as the operand of the INCLUDE command.The following examples show the use of the INCLUDE command:

INCLUDE	Includes the modules(s) that follow this command in the control file
MODULE	
.	
.	
.	
:	
/* (EOF)	
INCL CATA.APPL.OBJ.MODS	Includes the module(s) in the DX10 file MODS on the volume CATA.
INCL CS02	Includes the module(s) in the file on cassette unit two.

*At least one blank must precede the first <acnm>.

INCL (X)	Search DX10 random libraries for a file named X and includes the modules in X.
INCL DSC2:FILE/OBJ	Include a TXDS file (TXDS only).

3.2.2 LIBRARY COMMAND - DX10 ONLY. The LIBRARY command defines a random library by specifying the pathname of a directory. The specified directory is automatically searched to satisfy unresolved external references (which are treated as file names or aliases) in the module to be linked. The syntax for the command is:

```
LIBRARY <acnm>[,<acnm>] [, . . .]
```

The <acnm> parameters define the pathname to the directory that contains the desired files. These files must consist of object code produced by the assembler or higher level language compilers or previous output of the Link Editor. The following are examples of the LIBRARY command:

```
LIBRARY   CATA.APPL.OBJ

LIBR      VOLA.APPL.OBJ
```

In the above examples, CATA and VOLA are the volume IDs, APPL is the directory name, and OBJ defines the directory that the files are in. Note that different directories are accessed by the above commands even though both are named OBJ. DX10 allows files and directories to be differentiated by pathname. Since the volume names differ, the files are on different discs and are independent of one another.

3.2.3 SEARCH COMMAND - DX10 ONLY. The SEARCH command directs the Link Editor to perform a search of a random library at this point in the control stream rather than after all the control stream commands have been processed. The syntax of the command is as follows:

```
SEARCH       [<acnm>] [, . . .]
```

with the <acnm> parameters being the access names of the libraries that are to be searched. The order of these definitions determines the order of the search. If no <acnm> parameters are specified, the random libraries defined by the LIBRARY commands define the search ordering.

An example of the use of a SEARCH command is shown by the following example. The program being linked requires that a particular module, MODA, be the last module included in the linked output. Prior to the last INCLUDE command, the SEARCH command is used to resolve references in the random libraries. This sequence is necessary since automatic resolution occurs at the end of the entire Link Edit, and any modules included to resolve references would otherwise have been placed after MODA in the linked output.

```
LIBR      VOL2.MODS.EXAM1,VOL3.MOD.EXAM2

TASK      A

INCL      (MODB)
INCL      (MODC)

SEARCH

INCL      (MODA)

END
```

3.2.4 NOAUTO COMMAND - DX10 ONLY. Specification of the NOAUTO command inhibits the performing of automatic external reference resolution, even if LIBRARY commands are included in the control stream. The NOAUTO command applies to random libraries only. The NOAUTO command allows the user to explicitly control library searching through use of the SEARCH command. The syntax of the command is as follows:

```
NOAUTO

NOAU
```

3.2.5 FIND COMMAND. The FIND command is used to specify sequential libraries (see paragraph 2.5.4) that contain input modules for the Link Editor. The command can be used on both DX10 and TXDS, but it cannot specify a random library. The FIND command specifies those libraries that are to be searched for automatic external reference resolution. The syntax of the command and examples are as follows:

```
FIND <acnm>

FIND DSC2:TXLOBJ/LIB          TXDS example

FIND MYDISC.DIRA.MODS         DX10 example
```

Due to the way in which sequential libraries are searched, certain constraints apply. Only one pass is made through the library to resolve references, so if one module references a previous module in the library, that reference is not resolved unless the referenced module has already been included.

For example, if a library is structured as follows:

```
MODA    MODB    MODC
```

and MODB references MODA, that reference would only be resolved if MODA had been previously included by an INCLUDE command or by symbol resolution. References from MODB to MODC would be resolved by the FIND command.

Unresolved references can be handled by using two FIND commands with the same pathname. For example, if the library containing MODA, MODB and MODC has a TX990 pathname of DSC2:TXLOG/LIB and MODB references MODA, which has not been previously included, two FIND commands would solve the reference as follows:

```
.
.
.
.
```

```
INCL      DSC1:MODD/OBJ; REFERENCES MODB

FIND      DSC2:TXLOB/LIB; INCLUDES MODB WHICH REFERENCES MODA

FIND      DSC2:TXLOB/LIB; INCLUDES MODA

END
```

3.3 PROCEDURE, TASK, AND OVERLAY LINKING

The structure of the output of the Link Editor is controlled by the commands described in the following paragraphs. These commands define the task, procedure, and overlay sections of the program, and also define the basic structure of the output.

3.3.1 PROCEDURE COMMAND. The PROCEDURE command defines a phase of the Link Edit structure that can be installed as a procedure (a reentrant procedure may be shared among several tasks). When used, the PROCEDURE command must precede the TASK command, all PHASE commands, and the INCLUDE command that defines the Procedure module. The syntax of the command is as follows:

```
PROCEDURE  <name>
```

where <name> is the eight character, or less, identifier of the procedure that is to be used. Examples of the PROCEDURE command are shown in the following:

```
PROCEDURE                    FORLIB

PROC                         RUNLIB
```

The PROCEDURE command is used in conjunction with the INCLUDE command to define the procedure. The PROCEDURE command defines the name of the procedure, and the INCLUDE command defines the modules that are to be in that procedure. The following is an example of the use of the commands to define a DX10 Link Edit:

```
LIBRARY     CATA.APPL.OBJ          Define random library

PROCEDURE   PROC1                  Define procedure named PROC1

INCLUDE     (MOD1)                 Include MOD1 from the random
                                   library (CATA.APPL.OBJ)

INCLUDE     CATB.APPL.OBJ.MOD2     Include a file named MOD2

PROCEDURE   PROC2                  Define procedure named PROC2

INCLUDE     CSO2                   Include the module(s) in the file on
                                   cassette unit two

INCLUDE     (MOD2)                 Include MOD2 from the random library
    .
    .
    .
```

A DX10 Link Edit may contain none, one or two PROCEDURE commands. A TXDS Link Edit may contain none or one PROCEDURE command.

3.3.2 TASK COMMAND. The TASK command defines a phase of the Link Edit structure that is to be installed as a task. When used, the TASK command must follow all PROCEDURE commands and must precede all PHASE and INCLUDE commands that define the task module. The TASK command is used to link a task to a procedure, and a Link Edit may contain none or one TASK command. A TASK command and a level 0 PHASE command cannot appear in the same Link Edit, as they are logically identical. The syntax of the TASK command is as follows:

```
TASK    <name>
```

where <name> is the eight character, or less, identifier of the task module. If the <name> is omitted, the name of the first module encountered is used as the TASK name. Examples of the TASK command are as follows:

```
TASK    FORPRG

TASK    ARUN
```

The following example shows the use of the TASK command in a control file on DX10:

LIBR	CATA.APPL.OBJ	Define random library
PROC	PROC1	Define procedure name
INCLUDE	(MOD1)	Include MOD1 from library
INCLUDE	CATB.APPL.OBJ.MOD2	Include file MOD2
PROC	P2	Define procedure name
INCL	CSO2	Include the module(s) in the file on cassette unit two
INCL	(MOD2)	Include MOD2 from library
TASK	TSK1	Define task name
INCL	(MOD4)	Include MOD4 from library
INCL	CATB.APPL.OBJ.MOD3	Include file MOD3

.
.
.

The preceding control file causes a linked output to be created that has two procedures, PROC1 and P2, and the task, TSK1.

The following example shows the use of the TASK command in a control file on a TXDS system:

```
TASK        TSK2                  Define task name

INCL        DSC2.MODA/OBJ         Include MODA from disc 2

INCL        DSC2:MODB/OBJ         Include MODB from disc 2

FIND        DSC2:TXLOB/LIB        Search Library for references

END.
```

The preceding control stream causes the Link Editor to output a linked module consisting of MODA, MODB, and any modules included because of references within MODA or MODB to modules in the library DSC2:TXLOB/LIB.

3.3.3 PHASE COMMAND. The PHASE command specifies a new object module in the linked object file and specifies the level and name of the phase. PHASE commands may be followed by INCLUDE commands that define the modules included in the phase. In overlay structures, the initial phase of the task (the level 0 phase) is the resident portion and is loaded into memory with the system for a memory resident task, or loaded from the disc for a disc resident task. Phases at level 1 or higher are installed on the disc as overlays and each phase is loaded into the memory as called by a resident phase. On a DX10 system, overlays can be loaded by use of the Load Overlay supervisor call (see the *Model 990 Computer DX10 Operating System Documentation, Volume III, Applications Programming Guide*, manual number 946250-9703); or by the Automatic Overlay Manager, as described in Section II of this document. DX10 also provides an Install Overlay supervisor call and an Install Overlay System Command Interpreter command, or the Link Editor can be used to install overlays when IMAGE format is selected.

On a TX990 system, overlays are supported only on program files (see the description of the FORMAT command, IMAGE option). Overlays can be loaded by use of the overlay loader routine (see Appendix D) or by the Automatic Overlay Manager.

A task that has no overlays is processed by the Link Editor as a phase 0, therefore, a TASK and a PHASE 0 command cannot appear in the same Link Edit. Each phase is characterized by a phase name and a load point, with the load point normally being the next word past the end of the preceding phase in the same path. The syntax for the command is as follows:

```
PHASE   <level>,<name>
```

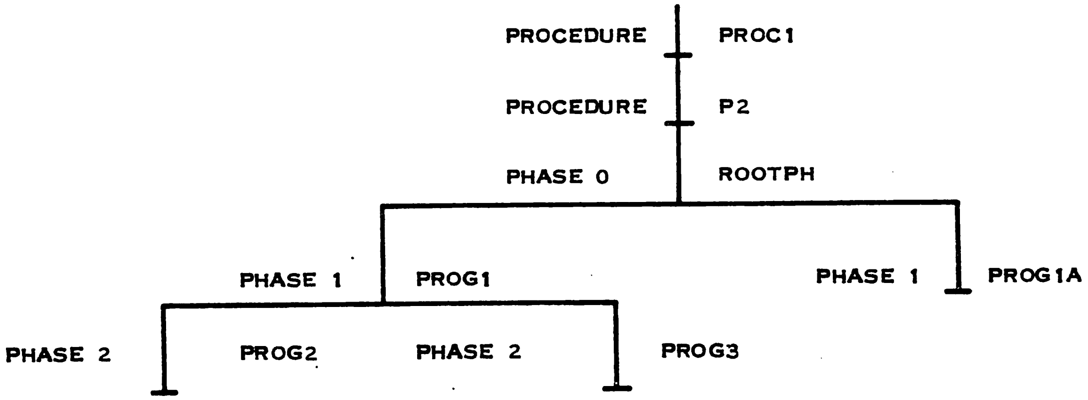
The <level> parameter identifies the level of the phase. Level numbers increase in unit steps within a path, with alternate phases that load at the same point having the same level number. The second operand, the <name> parameter, consists of from one to eight alphanumeric characters, the first of which must be alphabetic, that defines the name of the phase. The following are examples of the PHASE command:

```
PHASE       O,MAIN                Defines phase 0, named MAIN

PHAS        2,DISC                Defines phase 2, named DISC
```

The following example shows the use of PHASE commands in a DX10 Link Edit control file.

```
LIBR     CATA.APPL.OBJ          Define library
PROC     PROC1                  Define procedure name
INCL     (MOD1)                 Include MOD1 from library
INCL     CATB.APPL.OBJ.MOD2     Include file named MOD2
PROC     P2                     Define procedure name
INCL     CSO2                   Include module(s) in the file on cassette
                                unit two
INCL     (MOD2)                 Include MOD2 from library
PHASE    O,ROOTPH               Define Phase 0
INCL     (MOD4)                 Include MOD4 from library in Phase 0
INCL     CATB.APPL.OBJ.MOD3     Include file MOD3 in Phase 0
PHASE    1,PROG1                Define phase, level 1
INCL     (MOD5)                 Include MOD5 from library in Phase 2
PHASE    2,PROG2                Define phase, level 2
INCL     (MOD6)                 Include MOD6 from library in Phase 2
INCL     CATB.APPL.OBJ.MOD4     Include file MOD4 in Phase 2
PHASE    2,PROG3                Define phase, level 2
INCL     (MOD7)                 Include MOD7
PHASE    1,PROG1A               Phase 1
INCL     CATB.APPL.OBJ.MODZ     Include MODZ
END
```

The structure of the overlay program defined above is shown in figure 3-1.



(A)136924

Figure 3-1. Sample Overlay Program Structure

3.3.4 ALLOCATE COMMAND. This command is used to control the relative positioning of program, data, and common segments. The ALLOCATE command signals the Link Editor to allocate space for all outstanding data and common segments as if there were to be no more object modules included in the link. The primary purpose of the ALLOCATE command is to aid the user in sharing nonreentrant procedures between different tasks. ALLOCATE only works if all read/write data is contained in data segments, DSEGs, or common segments, CSEGs, as is the case for code generated by the FORTRAN and COBOL compilers. See the *Models 990 Computer TMS 9900 Microprocessor Assembly Language Programmer's Guide* for description of PSEG, DSEG, and CSEG.

The format of the ALLOCATE command is as follows:

```
ALLOCATE
    or
ALLO
```

ALLOCATE can only be used in the task portion of a link edit, after a TASK or PHASE 0 command and before a PHASE 1 command. ALLOCATE can not be used in partial links.

The following rules will help the user to obtain the desired results from ALLOCATE. The Link Editor cannot enforce these rules.

1. If ALLOCATE and LOAD are both used in the same link edit, ALLOCATE must occur before the LOAD command.

2. Care must be used when using COMMON or CSEGs, with ALLOCATE. The COMMON or CSEG reference before the ALLOCATE must not have elements added to it by modules after the ALLOCATE if the procedure is to be shared.

3. The procedure being shared must not reference anything occurring after the ALLOCATE command.

The following methods may be used to detect rule violations:

1. Find the section of the link map with "POST ALLOCATE" in place of a task ID.

2. Be sure that all link maps of tasks which are sharing a given procedure must have identical link maps up to "POST ALLOCATE".

Example: suppose task T1 and T2 are to share a procedure PX. The following two link edit control streams incorrectly link the two tasks to the shared procedure.

```
PROC PX                          PROC PX
INCL .X.Y.PX                     DUMMY
TASK TI                          INCL .X.Y.PX
INCL .X.Y.T1                     TASK T2
END                              INCL .X.Y.T2
                                 END
```

Assuming that the module in .X.Y.PX is coded using PSEGs, DSEGs, and CSEGs, and that the size of T1 is different from the size of T2, then task T2 will not execute correctly. When procedure PX was created by the link with T1, it was given addresses for the DSEGs and CSEGs it used. Those addresses were not the same in T2.

The following control streams correctly link T1 and T2 to PX.

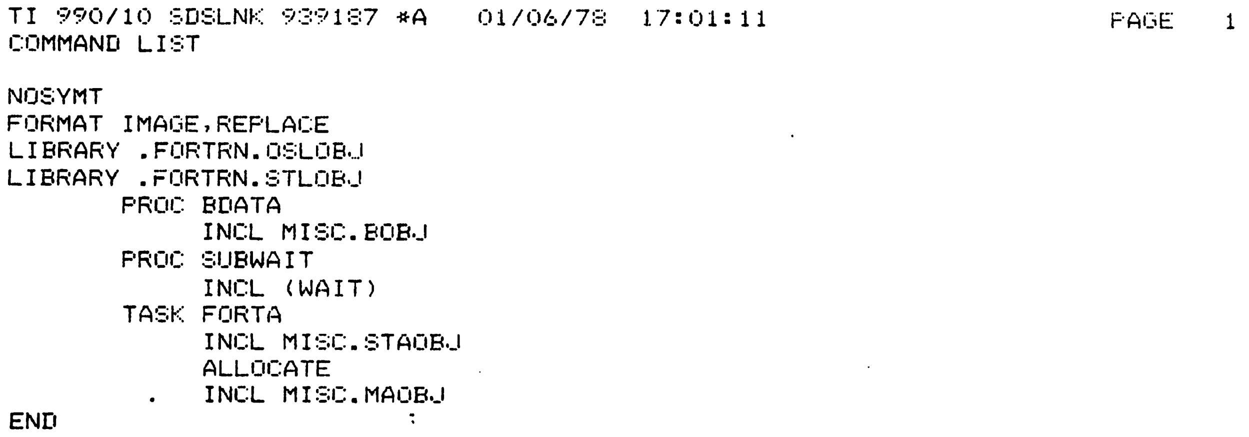
```
PROC PX                          PROC PX
INCL .X.Y.PX                     DUMMY
TASK T1                          INCL .X.Y.PX
INCL .X.Y.TC                     TASK T2
ALLOCATE                         INCL .X.Y.TC
INCL .X.Y.T1                     ALLO
END                              INCL .X.Y.T2
                                 END
```

The solution is a fixed length module, .X.Y.TC, and ALLOCATE. ALLOCATE forces all DSEGs and CSEGs from the procedure to be allocated storage now. If a fixed length part of the task is INCLUDED before the ALLOCATE command the procedure can be shared.

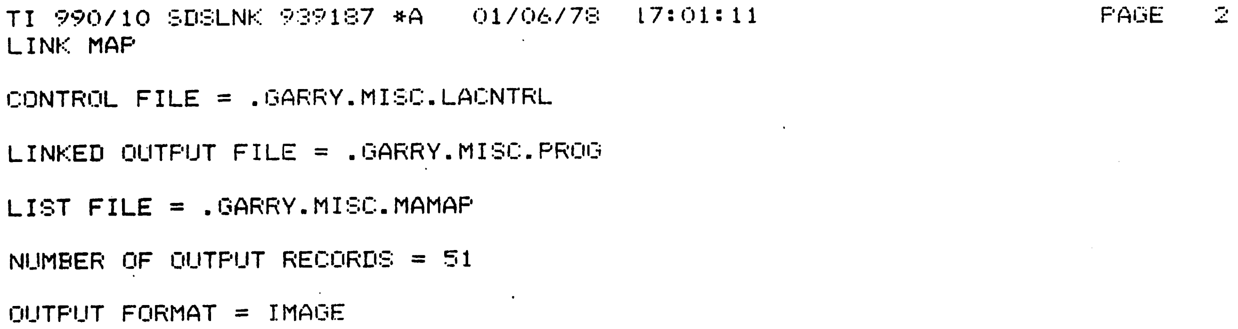
In COBOL the nonreentrant section of the runtime package, RCBTSK, may be used as a fixed length part of the task. In assembly language the three word vector (WP, PC, End Action) may be used as a fixed part of the task. In FORTRAN, a program which calls the main program as a subroutine (as shown below) will work.

```
CALL MAIN
STOP
END
```

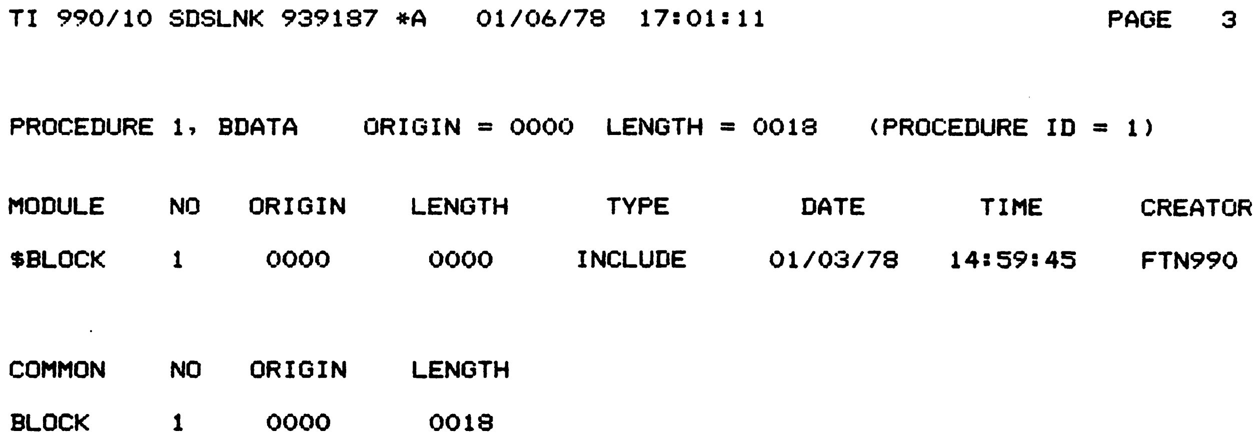
The following link maps show two FORTRAN programs which communicate through a shared procedure 1, $BLOCK. The programs also share a common subroutine, WAIT, in procedure 2.



```
TI 990/10 SDSLNK 939187 *A    01/06/78  17:01:11                     PAGE    1
COMMAND LIST

NOSYMT
FORMAT IMAGE,REPLACE
LIBRARY .FORTRN.OSLOBJ
LIBRARY .FORTRN.STLOBJ
       PROC BDATA
            INCL MISC.BOBJ
       PROC SUBWAIT
            INCL (WAIT)
       TASK FORTA
            INCL MISC.STAOBJ
            ALLOCATE
            INCL MISC.MAOBJ
END
```

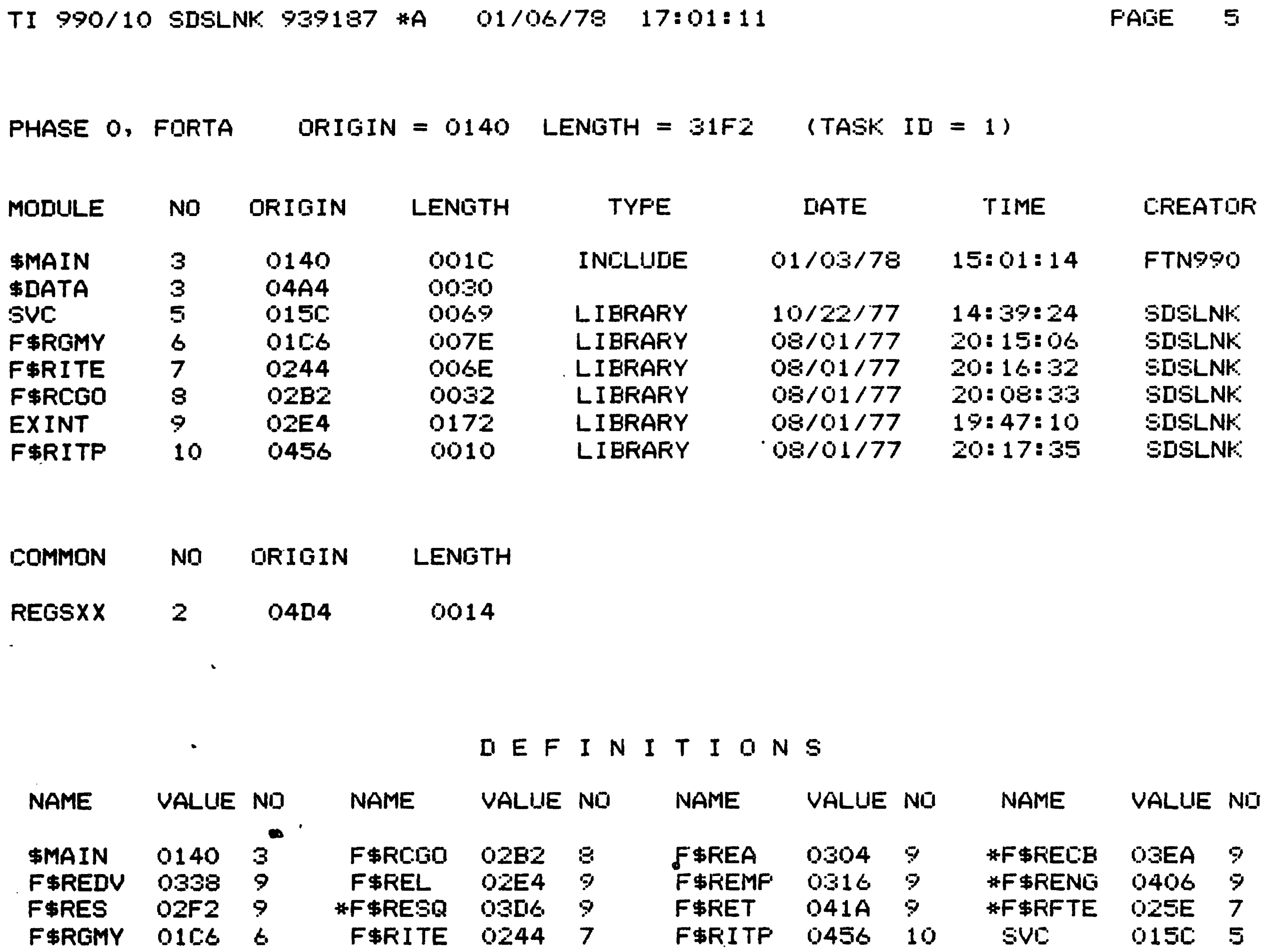


```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:01:11                    PAGE   2
LINK MAP

CONTROL FILE = .GARRY.MISC.LACNTRL

LINKED OUTPUT FILE = .GARRY.MISC.PROG

LIST FILE = .GARRY.MISC.MAMAP

NUMBER OF OUTPUT RECORDS = 51

OUTPUT FORMAT = IMAGE
```



```
TI 990/10 SDSLNK 939187 *A    01/06/78  17:01:11                          PAGE   3


PROCEDURE 1, BDATA    ORIGIN = 0000  LENGTH = 0018   (PROCEDURE ID = 1)

MODULE     NO   ORIGIN     LENGTH       TYPE         DATE       TIME       CREATOR

$BLOCK     1     0000       0000      INCLUDE     01/03/78   14:59:45     FTN990


COMMON     NO   ORIGIN     LENGTH

BLOCK      1     0000       0018
```

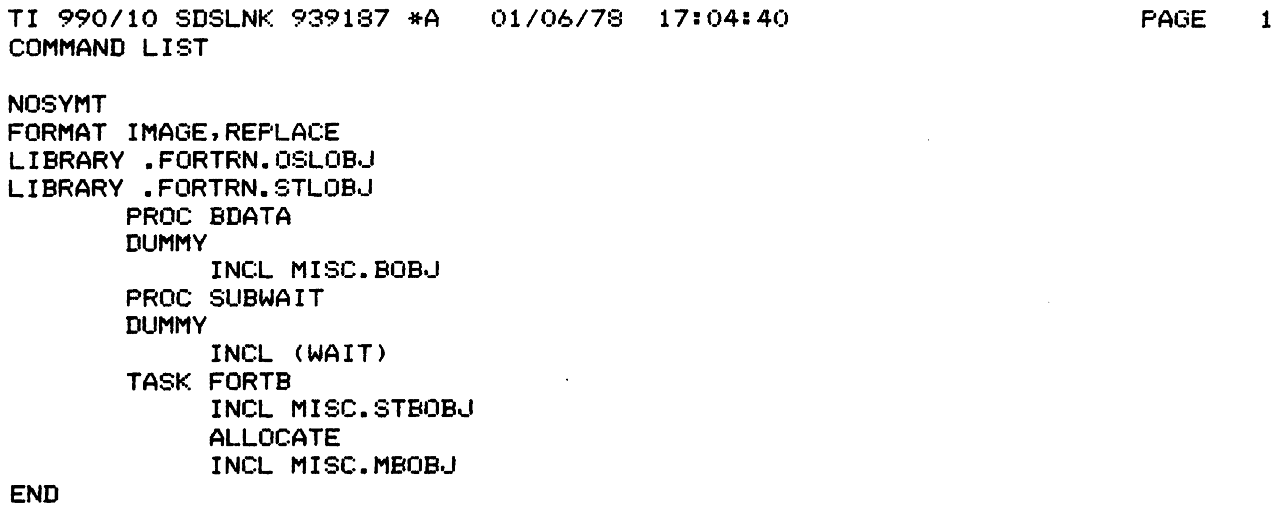
```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:01:11                        PAGE   4


PROCEDURE 2, SUBWAIT  ORIGIN = 0020  LENGTH = 0114   (PROCEDURE ID = 2)


MODULE      NO    ORIGIN      LENGTH       TYPE          DATE         TIME        CREATOR

WAIT        2      0020        0114      INCLUDE       08/01/77    22:32:05       SDSLNK
$DATA       2      0466        003E



                                  D E F I N I T I O N S

 NAME       VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 WAIT       0020   2
```



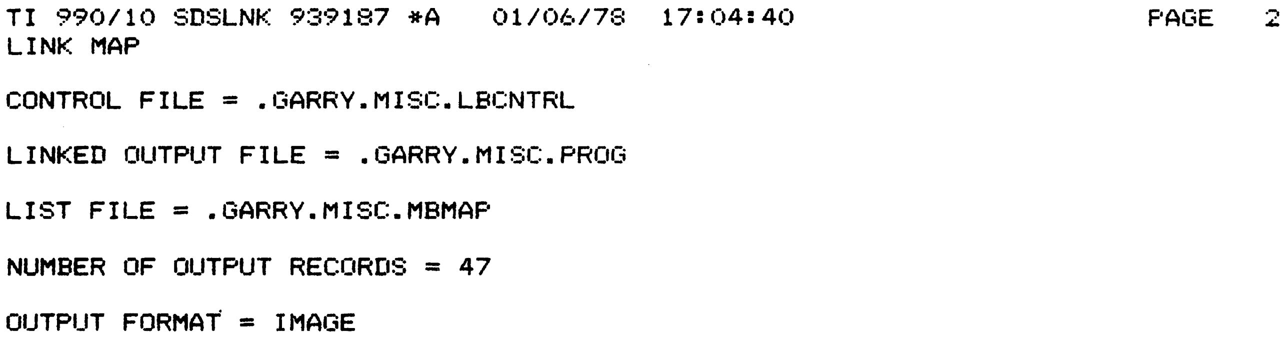
```
TI 990/10 SDSLNK 939187 *A    01/06/78   17:01:11                          PAGE    5


PHASE 0, FORTA    ORIGIN = 0140  LENGTH = 31F2   (TASK ID = 1)

MODULE     NO    ORIGIN    LENGTH      TYPE        DATE       TIME       CREATOR

$MAIN      3      0140      001C     INCLUDE     01/03/78   15:01:14    FTN990
$DATA      3      04A4      0030
SVC        5      015C      0069     LIBRARY     10/22/77   14:39:24    SDSLNK
F$RGMY     6      01C6      007E     LIBRARY     08/01/77   20:15:06    SDSLNK
F$RITE     7      0244      006E     LIBRARY     08/01/77   20:16:32    SDSLNK
F$RCGO     8      02B2      0032     LIBRARY     08/01/77   20:08:33    SDSLNK
EXINT      9      02E4      0172     LIBRARY     08/01/77   19:47:10    SDSLNK
F$RITP     10     0456      0010     LIBRARY     08/01/77   20:17:35    SDSLNK


COMMON     NO    ORIGIN    LENGTH

REGSXX     2      04D4      0014


                                 D E F I N I T I O N S

 NAME      VALUE  NO     NAME     VALUE  NO     NAME     VALUE  NO     NAME     VALUE  NO

 $MAIN     0140   3      F$RCGO   02B2   8      F$REA    0304   9     *F$RECB   03EA   9
 F$REDV    0338   9      F$REL    02E4   9      F$REMP   0316   9     *F$RENG   0406   9
 F$RES     02F2   9     *F$RESQ   03D6   9      F$RET    041A   9     *F$RFTE   025E   7
 F$RGMY    01C6   6      F$RITE   0244   7      F$RITP   0456   10     SVC      015C   5
```

```
TI 990/10 SDSLNK 939187 *A    01/06/78   17:01:11                          PAGE    6


PHASE 0, FORTA     ORIGIN = 04E8   LENGTH = 2E4A    (POST ALLOCATE)

MODULE      NO    ORIGIN     LENGTH       TYPE         DATE        TIME       CREATOR

MAINA       4      04E8       00AA      INCLUDE      01/03/78    15:10:54     FTN990
$DATA       4      0592       0034
F$RPAU      11     05C6       00B0      LIBRARY      08/01/77    21:59:27     SDSLNK
F$XPRE      12     0676       004E      LIBRARY      10/17/77    17:19:14     SDSLNK
F$RFZ       13     06C4       0354      LIBRARY      08/01/77    21:54:59     SDSLNK
F$RINP      14     0A18       0074      LIBRARY      08/01/77    21:56:53     SDSLNK
F$ERRC      15     0A8C       013C      LIBRARY      08/01/77    21:19:27     SDSLNK
F$RWRK      16     0BC8       0000      LIBRARY      09/26/77    19:23:13     SDSLNK
$DATA       16     0BC8       0170
F$RPRE      17     0D38       008E      LIBRARY      08/01/77    22:00:16     SDSLNK
F$XCLS      18     0DC6       0086      LIBRARY      08/01/77    22:05:29     SDSLNK
F$RBUF      19     0E4C       0000      LIBRARY      10/28/77    16:40:05     SDSLNK
$DATA       19     0E4C       008E
F$XFCB      20     0EDA       0000      LIBRARY      09/28/77    17:51:24     SDSLNK
$DATA       20     0EDA       00C8
F$XVFB      21     0FA2       02C0      LIBRARY      10/13/77    16:23:08     SDSLNK
F$XTBL      22     1262       0000      LIBRARY      09/26/77    19:24:11     SDSLNK
$DATA       22     1262       038D
F$XLWS      23     15F0       0000      LIBRARY      09/26/77    19:23:53     SDSLNK
$DATA       23     15F0       002C
F$XLOG      24     161C       008F      LIBRARY      08/01/77    22:11:02     SDSLNK
F$RFTS      25     16AC       006A      LIBRARY      08/01/77    21:50:55     SDSLNK
F$FINP      26     1716       0B54      LIBRARY      09/28/77    11:33:54     SDSLNK
F$XERR      27     226A       000A      LIBRARY      08/01/77    22:07:38     SDSLNK
F$XFTL      28     2274       0012      LIBRARY      08/01/77    22:08:42     SDSLNK
F$XBUT      29     2286       002D      LIBRARY      08/01/77    22:04:46     SDSLNK
F$RMSG      30     22B4       0100      LIBRARY      10/03/77    17:31:43     SDSLNK
F$XLIO      31     23B4       001A      LIBRARY      08/01/77    22:10:21     SDSLNK
F$XIOF      32     23CE       013C      LIBRARY      10/26/77    13:12:07     SDSLNK
F$REVP      33     250A       000C      LIBRARY      10/26/77    13:14:35     SDSLNK
F$RWSP      34     2516       0000      LIBRARY      10/22/77    15:20:19     SDSLNK
$DATA       34     2516       00FC
F$RAER      35     2612       0041      LIBRARY      08/01/77    20:07:25     SDSLNK
F$FITP      36     2654       01B8      LIBRARY      08/01/77    19:57:39     SDSLNK
F$XRST      37     280C       0002      LIBRARY      08/01/77    19:53:42     SDSLNK
S$GTCA      38     280E       0634      LIBRARY      08/15/77    14:33:44     SDSLNK
$DATA       38     2E42       04F0


                               D E F I N I T I O N S

  NAME     VALUE  NO      NAME     VALUE  NO      NAME     VALUE  NO      NAME     VALUE  NO

  A$BBUF   0106*  16      A$BFCB   00FA*  16      A$BPRB   0102*  16      A$BTCA   00FE*  16
  A$BWK1   0BC8   16      A$BWK2   2516   34      A$EFCB   00FC*  16      A$EPRB   0104*  16
  A$ETCA   0100*  16      F$ALN1   012C*  16      F$ASAD   0006*  17      F$ASLU   016C*  16
  F$ERRC   0A8C   15      F$ERRS   0A92   15     *F$ERST   0B16   15     *F$FACC   172E   26
 *F$FACD   1732   26      F$FCBE   000A*  17      F$FCOL   19AE   26      F$FCUS   1942   26
  F$FDEN   1748   26     *F$FDIS   1716   26     *F$FDIT   171A   26      F$FDOL   19A0   26
  F$FDUS   1934   26      F$FENN   173E   26      F$FEOL   1972   26      F$FEUS   1906   26
  F$FFOL   1980   26      F$FFUS   1914   26      F$FIOL   1964   26      F$FITP   2654   36
```

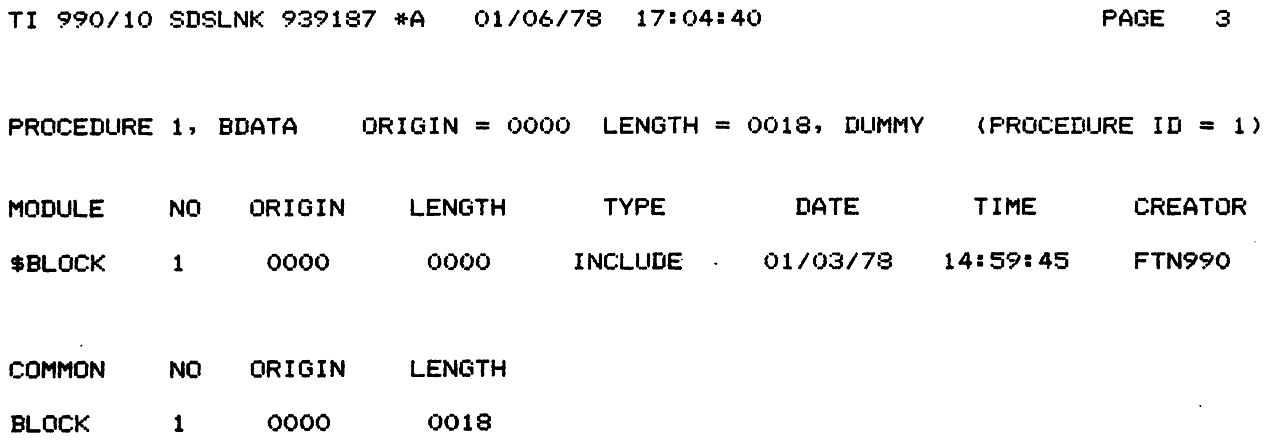
```
TI 990/10 SDSLNK 939187 *A    01/06/78   17:01:11                     PAGE   7
 NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 F$FIUS    18F8   26    F$FLAG    0005*  17    F$FLOL    19C2   26    F$FLUS    1956   26
 F$FRE     1778   26    F$FREB    1772   26    F$FRED    176C   26    F$FRER    1766   26
 F$FRF     1790   26    F$FRFB    178A   26    F$FRFD    1784   26    F$FRFR    177E   26
 F$FROL    1992   26    F$FRUS    1926   26    F$FSIO    1A30   26    F$FWE     1754   26
 F$FWER    174E   26    F$FWF     1760   26    F$FWFR    175A   26    F$ILOG    0CF2   16
 F$INAS    0166*  16    F$LSTA    0001*  17    F$LUNO    0000*  17    F$LUNT    0164*  16
 F$NAME    0008*  17   *F$OPEN    11B0   21    F$PRB     0002*  17    F$R10A    0014*  16
 F$R10B    011C*  16    F$RAER    2612   35   *F$RASN    0376*  22    F$RBUF    0E50   19
*F$RCAL    0375*  22   *F$RCOL    0A84   14   *F$RCUS    0A68   14   *F$RDEN    0A1C   14
*F$RDOL    0A7C   14   *F$RDUS    0A60   14   *F$RENN    0A18   14   *F$REOL    0A70   14
*F$REUS    0A54   14    F$REVP    250A   33    F$RFFD    208E   26    F$RFFQ    16DA   25
 F$RFI     06C4   13    F$RFL     07E8   13   *F$RFOL    0A74   14    F$RFRW    2016   26
 F$RFSI    1A40   26    F$RFSR    2128   26    F$RFSW    20C4   26    F$RFTS    16AC   25
*F$RFUS    0A58   14    F$RFWB    1702   25    F$RFWD    2006   26    F$RFZ     084E   13
 F$RIOL    0A6C   14   *F$RIUS    0A50   14    F$RLOG    0128*  16   *F$RLOL    0A80   14
 F$RLP2    012A*  16   *F$RLUS    0A64   14   *F$RMON    038C*  22   *F$RNUM    036F*  22
*F$ROP     0362*  22   *F$RPAU    05C6   11   *F$RPRB    0360*  22    F$RPRE    0D38   17
*F$RPRM    0370*  22   *F$RRE     0A3C   14   *F$RREB    0A38   14   *F$RRED    0A34   14
*F$RRER    0A30   14   *F$RRF     0A4C   14   *F$RRFB    0A48   14   *F$RRFD    0A44   14
*F$RRFR    0A40   14   *F$RROL    0A78   14   *F$RRUS    0A5C   14    F$RSIO    0A88   14
 F$RSTO    05D8   11   *F$RTCA    0366*  22    F$RTFG    0BF0   16   *F$RTID    0373*  22
 F$RVFB    0028*  16    F$RVP2    002A*  16   *F$RWE     0A24   14   *F$RWER    0A20   14
 F$RWF     0A2C   14   *F$RWFR    0A28   14    F$RWRK    0BCA   16    F$RWSP    2518   34
 F$STAT    0004*  17    F$UNIT    015F*  16    F$XBCH    0020*  23    F$XBFR    0027*  23
 F$XBFS    008A*  19    F$XBFX    0028*  23    F$XBFY    002A*  23   *F$XBUI    24B4   32
*F$XBUO    24A0   32    F$XBUT    2286   29   *F$XCAL    15D7   22    F$XCLS    0DC6   18
 F$XCPX    0022*  23    F$XCPY    0024*  23   *F$XEOF    2496   32    F$XERR    226A   27
 F$XFCB    0EDA   20    F$XFCE    0FA2   20   *F$XFND    249C   32    F$XFTL    2274   28
 F$XLIO    23B4   31    F$XLOG    161C   24    F$XLWS    15F0   23   *F$XMON    15EE   22
 F$XPRE    0676   12    F$XPSE    0DB4   17    F$XRED    23CE   32    F$XRST    280C   37
*F$XRWD    247E   32    F$XSTP    0DBC   17    F$XTBE    15C2   22    F$XTBL    1262   22
*F$XTID    15D5   22   *F$XTRA    2476   32    F$XVFB    0FAE   21    F$XVWS    0CD2   16
 F$XWRT    23EC   32    F$XWS0    253E   34    F$XWS1    2540   34    F$XWS2    2542   34
 F$XWS3    2544   34    G$XE01    22B4   30    G$XE08    22C7   30    G$XE09    22E9   30
 G$XE10    2316   30   *G$XE11    2331   30    G$XE12    234C   30    G$XE13    2364   30
 G$XE14    23A2   30    MAINA     04E8   4     P$ABUF    0006*  17   *P$AKEY    000C*  17
 P$CCNT    000A*  17    P$ERR     0001*  17   *P$IFA     001C*  17    P$LACN    0016*  17
 P$LUN     0003*  17    P$OP      0002*  17   *P$PASS    0018*  17    P$PFCB    0024*  17
*P$PRB     0000*  17    P$PRBE    0026*  17   *P$REC1    000D*  17   *P$REC2    000E*  17
 P$RECL    0008*  17   *P$RES1    000C*  17   *P$RES2    001A*  17   *P$SFA     0020*  17
*P$SFLG    0004*  17   *P$SVC0    0000*  17    P$UFLG    0005*  17    P$ULRL    0012*  17
 P$UPRL    0014*  17    P$UTF1    0010*  17   *P$UTF2    0011*  17   *PX$CUR    0012*  17
*PX$EVC    0011*  17   *PX$FIL    0010*  17   *PX$FLG    000E*  17   *PX$FST    0014*  17
*S$APRB    1174   21    S$CLOS    2B96   38    S$GTCA    280E   38    S$MAPS    28E0   38
 S$OPEN    2BF0   38    S$RTCA    2894   38    S$STOP    2DF8   38    S$WEOL    2C48   38
 S$WRIT    2CAE   38



****  LINKING COMPLETED
```



```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:04:40                     PAGE    1
COMMAND LIST

NOSYMT
FORMAT IMAGE,REPLACE
LIBRARY .FORTRN.OSLOBJ
LIBRARY .FORTRN.STLOBJ
       PROC BDATA
       DUMMY
            INCL MISC.BOBJ
       PROC SUBWAIT
       DUMMY
            INCL (WAIT)
       TASK FORTB
            INCL MISC.STBOBJ
            ALLOCATE
            INCL MISC.MBOBJ
END
```



```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:04:40                     PAGE   2
LINK MAP

CONTROL FILE = .GARRY.MISC.LBCNTRL

LINKED OUTPUT FILE = .GARRY.MISC.PROG

LIST FILE = .GARRY.MISC.MBMAP

NUMBER OF OUTPUT RECORDS = 47

OUTPUT FORMAT = IMAGE
```



```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:04:40                          PAGE   3


PROCEDURE 1, BDATA    ORIGIN = 0000  LENGTH = 0018, DUMMY   (PROCEDURE ID = 1)

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE       TIME       CREATOR

$BLOCK     1     0000      0000     INCLUDE    01/03/78   14:59:45    FTN990


COMMON     NO   ORIGIN    LENGTH

BLOCK      1     0000      0018
```

```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:04:40                      PAGE   4


PROCEDURE 2, SUBWAIT  ORIGIN = 0020  LENGTH = 0114, DUMMY   (PROCEDURE ID = 2)

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME       CREATOR

WAIT       2     0020      0114     INCLUDE     08/01/77    22:32:05    SDSLNK
$DATA      2     0466      003E


                                D E F I N I T I O N S
 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

 WAIT     0020  2
```

```
TI 990/10 SDSLNK 939187 *A    01/06/78  17:04:40                         PAGE   5


PHASE 0, FORTB    ORIGIN = 0140  LENGTH = 31B8   (TASK ID = 2)


MODULE      NO    ORIGIN     LENGTH       TYPE         DATE        TIME       CREATOR

$MAIN       3      0140       001C      INCLUDE     01/03/78    15:02:18     FTN990
$DATA       3      04A4       0030
SVC         5      015C       0069      LIBRARY     10/22/77    14:39:24     SDSLNK
F$RGMY      6      01C6       007E      LIBRARY     08/01/77    20:15:06     SDSLNK
F$RITE      7      0244       006E      LIBRARY     08/01/77    20:16:32     SDSLNK
F$RCGO      8      02B2       0032      LIBRARY     08/01/77    20:08:33     SDSLNK
EXINT       9      02E4       0172      LIBRARY     08/01/77    19:47:10     SDSLNK
F$RITP      10     0456       0010      LIBRARY     08/01/77    20:17:35     SDSLNK


COMMON      NO    ORIGIN     LENGTH

REGSXX      2      04D4       0014



                                   D E F I N I T I O N S

 NAME     VALUE  NO     NAME     VALUE  NO     NAME     VALUE  NO     NAME     VALUE  NO

 $MAIN    0140   3      F$RCGO   02B2   8      F$REA    0304   9     *F$RECB   03EA   9
 F$REDV   0338   9      F$REL    02E4   9      F$REMP   0316   9     *F$RENG   0406   9
 F$RES    02F2   9     *F$RESQ   03D6   9      F$RET    041A   9     *F$RFTE   025E   7
 F$RGMY   01C6   6      F$RITE   0244   7      F$RITP   0456   10     SVC      015C   5
```

```
TI 990/10 SDSLNK 939187 *A   01/06/78  17:04:40                          PAGE   6


PHASE 0, FORTB    ORIGIN = 04E8  LENGTH = 2E10   (POST ALLOCATE)

MODULE      NO    ORIGIN     LENGTH      TYPE         DATE        TIME       CREATOR

MAINB       4      04E8       0070     INCLUDE      01/03/78    15:16:13     FTN990
$DATA       4      0558       0034
F$RPAU      11     058C       00B0     LIBRARY      08/01/77    21:59:27     SDSLNK
F$XPRE      12     063C       004E     LIBRARY      10/17/77    17:19:14     SDSLNK
F$RFZ       13     068A       0354     LIBRARY      08/01/77    21:54:59     SDSLNK
F$RINP      14     09DE       0074     LIBRARY      08/01/77    21:56:53     SDSLNK
F$ERRC      15     0A52       013C     LIBRARY      08/01/77    21:19:27     SDSLNK
F$RWRK      16     0B8E       0000     LIBRARY      09/26/77    19:23:13     SDSLNK
$DATA       16     0B8E       0170
F$RPRE      17     0CFE       008E     LIBRARY      08/01/77    22:00:16     SDSLNK
F$XCLS      18     0D8C       0086     LIBRARY      08/01/77    22:05:29     SDSLNK
F$RBUF      19     0E12       0000     LIBRARY      10/28/77    16:40:05     SDSLNK
$DATA       19     0E12       008E
F$XFCB      20     0EA0       0000     LIBRARY      09/28/77    17:51:24     SDSLNK
$DATA       20     0EA0       00C8
F$XVFB      21     0F68       02C0     LIBRARY      10/13/77    16:23:08     SDSLNK
F$XTBL      22     1228       0000     LIBRARY      09/26/77    19:24:11     SDSLNK
$DATA       22     1228       038D
F$XLWS      23     15B6       0000     LIBRARY      09/26/77    19:23:53     SDSLNK
$DATA       23     15B6       002C
F$XLOG      24     15E2       008F     LIBRARY      08/01/77    22:11:02     SDSLNK
F$RFTS      25     1672       006A     LIBRARY      08/01/77    21:50:55     SDSLNK
F$FINP      26     16DC       0B54     LIBRARY      09/28/77    11:33:54     SDSLNK
F$XERR      27     2230       000A     LIBRARY      08/01/77    22:07:38     SDSLNK
F$XFTL      28     223A       0012     LIBRARY      08/01/77    22:08:42     SDSLNK
F$XBUT      29     224C       002D     LIBRARY      08/01/77    22:04:46     SDSLNK
F$RMSG      30     227A       0100     LIBRARY      10/03/77    17:31:43     SDSLNK
F$XLIO      31     237A       001A     LIBRARY      08/01/77    22:10:21     SDSLNK
F$XIOF      32     2394       013C     LIBRARY      10/26/77    13:12:07     SDSLNK
F$REVP      33     24D0       000C     LIBRARY      10/26/77    13:14:35     SDSLNK
F$RWSP      34     24DC       0000     LIBRARY      10/22/77    15:20:19     SDSLNK
$DATA       34     24DC       00FC
F$RAER      35     25D8       0041     LIBRARY      08/01/77    20:07:25     SDSLNK
F$FITP      36     261A       01B8     LIBRARY      08/01/77    19:57:39     SDSLNK
F$XRST      37     27D2       0002     LIBRARY      08/01/77    19:53:42     SDSLNK
S$GTCA      38     27D4       0634     LIBRARY      08/15/77    14:33:44     SDSLNK
$DATA       38     2E08       04F0



                                  D E F I N I T I O N S

  NAME      VALUE  NO      NAME      VALUE  NO      NAME      VALUE  NO      NAME      VALUE  NO

  A$BBUF    0106*  16      A$BFCB    00FA*  16      A$BPRB    0102*  16      A$BTCA    00FE*  16
  A$BWK1    0B8E   16      A$BWK2    24DC   34      A$EFCB    00FC*  16      A$EPRB    0104*  16
  A$ETCA    0100*  16      F$ALN1    012C*  16      F$ASAD    0006*  17      F$ASLU    016C*  16
  F$ERRC    0A52   15      F$ERRS    0A58   15     *F$ERST    0ADC   15     *F$FACC    16F4   26
 *F$FACD    16F8   26      F$FCBE    000A*  17      F$FCOL    1974   26      F$FCUS    1908   26
  F$FDEN    170E   26     *F$FDIS    16DC   26     *F$FDIT    16E0   26      F$FDOL    1966   26
  F$FDUS    18FA   26      F$FENN    1704   26      F$FEOL    1938   26      F$FEUS    18CC   26
  F$FFOL    1946   26      F$FFUS    18DA   26      F$FIOL    192A   26      F$FITP    261A   36
```

```
TI 990/10 SDSLNK 939187 *A    01/06/78   17:04:40                             PAGE   7
  NAME      VALUE  NO     NAME      VALUE  NO     NAME      VALUE  NO     NAME      VALUE  NO

  F$FIUS    18BE   26     F$FLAG    0005*  17     F$FLOL    1988   26     F$FLUS    191C   26
  F$FRE     173E   26     F$FREB    1738   26     F$FRED    1732   26     F$FRER    172C   26
  F$FRF     1756   26     F$FRFB    1750   26     F$FRFD    174A   26     F$FRFR    1744   26
  F$FROL    1958   26     F$FRUS    18EC   26     F$FSIO    19F6   26     F$FWE     171A   26
  F$FWER    1714   26     F$FWF     1726   26     F$FWFR    1720   26     F$ILOG    0CB8   16
  F$INAS    0166*  16     F$LSTA    0001*  17     F$LUNO    0000*  17     F$LUNT    0164*  16
  F$NAME    0008*  17    *F$OPEN    1176   21     F$PRB     0002*  17     F$R10A    0014*  16
  F$R10B    011C*  16     F$RAER    25D8   35    *F$RASN    0376*  22     F$RBUF    0E16   19
 *F$RCAL    0375*  22    *F$RCOL    0A4A   14    *F$RCUS    0A2E   14    *F$RDEN    09E2   14
 *F$RDOL    0A42   14    *F$RDUS    0A26   14    *F$RENN    09DE   14    *F$REOL    0A36   14
 *F$REUS    0A1A   14     F$REVP    24D0   33     F$RFFD    2054   26     F$RFFQ    16A0   25
  F$RFI     068A   13     F$RFL     07AE   13    *F$RFOL    0A3A   14     F$RFRW    1FDC   26
  F$RFSI    1A06   26     F$RFSR    20EE   26     F$RFSW    208A   26     F$RFTS    1672   25
 *F$RFUS    0A1E   14     F$RFWB    16C8   25     F$RFWD    1FCC   26     F$RFZ     0814   13
  F$RIOL    0A32   14    *F$RIUS    0A16   14     F$RLOG    0128*  16    *F$RLOL    0A46   14
  F$RLP2    012A*  16    *F$RLUS    0A2A   14    *F$RMON    038C*  22    *F$RNUM    036F*  22
 *F$ROP     0362*  22    *F$RPAU    058C   11    *F$RPRB    0360*  22     F$RPRE    0CFE   17
 *F$RPRM    0370*  22    *F$RRE     0A02   14    *F$RREB    09FE   14    *F$RRED    09FA   14
 *F$RRER    09F6   14    *F$RRF     0A12   14    *F$RRFB    0A0E   14    *F$RRFD    0A0A   14
 *F$RRFR    0A06   14    *F$RROL    0A3E   14    *F$RRUS    0A22   14     F$RSIO    0A4E   14
  F$RSTO    059E   11    *F$RTCA    0366*  22     F$RTFG    0BB6   16    *F$RTID    0373*  22
  F$RVFB    0028*  16     F$RVP2    002A*  16    *F$RWE     09EA   14    *F$RWER    09E6   14
  F$RWF     09F2   14    *F$RWFR    09EE   14     F$RWRK    0B90   16     F$RWSP    24DE   34
  F$STAT    0004*  17     F$UNIT    015F*  16     F$XBCH    0020*  23     F$XBFR    0027*  23
  F$XBFS    008A*  19     F$XBFX    0028*  23     F$XBFY    002A*  23    *F$XBUI    247A   32
 *F$XBUO    2466   32     F$XBUT    224C   29    *F$XCAL    159D   22     F$XCLS    0D8C   18
  F$XCPX    0022*  23     F$XCPY    0024*  23    *F$XEOF    245C   32     F$XERR    2230   27
  F$XFCB    0EA0   20     F$XFCE    0F68   20    *F$XFND    2462   32     F$XFTL    223A   28
  F$XLIO    237A   31     F$XLOG    15E2   24     F$XLWS    15B6   23    *F$XMON    15B4   22
  F$XPRE    063C   12     F$XPSE    0D7A   17     F$XRED    2394   32     F$XRST    27D2   37
 *F$XRWD    2444   32     F$XSTP    0D82   17     F$XTBE    1588   22     F$XTBL    1228   22
 *F$XTID    159B   22    *F$XTRA    243C   32     F$XVFB    0F74   21     F$XVWS    0C98   16
  F$XWRT    23B2   32     F$XWSO    2504   34     F$XWS1    2506   34     F$XWS2    2508   34
  F$XWS3    250A   34     G$XE01    227A   30     G$XE08    228D   30     G$XE09    22AF   30
  G$XE10    22DC   30    *G$XE11    22F7   30     G$XE12    2312   30     G$XE13    232A   30
  G$XE14    2368   30     MAINB     04E8   4      P$ABUF    0006*  17    *P$AKEY    000C*  17
  P$CCNT    000A*  17     P$ERR     0001*  17    *P$IFA     001C*  17     P$LACN    0016*  17
  P$LUN     0003*  17     P$OP      0002*  17    *P$PASS    0018*  17     P$PFCB    0024*  17
 *P$PRB     0000*  17     P$PRBE    0026*  17    *P$REC1    000D*  17    *P$REC2    000E*  17
  P$RECL    0008*  17    *P$RES1    000C*  17    *P$RES2    001A*  17    *P$SFA     0020*  17
 *P$SFLG    0004*  17    *P$SVCO    0000*  17     P$UFLG    0005*  17     P$ULRL    0012*  17
  P$UPRL    0014*  17     P$UTF1    0010*  17    *P$UTF2    0011*  17    *PX$CUR    0012*  17
 *PX$EVC    0011*  17    *PX$FIL    0010*  17    *PX$FLG    000E*  17    *PX$FST    0014*  17
 *S$APRB    113A   21     S$CLOS    2B5C   38     S$GTCA    27D4   38     S$MAPS    28A6   38
  S$OPEN    2BB6   38     S$RTCA    285A   38     S$STOP    2DBE   38     S$WEOL    2C0E   38
  S$WRIT    2C74   38


 ****  LINKING COMPLETED
```

3.3.5 LOAD COMMAND. The inclusion of the overlay manager is specified by the LOAD command in the Link Editor Control File. Use of the LOAD command is only valid when the IMAGE Format is selected. The overlay manager consists of both read only and read write segments, and the position of the LOAD command in relation to the PROCEDURE and TASK commands determines where the read only parts are to be included in the linked output. The read write parts are always included in the task.

A detailed explanation of the use of the overlay manager is given in Section 2 and in Appendix A. On TXDS, the overlay manager resides on the same diskette as the Link Editor and has the pathname DSCx:TXLOVM/SYS. On DX10, the overlay manager is included as a part of the system subroutine library, DS01.S$SYSLIB. When using automatic overlay loading on DX10, the user must supply a LIBRARY command that specifies the subroutine library.

The format of the LOAD command is as follows:

```
LOAD
```

If the LOAD command is not specified, the default condition is NOLOAD.

The following examples illustrate the use of the LOAD command in a two procedure link edit. The first example places the reentrant parts in procedure P1, and the second example includes the reentrant parts in procedure P2. The non-reentrant parts are always included in the task. The third example includes all of the overlay manager in the task (PHASE 0).

NOTE

> When the LOAD command is used in a PROCEDURE definition, that procedure only can be shared by multiple copies of the same task segment (not by different tasks).

Example 1.

```
LIBRARY              .USER
LIBRARY              .S$SYSLIB
FORMAT               IMAGE
PROCEDURE            P1
INCLUDE              (X)
LOAD
PROCEDURE            P2
INCLUDE              (Y)
TASK                 T1
INCLUDE              (Z)
 .
 .
 .
END
```

Example 2:

```
LIBRARY              .USER
LIBRARY              .S$SYSLIB
FORMAT               IMAGE
PROCEDURE            P1
INCLUDE              (X)
PROCEDURE            P2
LOAD
INCLUDE              (Y)
TASK                 T1
INCLUDE              (Z)
 .
 .
 .
END
```

Example 3:

```
    LIBRARY                    .USER
    LIBRARY                    .S$SYSLIB
    FORMAT                     IMAGE
    PROCEDURE                  P1
    INCLUDE                    (X)
    PROCEDURE                  P2
    INCLUDE                    (Y)
    TASK                       T1
    INCLUDE                    (Z)
    LOAD
     .
     .
     .
    END
```

When no procedures are used, both reentrant and non-reentrant parts are included in the task.

3.3.6 NOLOAD COMMAND. The NOLOAD command specifies that the overlay manager and its tables are not to be included in the linked output. NOLOAD is the default condition. The format of the command is as follows:

```
    NOLOAD
```

3.3.7 SHARE COMMAND. The SHARE command is used to specify modules that are to share the same data area ($DATA). The syntax of the command is as follows:

```
    SHARE   <module name>,<module name>[. . .]
```

The first module included by an INCLUDE command or as the result of a library search defines the maximum size of the area. SHARE commands can be continued by repeating any task name that has been previously referenced by a SHARE command. The SHARE command only applies within a phase, and it cannot cross a phase boundary. An example of the SHARE command is as follows:

```
    SHARE   MOD1,MOD2,MOD3
```

3.3.8 PARTIAL LINK EDITS. The Link Editor can perform partial Link Edits of single phases and output either normal tagged object or compressed tagged object. The output of a partial Link Edit is not executable and must be linked again before the program can be executed. Partial linking is useful when a single phase is to be used in more than one program since it can be linked by itself once, and then linked again into a procedure or phase in other Link Edits. Partial link outputs are also used to build sequential libraries. The PARTIAL command is used to specify a partial Link Edit.

Commands are also provided for use in conjunction with the PARTIAL command to define the scope of symbols defined (DEFed) within the phase. Symbols within a phase can be defined as being global or local. Global symbols are externally defined for the module and may be relinked. External definition data is included in the partial link output for each global symbol. Local (or not global) symbols are not externally defined and can be referenced only within the phase. Link Editor commands are provided to specify only certain symbols or all symbols to be global, and a command is provided to exempt certain symbols or all symbols from the global definition.

3.3.8.1 PARTIAL Command. The PARTIAL command performs a partial link edit and outputs either a normal tagged object or compressed tagged object output file. The output of a partial Link Edit is not executable and must be linked again without the PARTIAL command before the program can be executed. Partial linking is only allowed for single phases. The PARTIAL command causes the Link Editor to do the following:

- Resolve all external references (REFs) that are defined by any module included in the partial link.
- Retain all entry points in the partial link as an entry in the output (subject to GLOBAL, NOTGLOBAL, and ALLGLOBAL commands).
- Retain the common tags updating common numbers.
- Output one data section that is the total of all input data sections (subject to the SHARE command).
- Resolve all SHARE references.

The user should note that automatic overlay loading information is not included as a part of the partial output (i.e., the LOAD command cannot be used in a partial link edit)..

The syntax of the command is as follows:

```
PARTIAL

PART
```

NOTE

> A PARTIAL command may only be used in a link edit which contains a TASK or PHASE 0 command. PARTIAL may not be used with higher numbered phases.

When the PARTIAL command is used, the command stream must contain a TASK or PHASE command to define a name for the phase. However, the phase level and name assigned do not have to be used in future links. For example, the following illustrates the responses to the prompts presented in a DX10 system and the command.stream to produce a partial link.

Prompt Responses:

```
XLE

EXECUTE LINKAGE EDITOR

           CONTROL ACCESS NAME:   VOL2.EXONE.CONT

     LINKED OUTPUT ACCESS NAME:   VOL2.EXTWO.MODA

           LISTING ACCESS NAME:   VOL2.LIST.ONE

                   PRINT WIDTH:   80
```

Control Stream:

```
PARTIAL

LIBRARY     VOL3.EXAMPS

PHASE       O,OVLAY1

INCLUDE     (MOD1)

INCLUDE     (MOD2)

END
```

The preceding example causes a partial link to be performed that includes two modules (MOD1 and MOD2) from a random library. The phase is defined as being level zero and is named 0VLAY1. The output is written to file MODA.

The following control stream illustrates the inclusion of the output from the preceding example into another DX10 link edit:

```
LIBRARY         VOL3.MODS

PROC            P1

INCL            (MOD4)

PROC            P2

INCL            (MOD5)

PHASE           O,ROOTPH

INCL            VOL2.CATA.TASK

PHASE           1,LVL1
```

```
INCL           VOL2.EXTWO.MODA

PHASE          1,LVL1B

INCL           (MOD6)

END
```

The preceding control stream causes the output of the partial link edit to be included as a phase one overlay, with the overlay named LVL1A.

The following illustrates an example of the responses to the TXDS prompts to activate the Link Editor:

```
PROGRAM:  :TXSLNK/SYS          Activate the Link Editor

  INPUT:  DSC2:FILEA/CTL       Control file pathname

 OUTPUT:  DSC2:LNKOUT/OBJ      Output pathname, accept the system default
                               printer for the load map and the TXSLNK
                               diskette for scratch files.
```

The following is an example of the use of the PARTIAL command in a TXDS Link Edit:

```
PARTIAL

INCLUDE      DSC2:MODA/OBJ       Include file MODA on disc two

INCLUDE      DSC2:MODB/OBJ       Include file MODB on disc two

FIND         DSC2:TXLOB/LIB      Use library TXLOB to resolve references

END
```

The output of the preceding partial Link Edit is included in the following example:

```
TASK        TSK2

INCLUDE     DSC2:LNKOUT/OBJ

INCLUDE     DSC2:MODA/OBJ

FIND        DSC2:TXLOB/LIB

END
```

3.3.8.2 NOTGLOBAL Command. The NOTGLOBAL command identifies symbols defined in the current phase as not global. The default is ALLGLOBAL. The command is optional and several commands may be used in a phase. The syntax for the command is as follows:

```
NOTGLOBAL   [<symbol>] [,<symbol>] [, . . .]
```

If no operand is specified, all symbols are NOTGLOBAL, except those specified in a GLOBAL command.

The command may include several operands, limited only by the maximum size of the record. Each operand is a symbol not to be processed as a global symbol. The following examples illustrate the use of a NOTGLOBAL command:

DX10:

```
PARTIAL
LIBRARY      .VOL3.EXAMPS
PHASE        0,OVLAY1
INCL         (MOD1)
INCL         (MOD2)
NOTGLOBAL    ENTR3,ENTR5,ENTR7
END
```

TXDS:

```
PARTIAL
PHASE        0,OVLAY1
INCLUDE      DSC3:MODC/OBJ
FIND         DSC2:TXLIBR/LIB
NOTGLOBAL    ENTR,ENTR5,ENTR7
END
```

The NOTGLOBAL command exempts three symbols, ENTR3, ENTR5, and ENTR7 from the global definition. These three symbols are local to the phase.

3.3.8.3 ALLGLOBAL Command. The ALLGLOBAL command declares all external definitions in the current phase to be global symbols. The ALLGLOBAL command only applies to partial links. This command has the same effect as a GLOBAL command with all the symbols of the phase as operands. The command is optional and requires only the command name, as in the following examples:

DX10:

```
PARTIAL
LIBRARY      VOL3.EXAMPS
PHASE        0,OVLAY1
```

```
  INCL          (MOD1)

  INCL          (MOD2)

  ALLGLOBAL

  END

TXDS:

  PARTIAL

  PHASE         0,OVLAY1

  INCLUDE       DSC3:MODA/OBJ

  FIND          DSC2:TXLIBR/LIB

  ALLGLOBAL

  END
```

All symbols externally defined (DEFed) in the included modules are externally defined in the output module.

3.3.8.4 GLOBAL Command. The GLOBAL command identifies symbols defined in the current phase as global. Note that this command only applies to partial Link Edits and is only used in conjunction with the NØTGLOBAL command. Global symbols are externally defined for the module, and may be relinked. For each global symbol, external definition data is included in the linked object module. The command is optional and, when it is used, the command should be the last command of a phase. The syntax for the command is as follows:

```
GLOBAL   [<symbol>] [,<symbol>] [, . . .]
```

The command may include several operands, limited only by the maximum size of the record. If no operands are specified, the command functions as ALLGLOBAL. Each operand is a symbol to be processed as a global symbol. The following examples illustrate the use of the GLOBAL command:

```
DX10:

  PARTIAL

  LIBRARY       VOL3.EXAMPS

  PHASE         0,OVLAY1

  INCLUDE       (MOD1)

  INCLUDE       (MOD2)

  NOTGLOBAL

  GLOBAL        ENTER1,ENTER2

  END
```

TXDS:

```
PARTIAL

PHASE       0,OVLAY1

INCLUDE     DSC2:MOD1/OBJ

FIND        DSC2:TXLOB/LIB

NOTGLOBAL

GLOBAL      ENTER1,ENTER2
```

The GLOBAL command in this example causes the symbols ENTER1 and ENTER2 to be externally defined within the module. Note that the symbols must have been previously defined in an object module.

3.3.9 DUMMY COMMAND. The DUMMY command causes the Link Editor to suppress the linked output for the phase, procedure, or task in which it appears. No linked output is written if the DUMMY command precedes the first PHASE, PROCEDURE, or TASK command. The syntax of the command is as follows:

```
DUMMY

DUMM
```

The DUMMY command may be used in either of two ways:

- If the DUMMY command precedes the first PHASE, PROCEDURE, or TASK command in the control file, no linked output is generated for that or subsequent phases. This method is used when only a load map listing is required, or for error identification.

- If the DUMMY command follows a PHASE, PROCEDURE, or TASK command, no output is generated for that specific phase. This is appropriate when only a portion of the linked output is needed, such as linking a new task to a previously installed procedure.

NOTE

In a Link Edit containing two procedures, the second procedure may be dummied only if the first procedure is dummied.

The following are examples of the use of the DUMMY command to produce a load map listing:

```
LIBRARY     CATA.APPL.OBJ

DUMMY

PROC        PROC1

INCL        (MOD1)
```

```
  INCL        CATB.APPL.OBJ.MOD2
  PROC        P2
  INCL        CSO2
  INCL        (MOD2)
  TASK        TSK1
  INCL        (MOD4)
  INCL        CATB.APPL.OBJ.MOD3
  END
TXDS:
  DUMMY
  TASK        TSK1
  INCL        DSC3:MOD1/OBJ
  INCL        DSC2:MOD2/OBJ
  FIND        DSC3:TXLIBR/LIB
  FIND        DSC2:TXLOB/LIB
  END
```

No linked output would be generated by the above examples.

The following DX10 example illustrates the use of a DUMMY command to link a new task to a previously installed procedure (PROC1). No output is generated for PROC1, but all references are resolved.

```
  FORMAT   IMAGE
  LIBRARY  CATB.PROGS.OBJ
  PROC     PROC1
  DUMMY
  INCL     (MODA)
  INCL     (MODC)
  PROC     P2
  INCL     CATC.APPL.FILEA
```

```
INCL        (MODB)

TASK        TSK2

INCL        CATC.APPL.FILEC

END
```

3.3.10 ADJUST COMMAND. The ADJUST command is used to specify alignment of a phase, or of a module within a phase. The format of the command is as follows:

```
ADJUST  n
```

where n is a specified power of two bytes, which must be less than 16. When the operand is omitted or equal to zero, alignment is on the next word boundary.

Adjustment on a boundary is useful in such areas as diagnostic programming.

When the ADJUST command appears immediately before a PHASE command, the next phase and all subsequent phases of the same level and with the same parent node are aligned on the specified boundary, relative to the beginning of the program. For example, if the specified adjustment is 2^5, the phase is aligned at the next 32-byte boundary relative to the beginning of the program. The following is an example of the use of the ADJUST command to align a phase:

```
PROC      P1

INCL      VOL1.OBJ.MOD1

PHASE     0,ROOT

INCL      VOL1.OBJ.MOD2

ADJUST    5

PHASE     1,LEVEL1

INCL      VOL2.OBJ.MODA

INCL      VOL2.OBJ.MODB

PHASE     2,LEVEL2

INCL      VOL2.OBJ.MODC

INCL      VOL1.OBJ.MOD3

PHASE     1,LEVEL1A

INCL      VOL1.OBJ.MOD4

PHASE     2,LEVEL2A

INCL      VOL1.OBJ.MOD5

END
```

In the example, the phases LEVEL1 and LEVEL1A are aligned on a 32-byte boundary, relative to the beginning of the program.

When the ADJUST command follows a PHASE command but precedes an INCLUDE command, the next module in that phase is aligned on the specified boundary, relative to the beginning of the phase. If ADJUST follows a PHASE command but precedes all INCLUDES in the phase, the effect is the same as when ADJUST precedes PHASE. The following are examples of the use of the ADJUST command to align a module:

DX10:

```
PROC      P1
INCL      VOL1.OBJ.MOD1
PHASE     O,ROOT
INCL      VOL1.OBJ.MOD4
PHASE     1,LVL1A
INCL      VOL.OBJ.MOD2
ADJUST    5
INCL      VOL1.OBJ.MOD3
INCL      VOL1.OBJ.MOD6
PHASE     2,LVL2A
INCL      VOL2.OBJ.MOD1
INCL      VOL2.OBJ.MOD2
PHASE     1,LVL1B
INCL      VOL2.OBJ.MOD7
INCL      VOL2.OBJ.MOD6
PHASE     2,LVL2B
INCL      VOL2.OBJ.MOD5
PHASE     3,LVL3B
INCL      VOL2.OBJ.MOD4
INCL      VOL2.OBJ.MOD3
END
```

In this example, the module with the pathname VOL1.OBJ.MOD3 is aligned on a 32-byte boundary relative to the origin of phase LVL1A.

```
TXDS:

  TASK       TSK1

  INCLUDE    DSC2:MODB/OBJ

  ADJUST     5

  INCLUDE    DSC3:MODC/OBJ

  INCLUDE    DSC2:MODE/OBJ

  INCLUDE    DSC2:MODG/OBJ

  FIND       DSC2:TXLIBR/LIB

  END
```

The TXDS example causes module MODC to be aligned (started) on the next 32-byte boundary relative to the beginning of the task.

The specified power of two adjustments can be any decimal number from one to fifteen. A value greater than fifteen causes an error condition.

3.4 SYMBOL PROCESSING

The symbol processing commands described in the following paragraphs are those that define how symbols contained in the object modules are to be handled. Object modules being linked for a DX10 system can have the object module symbol table included in the linked object output module. Inclusion of the symbol table allows for symbolic debugging of the program by the DX10 SCI Debug commands (refer to the *DX10 Operating System Developmental Operations Guide,* manual number 946250-9704). Information relative to symbols and symbol tables can be found in the *Model 990 Computer TMS 9900 Microprocessor Assembly Language Programmer's Guide,* manual number 943441-9701.

3.4.1 SYMT COMMAND — DX10 ONLY. The SYMT command causes the Link Editor to include the symbol tables from the object modules that are included in the link operation. Note that the SYMT command has no effect when the image format is selected. Object modules contain symbol tables only if the SYMT assembler option was used in the assembly. If an overlay structure is used and SYMT is also used, the SYMT command must appear before the end of the root phase (phase 0). The object modules produced by the assemblers may include symbol tables consisting of G and H tag character fields as described in the *Model 990 Computer TMS 9900 Microprocessor Assembly Language Programmer's Guide.* Figure 3-3 shows two object modules and the symbol table fields for the modules. To identify the module in which the symbol occurs, the Link Editor inserts an I tag character followed by a four-character hexadecimal field and an eight-character decimal field. The syntax of the command is as follows:

```
SYMT
```

3.4.2 NOSYMT COMMAND. Specification of the NOSYMT command causes the Link Editor to omit the symbol tables from the linked object module. The command may appear anywhere in the control file, except that if an overlay structure is used and NOSYMT is desired, the command must appear in the root phase (phase 0). The syntax of the command is as follows:

```
NOSYMT

NOSY
```

If NOSYMT is not specified, the default condition is SYMT (DX10 only), as previously described. On TX990, omitting the NOSYMT command causes certain object tags to be included in the linked output. NOSYMT provides for more compact object code.

3.5 EXECUTION AND LISTING OPTIONS

The commands described in the following paragraphs are those which control the execution of the Link Editor and define the listing output options. The commands that control the execution include those that define the format of the output object code and define how errors are to be handled during execution of the Link Editor. Listing options include the specification of page ejects between the load maps for each phase, and whether the load map is to be printed. The user is also able to specify what is to be included in the map listing.

3.5.1 FORMAT COMMAND. The FORMAT command defines the format of the linked output. Three formats are supported by the Link Editor, as defined by the following:

> Normal Tagged Object — Consists of ASCII characters and ASCII control characters (TAGS). Execpt for COBOL programs, this format must be output to a sequential file. Except for COBOL, the normal tagged object is not executable and must be installed (DX10) or loaded (TXDS) as a task/procedure/overlay before it can be executed. COBOL tagged object can be executed by use of the Execute COBOL Program. or the Execute COBOL Program in Foreground DX10 System Command Interpreter commands. Normal tagged object format is generally transportable between 990 computer systems, and can be linked again if generated using a PARTIAL command.

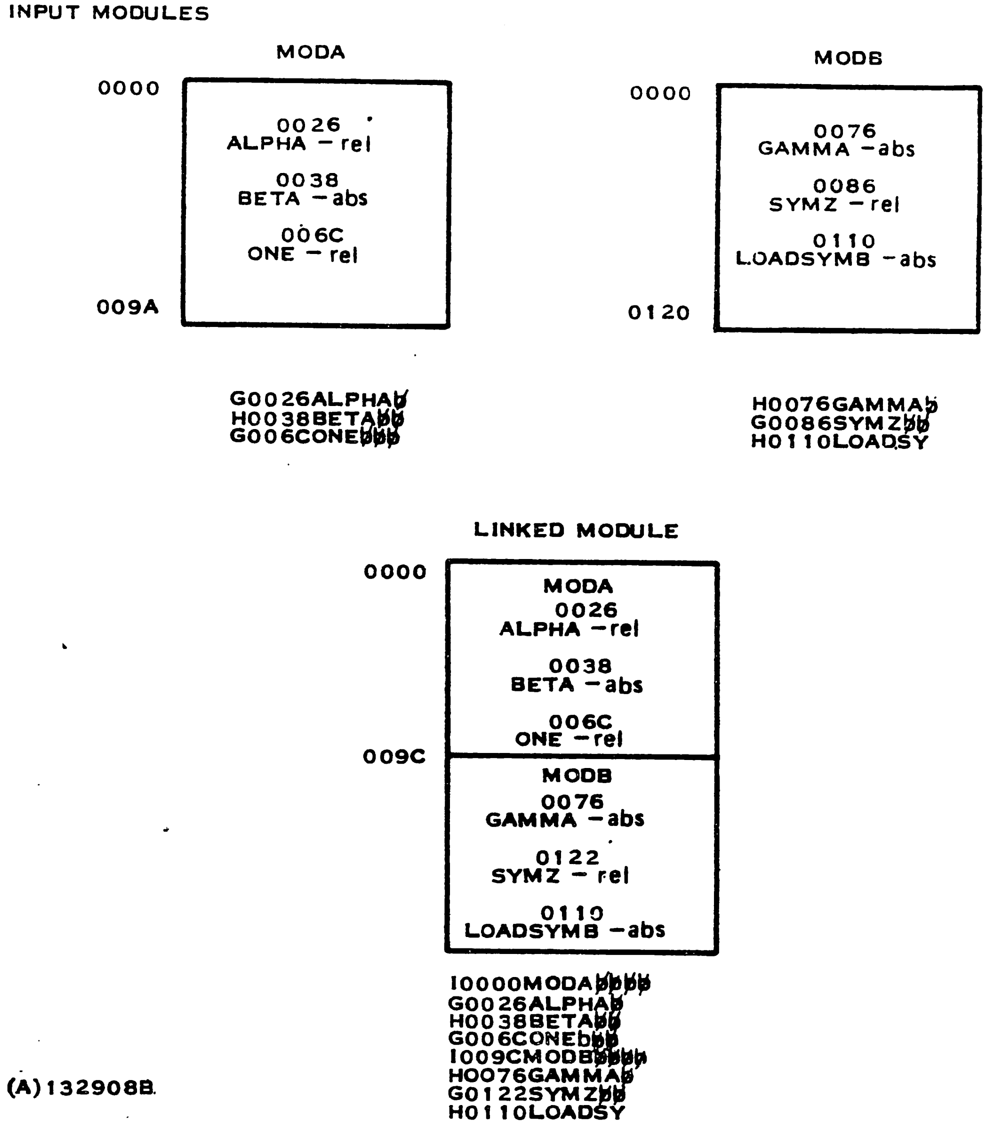


Figure 3.2. Symbol Table for Linked Object Module

Compressed Tagged Object — This format is a condensed version of the normal tagged object and can only be output to a file that supports binary data. Except for this, compressed object is treated as normal tagged object. Compressed object results in a savings of approximately 47% of disk space as compared to the normal tagged format. The difference between compressed and normal object is that in compressed the numeric fields are expressed in binary instead of ASCII. Also, in compressed format, the zero tag is used instead of binary '01'.

Memory Image Format — Memory image format appears exactly as the program appears in memory and is loaded directly to a DX10 or TXDS Program File or a DX10 Image File.

The syntax of the FORMAT command is as follows:

```
          ( ASCII        )               (            )
FORMAT    { COMPRESSED   }   [,REPLACE] { <priority> }
          ( IMAGE        )               (    4       )
```

If the IMAGE format is selected, the user may also enter the REPLACE parameter, which causes new procedures, tasks, and overlays to replace existing ones, with the same names, in the program file (defined by the LINKED OUTPUT ACCESS NAME), and the priority parameter defines the priority (1, 2, 3 or 4) at which the task is to execute, with the default being 4.

In a DX10 system, the IMAGE format can also be used to install the Linked Output on an Image File. The System Image File is a unique file type in that it contains the loadable image of an operating system, and is used for the Initial System Load. The System Image File can contain either a DX10 or a TX990 image.

The Link Editor cannot be used to install memory resident, system, or privileged tasks on a program file. These tasks must be installed using the Install Task Supervisor call, or the Install Task System Command Interpreter command.

The default format is ASCII (Normal Tagged Object).

3.5.2 MAP COMMAND. The MAP command allows the user to control the format of the link map. The user may specify that only referenced names be listed, or that only names which don't begin with a specified character string be listed. The purpose of this command is to suppress the listing of external symbols in runtime library subroutines. The syntax of the command is as follows:

```
MAP        {REFS                                  }
           {NO<'string>[,NO<'string'>] [, . . .] }
```

For example, the following shows two uses of the MAP command. The first shows the use of the MAP command to cause only referenced names to be output in the Link Map listing file, and the second example shows the use of the MAP command to suppress the listing of names that begin with 'S$' and 'CXS'.

Example 1:

```
   .
   .
   .
MAP        REFS
   .
   .
   .
END
```

Example 2:

```
   .
   .
   .
MAP        NO'S$',NO'CXS'
```

3.5.3 NOMAP COMMAND. The NOMAP command causes the Link Editor to suppress the load map listing. The following information is still printed on the list file:

- Length of task and procedure(s)
- Unresolved references
- Release number of the Link Editor
- If the format is memory image and the file is a program file, the number of output records.

The syntax of the command is as follows:

```
NOMAP

NOMA
```

3.5.4 PAGE/NOPAGE COMMANDS. The PAGE/NOPAGE commands allow the user to control the format of the output listing. The PAGE command causes page ejects to separate the beginnings of the load maps for each phase, and the NOPAGE command specifies that page ejects do not separate the phases. The default is PAGE. The syntax of the commands is as follows:

```
PAGE

NOPAGE
```

3.5.5 ERROR/NOERROR COMMANDS. The ERROR/NOERROR commands allow the user to specify the way errors are to be handled by the Link Editor. The NOERROR command causes the Link Editor to terminate processing whenever an error occurs. The ERROR command allows the Link Editor to continue processing the control commands when an error occurs. In addition, the Link Editor attempts to recover from the error and to complete the linking operation. Error recovery consists of not processing the line in which the error occurs. Error messages are generated for all errors encountered. If the Link Editor is unable to process an INCLUDE command, processing always terminates. The default mode is NOERROR. The syntax of the commands is as follows:

```
ERROR

NOERROR
```

3.6 ABSOLUTE MEMORY PARTITIONING

The commands described in the following paragraphs provide the user with the means to create program modules that will be run on systems using a combination of Read-Only-Memory (ROM) and Read/Write memory (RAM). These Link Editor commands allow the user to specify the beginning of a Read Only area, the beginning of the Read/Write area, and the beginning of common data areas.

The commands described in this paragraph and the following subparagraphs do not apply to generating linked output for execution on DX10 or TXDS. These commands are for linking standalone systems, or those that are linked to the Texas Instruments Execute Only Operating System (EX990).

The Link Editor accepts input object that has been defined as being program, data, or common segments. Program segments are defined in the assembler by the PSEG directive. Program segments generally contain instructions and nonvariable data (read-only). Data segments are defined by the DSEG assembler directive and generally contain variable data (read/write). Data segments are labelled by the Link Editor as $DATA. Common segments are defined by the CSEG assembler directive and contain data that may be shared by more than one module.

The high-level language compilers, COBOL and FORTRAN, automatically output code that is defined as data, program, or common.

The commands provided to specify Absolute Memory Partitioning are the PROGRAM, COMMON, and DATA commands. When any of these commands are used, the PROGRAM command must be used. If COMMON or DATA are used without PROGRAM, they are ignored.

Note that the output of the Link Editor cannot be installed in a DX10 system or loaded in a TXDS system if any of these commands are used.

3.6.1 PROGRAM COMMAND. The PROGRAM command defines the starting location counter value for program segments. The command may be used more than once, and the first PROGRAM command must appear before the first INCLUDE command. Note that the PROGRAM command is specifically designed to allow users to link edit programs for other Texas Instruments Model 990 Computers that have special memory configurations (combinations of ROM, RAM, and Common Memory). These programs are executed in a stand-alone environment and do not require the support of the operating system, or they can be linked to an operating system such as the Texas Instruments Execute Only Operating System (EX990). The syntax of the PROGRAM command is as follows:

```
PROGRAM    <base>
```

where the <base> parameter is up to five digits. A preceding ">" or 0 indicates a hexadecimal value. The entered value defines the beginning address for program segments. Examples of the command are as follows:

PROGRAM	01F00	Program segment at location $1F00_{16}$
PROG	01F00	Same as preceding
PROG	7936	Program segment at location 7936_{10}, or $1F00_{16}$.

Use of the PROGRAM command without the COMMON and DATA command causes a linked output that is to be loaded at the specified address (base).

3.6.2 DATA COMMAND. The DATA command defines the starting location counter value for data segments. The command may appear more than once, and the first DATA command must appear before the first INCLUDE command. If the DATA command is omitted, the starting location for each data area defaults to the end location of the corresponding program area. This command can only be used in conjunction with the PROGRAM command and is ignored if used without the PROGRAM command.

The syntax of the DATA command is as follows:

```
DATA   <base>
```

where <base> is up to five digits. A preceding ">" or 0 indicates a hexadecimal value. Examples of the DATA command are as follows:
con

DATA	01000	Data segment begin at location 1000_{16}
DATA	01000	Same as preceding
DATA	4096	Data segment at location 4096 (1000_{16})

3.6.3 COMMON COMMAND. The COMMON command is used to define the starting location counter value for the specified common data areas. Commons that are to be loaded at the specified address must be specifically identified within the command. More than one COMMON command may be used, and a continuation can be performed by repeating the command using a previously named COMMON instead of a starting location. The named commons are allocated in the order the definitions appear in the object module. All commons not given a starting location are loaded after the last data area encountered by the Link Editor during the linking process. Note that the COMMON command can only be used in conjunction with the PROGRAM command, and it is ignored if used alone. The syntax of the command is as follows:

```
COMMON    <base>[,<name>] [,<name>]
```

where the <base> parameter is the starting location of the common area and can be expressed as either a decimal or hexadecimal number up to five digits long. A preceding ">" or 0 indicates a hexadecimal value. The <name> parameter is the name of the common, and more than one may be specified. If no <name> is given, the common data area is loaded after the last data area encountered. The following are examples of the COMMON command:

COMMON	01000,COMA	Load COMA at location 1000_{16}
COMMON	01000,COMA	Same as preceding
COMMON	COMA,COMB	COMB immediately follows COMA
COMMON	4096,COMA,COMB	Same as preceding two commands.

SECTION IV

LINK EDITOR EXAMPLES

4.1 GENERAL

This section contains examples of Link Editor runs performed on DX10 and TXDS systems. The purpose of these examples is to provide the user with a visual reference for the use of the Link Editor.

The Link Editor functions the same on both TXDS and DX10 systems. However, the interfaces with the user differ between the two systems. Details on operating the Link Editor with DX10 are given in Section V, and details on operating the Link Editor with TXDS are given in Section VI.

4.2 DX10 LINK MAP

The example shown in figure 4-1 illustrates the DX10 format of the output listing generated by the Link Editor. This example linked three modules to form a task. The three modules are named SUBT1, SUBR1, and MODX, and the task itself is named LSCAN. All of the modules are files within the random library defined by the LIBRARY .LEE.EXO command. This example was generated on a DX10 system.

Page one in the example, titled COMMAND LIST, is the list of the Control Stream used to control the linking operations. This list is generated at the beginning of each Link Edit. Page two, titled LINK MAP, lists the parameters entered at the terminal when the Link Editor was activated. This page also gives the format of the output from the Link Editor (ASCII in the example).

The last page of the example, page three, is the actual link map. The first line of the listing defines the phase, as shown in the following:

```
PHASE 0,   LSCAN   ORIGIN = 0000   LENGTH = 0056
```

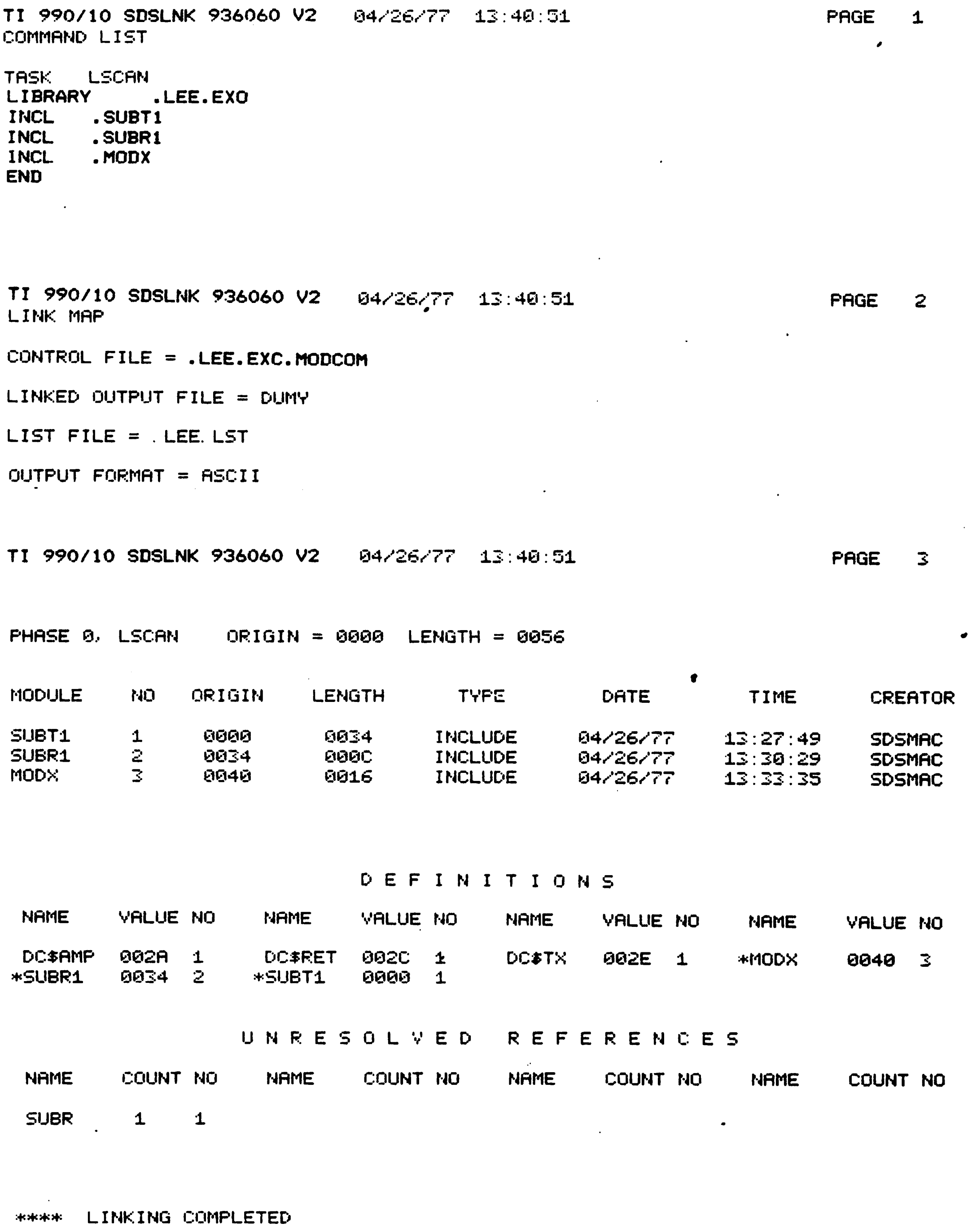
In an overlay structured program, phase 0 is the root, or memory resident, phase. Note, however, that the TASK command also causes a PHASE 0 definition. LSCAN is the name assigned to the task by the TASK command. ORIGIN = 0000 specifies that the phase begins at location 0000_{16} relative to the beginning of the program. The origin of phase 0 (whether specified by a PHASE or a TASK command) is always a multiple of 32-bytes, as is the origin of a second procedure on a two attached procedure Link Edit. For example, if a Link Edit consisted of two procedures, each eight-bytes in length, and a task (phase 0), the origin would be as follows:

```
Procedure One —  0000

Procedure Two —  0020

Phase 0       —  0040
```

When the PROGRAM command is not used, the Link Editor output is relocatable and the origin of each program is defined as being 0000. The PROGRAM command specifies an origin. LENGTH = 0056_{16} specifies the actual number of bytes of memory required to hold the phase.

The next line of the listing is the heading for the module definition. The entries below the heading define each module included in the phase. The heading line is as follows:

```
MODULE   NO   ORIGIN   LENGTH   TYPE   DATE   TIME   CREATOR
```



```
TI 990/10 SDSLNK 936060 V2   04/26/77  13:40:51                        PAGE   1
COMMAND LIST

TASK   LSCAN
LIBRARY     .LEE.EXO
INCL   .SUBT1
INCL   .SUBR1
INCL   .MODX
END


TI 990/10 SDSLNK 936060 V2   04/26/77  13:40:51                        PAGE   2
LINK MAP

CONTROL FILE = .LEE.EXC.MODCOM

LINKED OUTPUT FILE = DUMY

LIST FILE = . LEE. LST

OUTPUT FORMAT = ASCII


TI 990/10 SDSLNK 936060 V2   04/26/77  13:40:51                        PAGE   3


PHASE 0, LSCAN    ORIGIN = 0000  LENGTH = 0056


MODULE     NO   ORIGIN     LENGTH      TYPE        DATE         TIME       CREATOR

SUBT1      1     0000       0034     INCLUDE    04/26/77     13:27:49     SDSMAC
SUBR1      2     0034       000C     INCLUDE    04/26/77     13:30:29     SDSMAC
MODX       3     0040       0016     INCLUDE    04/26/77     13:33:35     SDSMAC


                                D E F I N I T I O N S

 NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 DC$AMP    002A  1      DC$RET    002C  1      DC$TX     002E  1     *MODX      0040  3
*SUBR1     0034  2     *SUBT1     0000  1

                         U N R E S O L V E D   R E F E R E N C E S

 NAME      COUNT NO     NAME      COUNT NO     NAME      COUNT NO     NAME      COUNT NO

 SUBR        1    1



****  LINKING COMPLETED
```

Figure 4-1. Linked Map Example

These entries are defined as follows:

MODULE — Specifies the names of the modules included in the phase.

NO — Specifies the number of the module within the phase (used for reference in other parts of the Link Map).

ORIGIN — Defines the beginning of the module relative to the beginning of the program.

LENGTH — Specifies the length of the module, in bytes.

TYPE — Specifies the method by which the module was included in the phase; i.e., INCLUDE, FIND, SEARCH command.

DATE — The date the module was created, if present (some assemblers do not supply this information).

TIME — The time the module was created, if present (some assemblers do not supply this information).

CREATOR — The assembler or compiler that generated the module (SDSMAC, FTN990, etc.).

The next section of the listing has the following heading:

```
                    DEFINITIONS

NAME   VALUE   NO   NAME   VALUE   NO   NAME   VALUE   NO
```

The entries under these headings describe all external definitions (DEFs) in the phase and have the following meanings:

NAME — The symbol specified by the DEF statement

VALUE — The address within the phase associated with the symbol

NO — The number of the module within the phase in which the symbol is DEFed.

Names that are DEFed within the phase but not referenced (REFed) within the program are preceded by an asterisk (*). Symbols that are self-defining (i.e., absolute) are identified by a trailing asterisk (*).

The final section of the listing defines any references that are unresolved within the phase. The heading appears as follows:

```
              UNRESOLVED REFERENCES

NAME   COUNT   NO   NAME   COUNT   NO   NAME   COUNT   NO
```

with the entries under these headings having the following meanings:

NAME — The symbol that was referenced and could not be found

COUNT — The number of times the symbol was referenced

NO — The module within the phase in which the reference occurred.

Unresolved references cause a warning message to be output at the end of the link map. The message is of the form:

```
n   WARNINGS
```

where n is the number of unresolved references.

NOTE

> Partial link edits do not produce a warning message for unresolved references.

The end of the Link Edit processing is indicated by the following message:

```
**** LINKING COMPLETED
```

4.3 LINK EDITOR EXAMPLES — DX10

The following paragraphs contain examples of Link Edits on a DX10 system. Provided for each example is the complete Link Map, as described in paragraph 4.2, and the parameters entered when the Link Editor is called from a VDT.

4.3.1 SINGLE TASK, NO PROCEDURE EXAMPLE. The example shown in figure 4-2 illustrates the use of the Link Editor to build a task consisting of two modules with no attached procedures. The parameters entered in response to the prompts presented at the VDT are as follows:

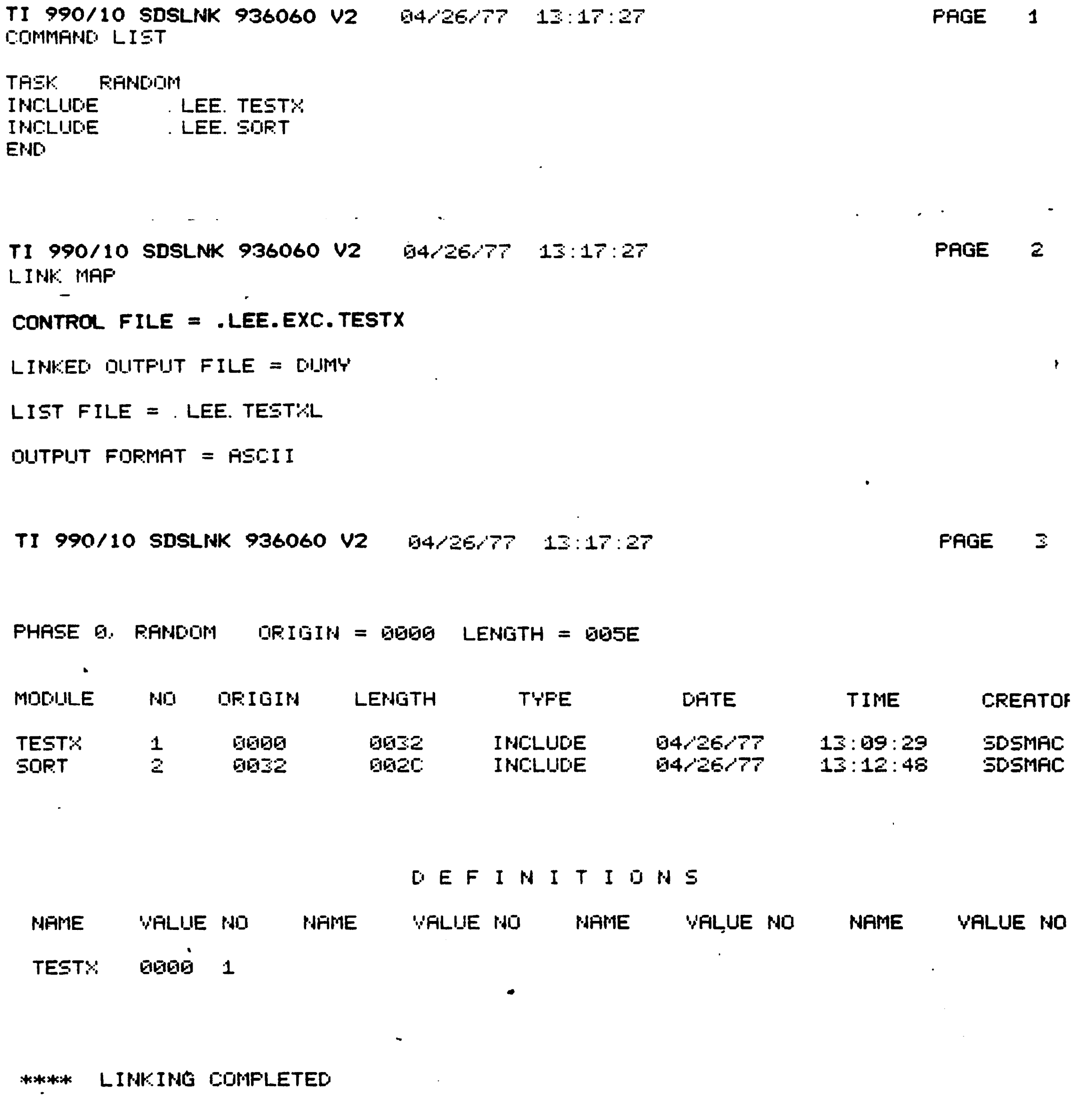
```
XLE

EXECUTE LINKAGE EDITOR

          CONTROL ACCESS NAME:   .LEE.EXC(TESTX)

    LINKED OUTPUT ACCESS NAME:   DUMY

          LISTING ACCESS NAME:   .LEE.TESTXL

                  PRINT WIDTH:   80
```

Note that no linked output is created since the output access name default, DUMY, was accepted. The default value was also used in response to the LINE WIDTH prompt.

The control stream defines the task name as being RANDOM, with files TESTX and SORT included by use of the INCLUDE command. The default format, ASCII, is used.

The Link Map shows that PHASE 0, RANDOM, begins at address 0000_{16} (relative to the beginning of the program) and has a length of $005E_{16}$ bytes. Module TESTX is 32_{16} bytes in length and begins at relative address 0000, and module SORT is $2C_{16}$ bytes in length and begins at relative address 0032_{16}.

Only one external definition, TESTX, is made.



```
TI 990/10 SDSLNK 936060 V2   04/26/77  13:17:27                        PAGE   1
COMMAND LIST

TASK   RANDOM
INCLUDE      .LEE.TESTX
INCLUDE      .LEE.SORT
END


TI 990/10 SDSLNK 936060 V2   04/26/77  13:17:27                        PAGE   2
LINK MAP

CONTROL FILE = .LEE.EXC.TESTX

LINKED OUTPUT FILE = DUMY

LIST FILE = .LEE.TESTXL

OUTPUT FORMAT = ASCII


TI 990/10 SDSLNK 936060 V2   04/26/77  13:17:27                        PAGE   3


PHASE 0, RANDOM   ORIGIN = 0000  LENGTH = 005E

MODULE    NO   ORIGIN    LENGTH     TYPE        DATE       TIME      CREATOF

TESTX     1     0000      0032     INCLUDE    04/26/77   13:09:29   SDSMAC
SORT      2     0032      002C     INCLUDE    04/26/77   13:12:48   SDSMAC


                               D E F I N I T I O N S

 NAME     VALUE NO    NAME     VALUE NO    NAME     VALUE NO    NAME     VALUE NO

 TESTX    0000  1



****  LINKING COMPLETED
```

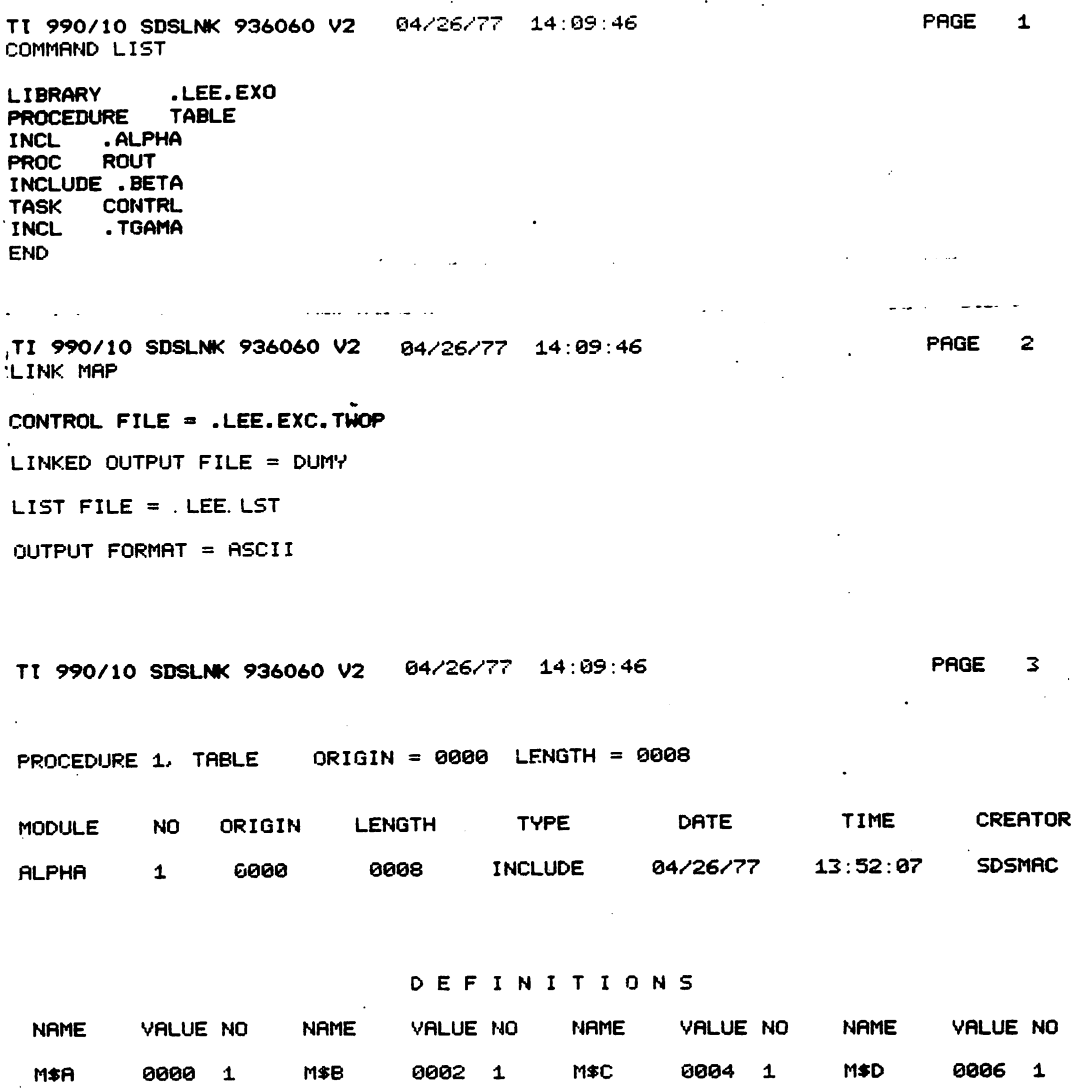
Figure 4-2. Single Task, No Procedure Example

4.3.2 TASK WITH TWO ATTACHED PROCEDURES EXAMPLE. The example shown in figure 4-3 is a Link Edit for a program having a task, CONTRL, and two attached procedures, TABLE and ROUT. The parameters entered when the Link Editor is activated from the VDT are as follows:

XLE

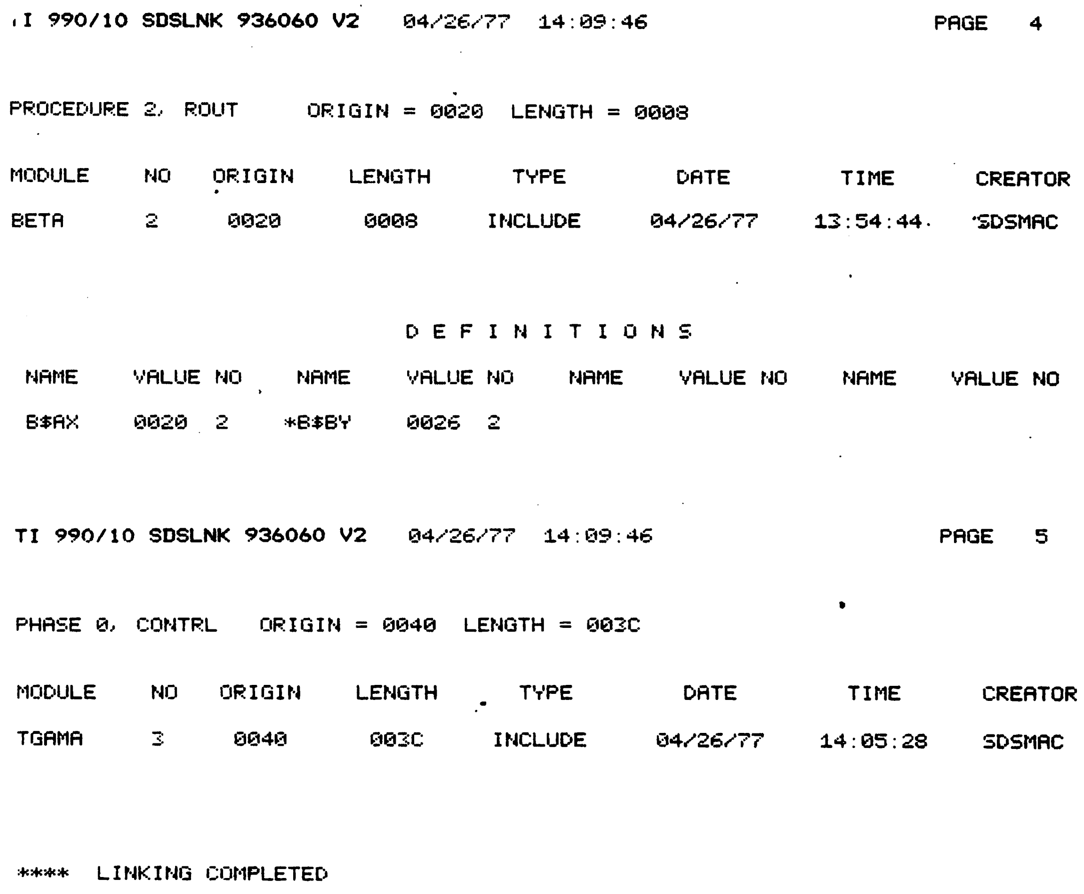
EXECUTE LINKAGE EDITOR

CONTROL ACCESS NAME:	.LEE.EXC9(TWOP)
LINKED OUTPUT ACCESS NAME:	DUMY
LISTING ACCESS NAME:	.LEE.LST
PRINT WIDTH:	80



```
TI 990/10 SDSLNK 936060 V2    04/26/77  14:09:46                 PAGE   1
COMMAND LIST

LIBRARY      .LEE.EXO
PROCEDURE   TABLE
INCL    .ALPHA
PROC    ROUT
INCLUDE .BETA
TASK    CONTRL
INCL    .TGAMA
END


TI 990/10 SDSLNK 936060 V2    04/26/77  14:09:46                 PAGE   2
LINK MAP

CONTROL FILE = .LEE.EXC.TWOP

LINKED OUTPUT FILE = DUMY

LIST FILE = . LEE. LST

OUTPUT FORMAT = ASCII


TI 990/10 SDSLNK 936060 V2    04/26/77  14:09:46                 PAGE   3

PROCEDURE 1, TABLE     ORIGIN = 0000  LENGTH = 0008

MODULE     NO    ORIGIN     LENGTH       TYPE        DATE        TIME      CREATOR

ALPHA      1      0000       0008      INCLUDE     04/26/77    13:52:07    SDSMAC


                                 D E F I N I T I O N S

 NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 M$A       0000  1      M$B       0002  1      M$C       0004  1      M$D       0006  1
```

Figure 4-3. Task With Two Attached Procedures (Sheet 1 of 2)



```
TI 990/10 SDSLNK 936060 V2   04/26/77  14:09:46                        PAGE   4

PROCEDURE 2, ROUT      ORIGIN = 0020  LENGTH = 0008

MODULE      NO   ORIGIN     LENGTH      TYPE         DATE         TIME       CREATOR

BETA        2      0020       0008     INCLUDE     04/26/77     13:54:44    SDSMAC


                                    D E F I N I T I O N S

  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  B$AX     0020  2     *B$BY     0026  2


TI 990/10 SDSLNK 936060 V2   04/26/77  14:09:46                        PAGE   5

PHASE 0, CONTRL    ORIGIN = 0040  LENGTH = 003C

MODULE      NO   ORIGIN     LENGTH      TYPE         DATE         TIME       CREATOR

TGAMA       3      0040       003C     INCLUDE     04/26/77     14:05:28    SDSMAC



****  LINKING COMPLETED
```

Figure 4-3. Task With Two Attached Procedures (Sheet 2 of 2)

Note that within the Control Stream, the procedures are defined before the task. On the Link Map, the procedures are also presented first. Page three of the example contains the Link Map for PROCEDURE 1, TABLE, which has an origin at relative address 0000 and a length of eight bytes. One module, ALPHA, is included in TABLE and it is taken from random library .LEE.EXO.

PROCEDURE 2, ROUT, is shown in the Link Map on page four of the example. ROUT consists of one module, BETA, has a relative origin of 0020_{16}, and a length of eight bytes. BETA is specified by an INCLUDE command and is read from the random library .LEE.EXO. Note that BETA contains one external definition, B$BY, that is not referenced.

PHASE 0, shown on page five of the example, is defined by the TASK command and is named CONTRL. CONTRL consists of one module, TGAMA, specified by the INCLUDE command and read from the random library. CONTRL has an origin at relative address 0040_{16} and a length of $3C_{16}$ bytes. CONTRL contains no external definitions.

The output format of the Link Edit is ASCII. The two procedures have to be installed in the DX10 system by the Install Procedure SCI command before the task is installed by use of the Install Task SCI command.

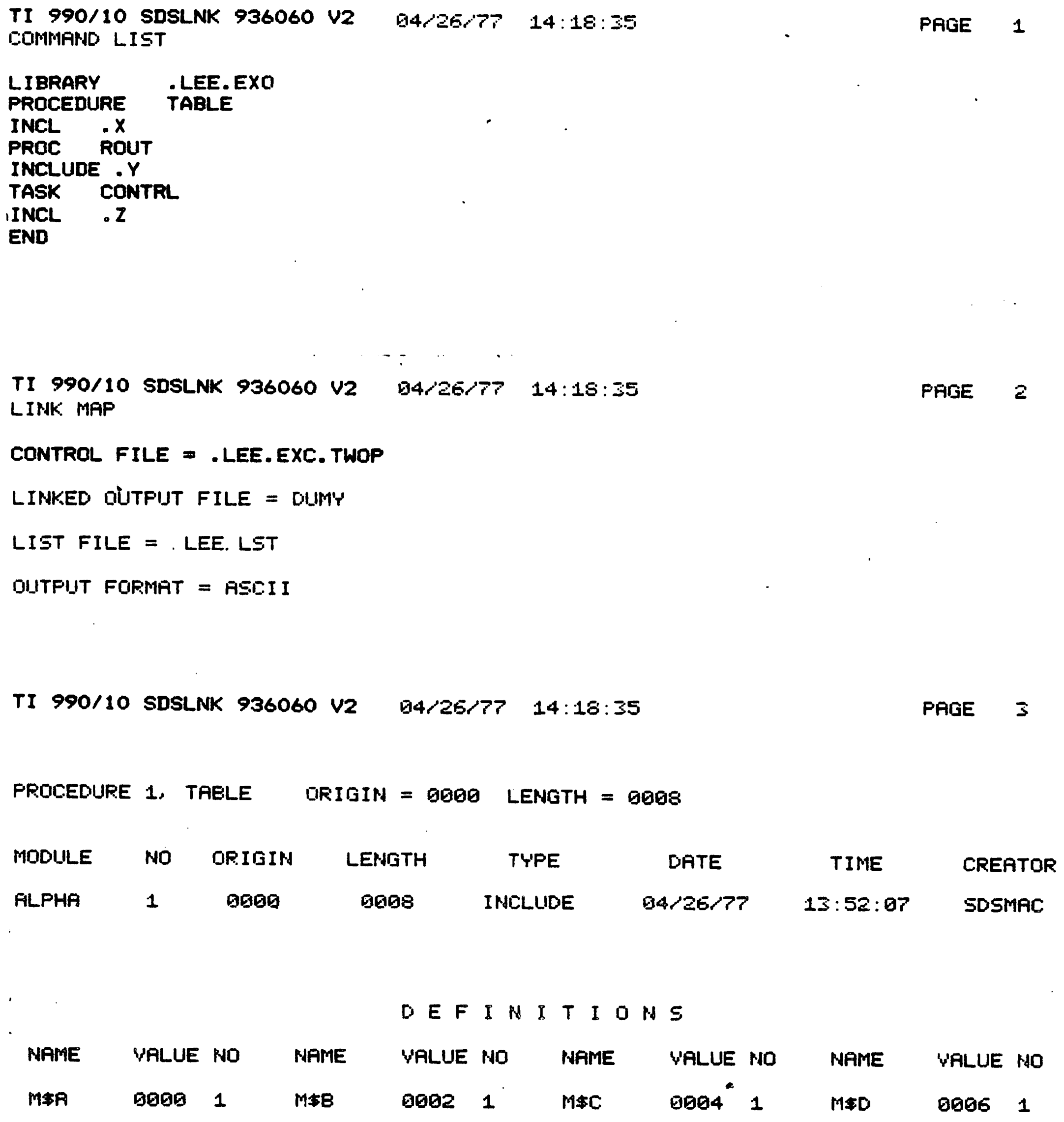
4.3.3 TWO PROCEDURE EXAMPLE. The example shown in figure 4-4 performs the same link as that shown in figure 4-3. The difference in the two occurs in the Control Stream. Figure 4-3 uses INCLUDE commands that specify ALPHA, ROUT, and TGAMA, whereas figure 4-4 uses INCLUDE commands that specify X, Y, and Z. However, note that the link maps (pages 3, 4, and 5 of figure 4-4) show that the modules ALPHA, ROUT, and TGAMA are included in the linked output.

The purpose of this example is to show that the names specified by the INCLUDE commands do not have to be the same as the module name. In this example, module ALPHA is a File X, which is a part of the random library specified by LIBRARY .LEE.EXO. In other words, File X consists of one module named ALPHA. Similarly, file Y contains only module ROUT, and file Z contains only module TGAMA. The file names, not the module names, are specified.

4.3.4 OVERLAY LINK EDIT EXAMPLE — DX10. The listing shown in figure 4-5 illustrates the Control Stream, Link Map, parameters and structure required to produce an overlay structured program. Automatic Overlay Loading is not used in the example.

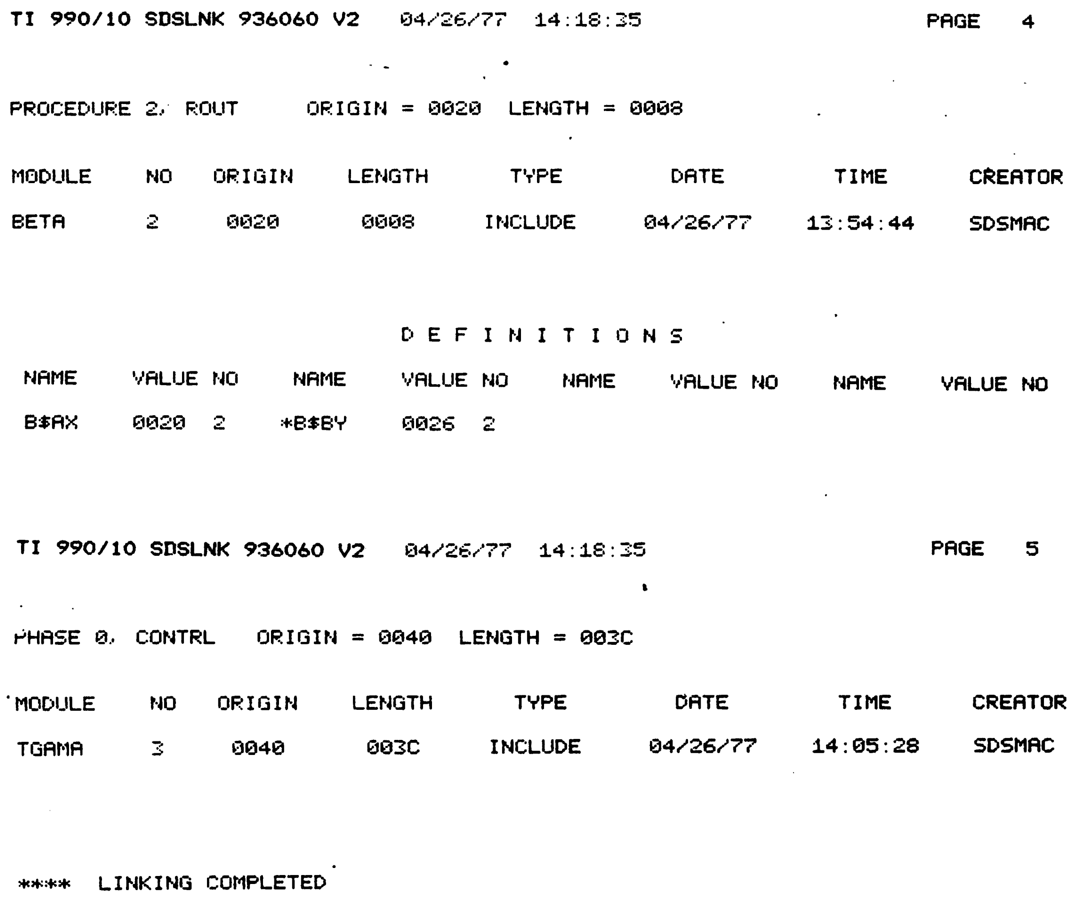
The overlayed program consists of seven modules and three levels. The procedure, XS$AM, begins at relative location 0000. Phase 0, T$CAL, is specified by the TASK command, consists of one module, ROOT, and begins at relative address 0040_{16}. P$ONE and T$CAL are always memory resident when the task is active.

The program uses four overlays, two at level one and two at level two. The level one phases are O$ONEA, which consists of the module MOD1, and O$ONEB, which consists of modules MOD4 and MODDAT. Both phases begin at relative locations 0090_{16}. Note that only one of these phases can be in memory at one time, and they are mutually exclusive.



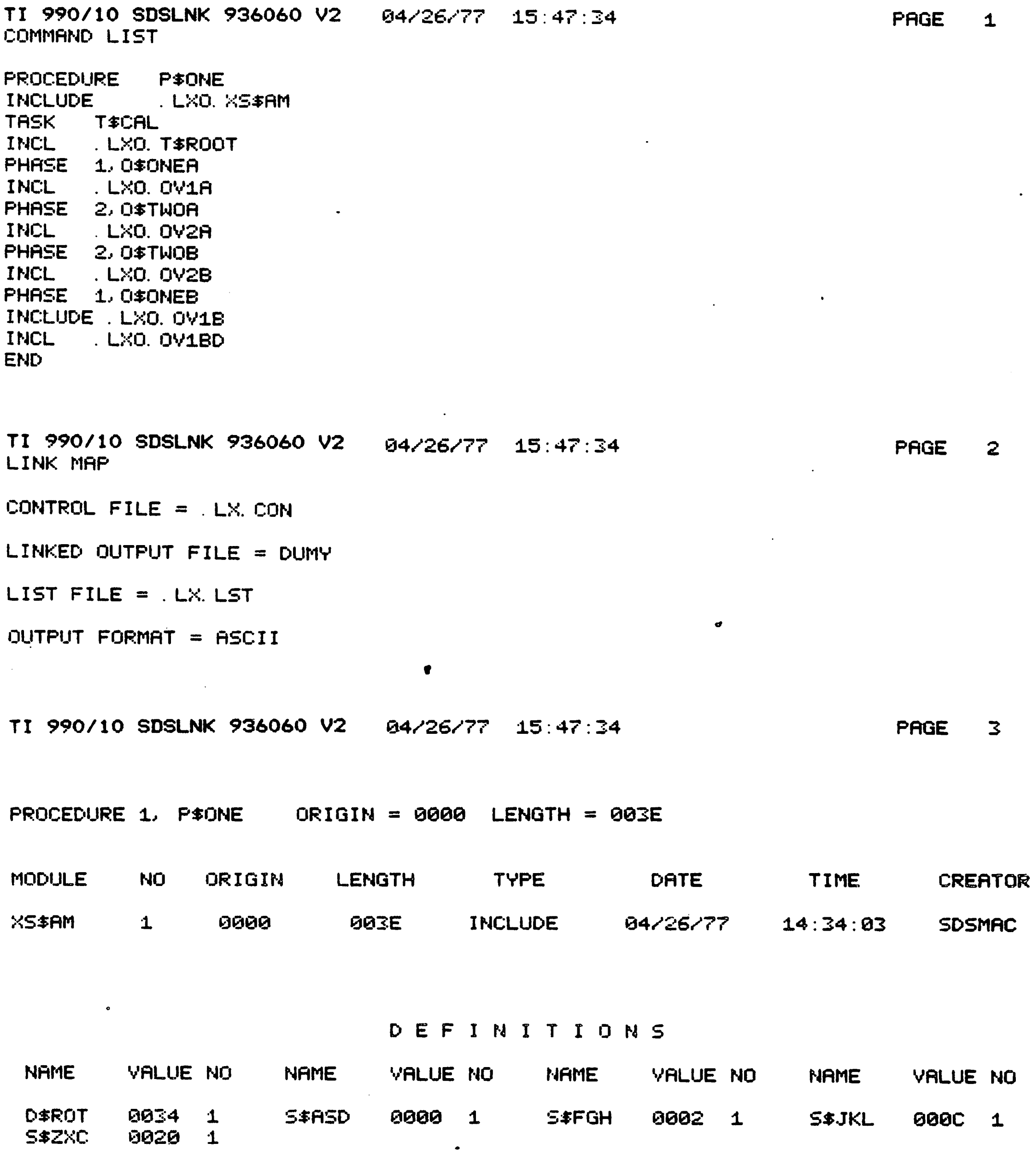
```
TI 990/10 SDSLNK 936060 V2    04/26/77  14:18:35                          PAGE   1
COMMAND LIST

LIBRARY      .LEE.EXO
PROCEDURE    TABLE
INCL    .X
PROC    ROUT
INCLUDE .Y
TASK    CONTRL
INCL    .Z
END


TI 990/10 SDSLNK 936060 V2    04/26/77  14:18:35                          PAGE   2
LINK MAP

CONTROL FILE = .LEE.EXC.TWOP

LINKED OUTPUT FILE = DUMY

LIST FILE = . LEE. LST

OUTPUT FORMAT = ASCII


TI 990/10 SDSLNK 936060 V2    04/26/77  14:18:35                          PAGE   3


PROCEDURE 1, TABLE    ORIGIN = 0000  LENGTH = 0008

MODULE     NO   ORIGIN    LENGTH     TYPE      DATE       TIME      CREATOR

ALPHA      1     0000      0008     INCLUDE   04/26/77   13:52:07   SDSMAC


                              D E F I N I T I O N S

 NAME    VALUE NO    NAME    VALUE NO    NAME    VALUE NO    NAME    VALUE NO

 M$A     0000  1     M$B     0002  1     M$C     0004  1     M$D     0006  1
```

Figure 4-4. Two PROC Example — DX10 (Sheet 1 of 2)



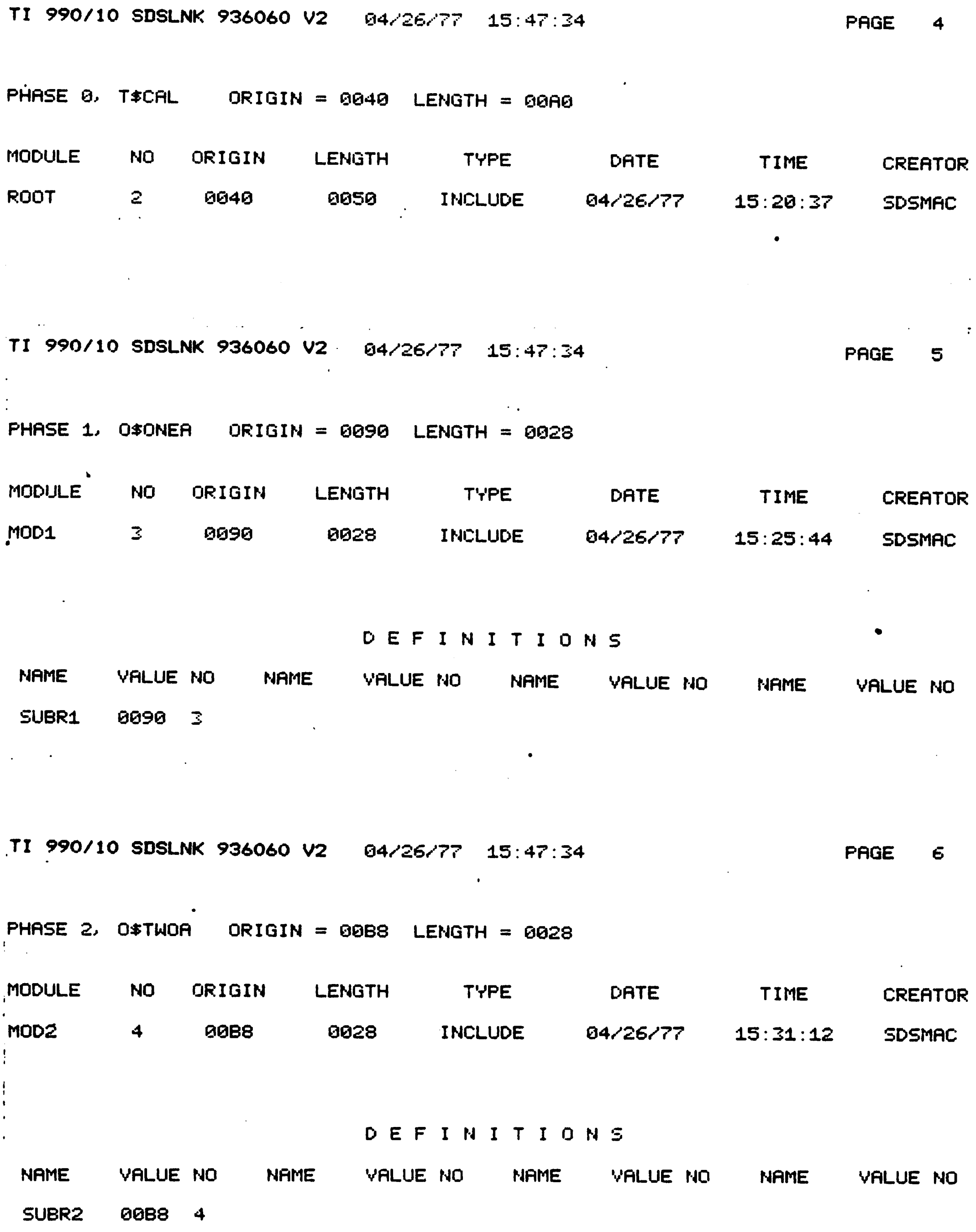
```
TI 990/10 SDSLNK 936060 V2   04/26/77  14:18:35                        PAGE   4

PROCEDURE 2, ROUT     ORIGIN = 0020  LENGTH = 0008

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME      CREATOR

BETA       2     0020      0008     INCLUDE     04/26/77    13:54:44    SDSMAC


                                 D E F I N I T I O N S

 NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 B$AX      0020  2     *B$BY      0026  2


TI 990/10 SDSLNK 936060 V2   04/26/77  14:18:35                        PAGE   5

PHASE 0, CONTRL   ORIGIN = 0040  LENGTH = 003C

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME      CREATOR

TGAMA      3     0040      003C     INCLUDE     04/26/77    14:05:28    SDSMAC


****  LINKING COMPLETED
```

Figure 4-4. Two PROC Example — DX10 (Sheet 2 of 2)



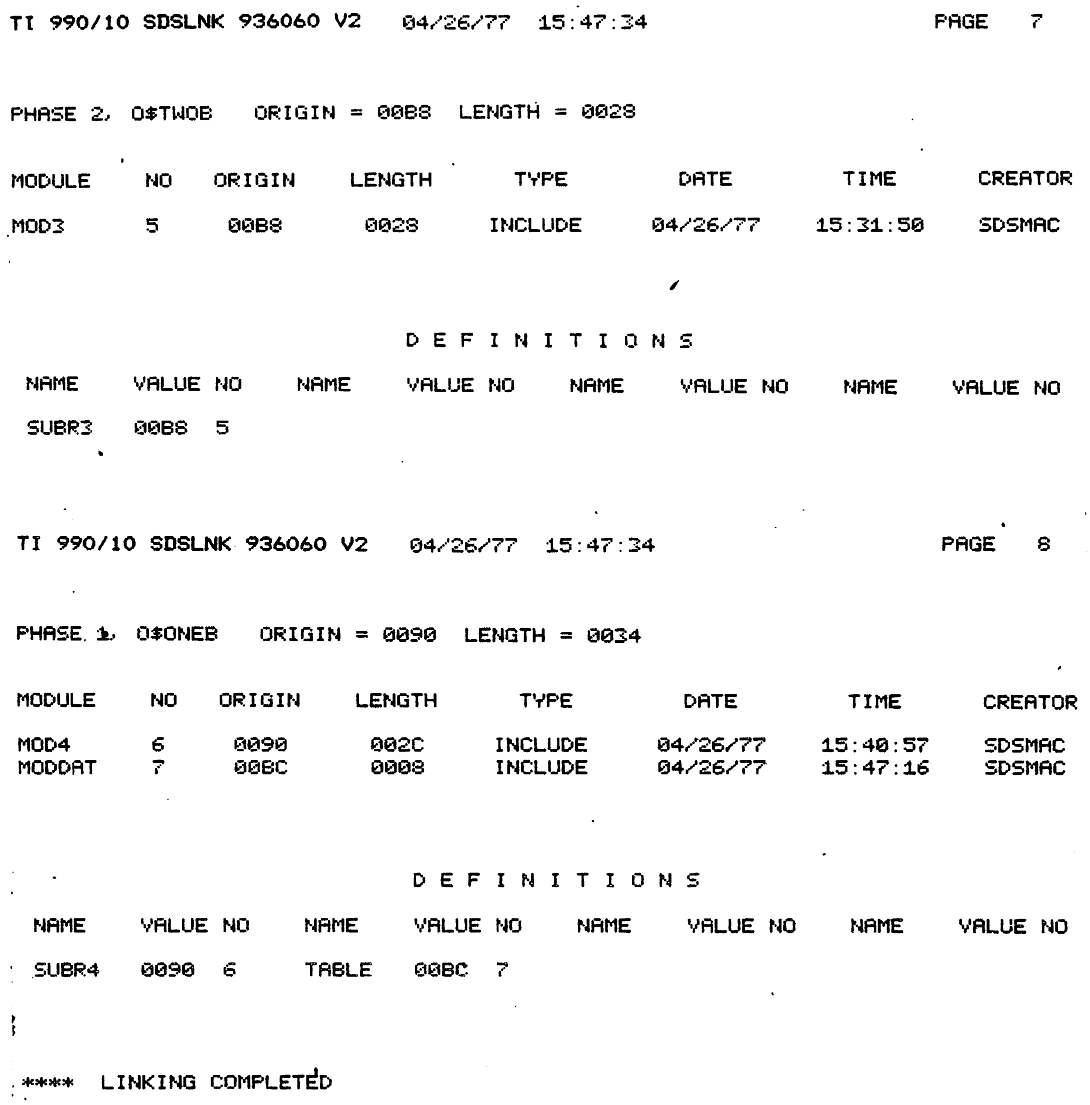
```
TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                         PAGE   1
COMMAND LIST

PROCEDURE    P$ONE
INCLUDE      .LXO.XS$AM
TASK   T$CAL
INCL    .LXO.T$ROOT
PHASE  1,O$ONEA
INCL    .LXO.OV1A
PHASE  2,O$TWOA
INCL    .LXO.OV2A
PHASE  2,O$TWOB
INCL    .LXO.OV2B
PHASE  1,O$ONEB
INCLUDE .LXO.OV1B
INCL    .LXO.OV1BD
END


TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                         PAGE   2
LINK MAP

CONTROL FILE = .LX.CON

LINKED OUTPUT FILE = DUMY

LIST FILE = .LX.LST

OUTPUT FORMAT = ASCII


TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                         PAGE   3


PROCEDURE 1, P$ONE    ORIGIN = 0000  LENGTH = 003E

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME       CREATOR

XS$AM      1     0000      003E     INCLUDE     04/26/77    14:34:03     SDSMAC


                                D E F I N I T I O N S

  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  D$ROT    0034  1      S$ASD    0000  1      S$FGH    0002  1      S$JKL    000C  1
  S$ZXC    0020  1
```

Figure 4-5. Overlayed Program Example (Sheet 1 of 3)



```
TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                    PAGE   4

PHASE 0, T$CAL     ORIGIN = 0040  LENGTH = 00A0

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME      CREATOR
ROOT       2     0040      0050     INCLUDE     04/26/77    15:20:37    SDSMAC


TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                    PAGE   5

PHASE 1, O$ONEA    ORIGIN = 0090  LENGTH = 0028

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME      CREATOR
MOD1       3     0090      0028     INCLUDE     04/26/77    15:25:44    SDSMAC

                              D E F I N I T I O N S
 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO
 SUBR1    0090  3


TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                    PAGE   6

PHASE 2, O$TWOA    ORIGIN = 00B8  LENGTH = 0028

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE        TIME      CREATOR
MOD2       4     00B8      0028     INCLUDE     04/26/77    15:31:12    SDSMAC

                              D E F I N I T I O N S
 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO
 SUBR2    00B8  4
```

Figure 4-5. Overlayed Program Example (Sheet 2 of 3)



```
TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                      PAGE   7

PHASE 2, O$TWOB   ORIGIN = 00B8  LENGTH = 0028

MODULE     NO   ORIGIN    LENGTH     TYPE        DATE       TIME      CREATOR

MOD3       5     00B8      0028     INCLUDE    04/26/77   15:31:50    SDSMAC


                                 D E F I N I T I O N S
  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  SUBR3    00B8  5


TI 990/10 SDSLNK 936060 V2   04/26/77  15:47:34                      PAGE   8

PHASE 1, O$ONEB   ORIGIN = 0090  LENGTH = 0034

MODULE     NO   ORIGIN    LENGTH     TYPE        DATE       TIME      CREATOR

MOD4       6     0090      002C     INCLUDE    04/26/77   15:40:57    SDSMAC
MODDAT     7     00BC      0008     INCLUDE    04/26/77   15:47:16    SDSMAC


                                 D E F I N I T I O N S
  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  SUBR4    0090  6      TABLE    00BC  7


****  LINKING COMPLETED
```

Figure 4-5. Overlayed Program Example (Sheet 3 of 3)

The level two phases are O$TWOA, which consists of module MOD2, and O$TWOB, which consists of module MOD3. Both phases begin at relative address $00B8_{16}$ and the two phases are exclusive. Note that when either phase O$TWOA or O$TWOB is called, phase O$ONEA must be in memory.

The total memory requirement for this program is the total of the requirements for PONE, TCAL and the longest overlay path, which is O$ONEA and either O$TWOA or O$TWOB (these two phases are the same length). Therefore, the total memory required is 224 bytes. If overlays were not used and the entire program was in memory at one time, the memory requirement would be 314 bytes.

The user should note that the LENGTH specified for PHASE 0 is the length of the longest overlay path. In this example, the longest path consists of TCAL, OONEA, and either O$TWOA or O$TWOB.

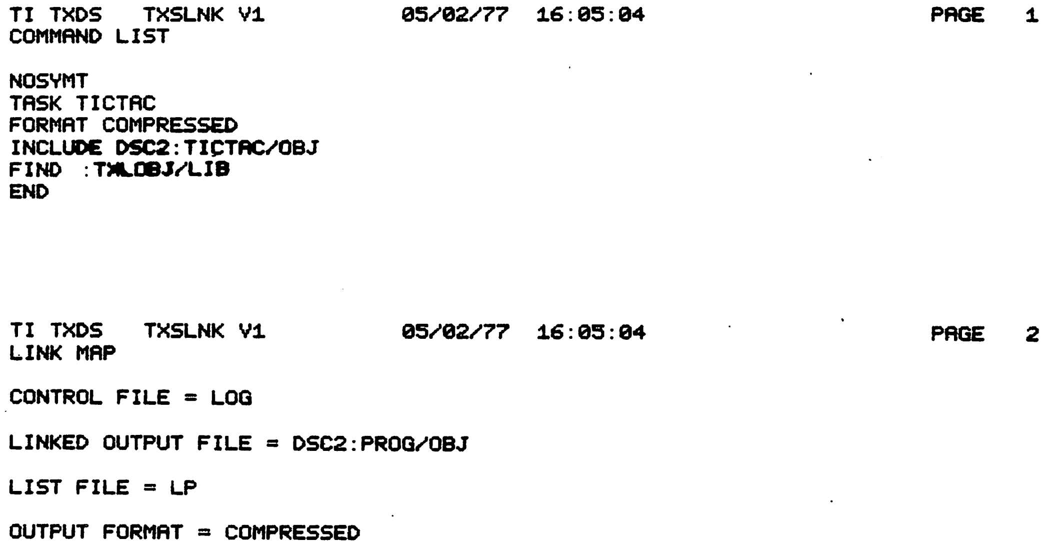
~~The example in figure 4-6 illustrates the use of the Link Editor to link a task named TICTAC. The INCLUDE command specifies that the file with the pathname DSC2:TICTAC/OBJ is to be included. OBJ indicates that it is an object module file. One FIND command is used to specify the sequential library :TXLOBJ/LIB which is the TXDS FORTRAN runtime library.~~

4.4 TXDS EXAMPLES

The following paragraphs contain examples of link edits on a TXDS system. Provided for each example is the complete link map, as described in paragraph 4.2.

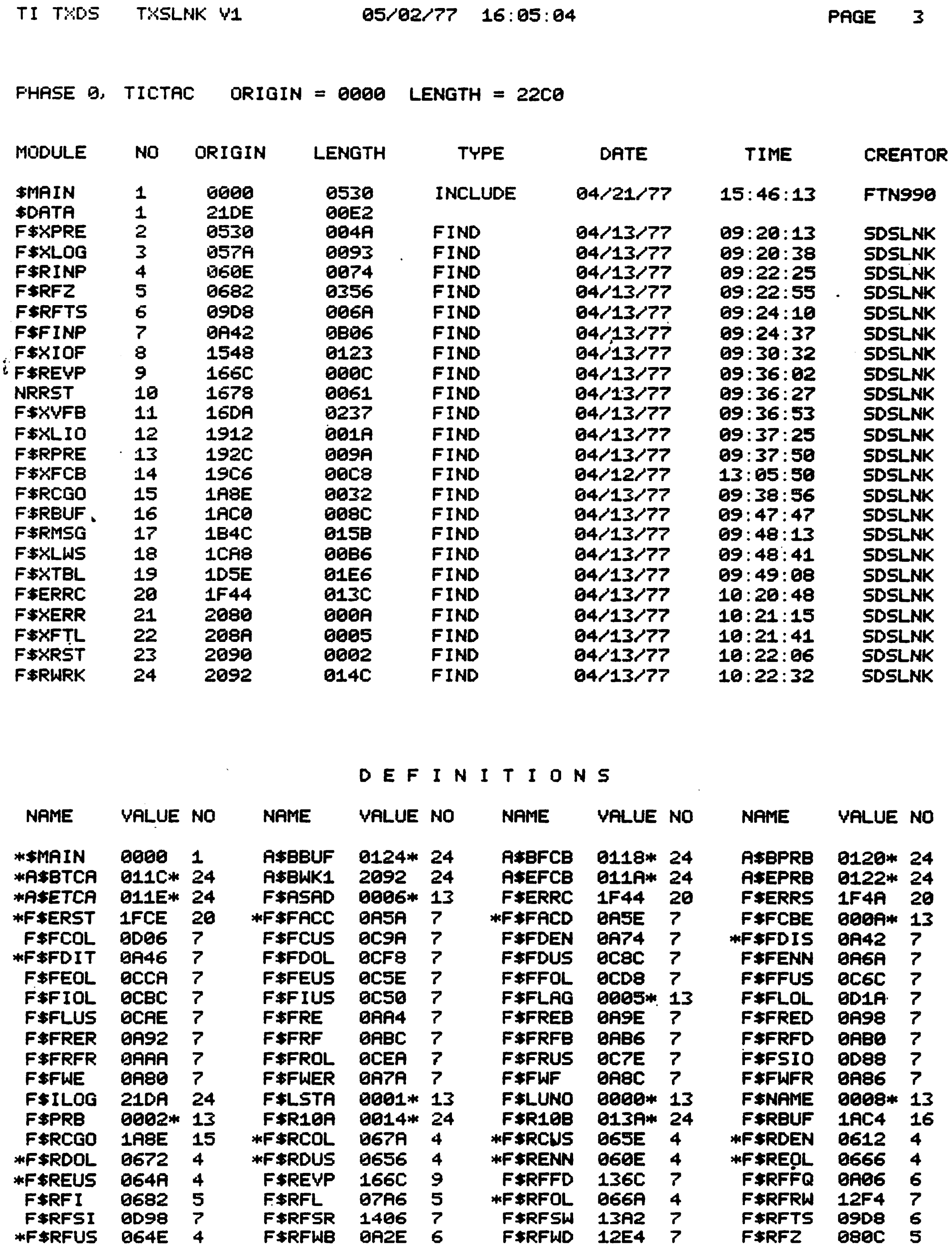
4.4.1 SINGLE TASK, NO PROCEDURE EXAMPLE. The example in figure 4-6 illustrates the use of the Link Editor to link a task named TICTAC. The INCLUDE command specifies that the file with the pathname DSC2:TICTAC/OBJ is to be included. OBJ indicates that it is an object module file. One FIND command is used to specify the sequential library :TXLOBJ/LIB which is the TXDS FORTRAN runtime library.

4.4.2 SINGLE TASK WITH OVERLAYS EXAMPLE. The example in figure 4-7 illustrates the use of the Link Editor to link a task named LOVMTST. A FIND command is used in every PHASE so all FORTRAN references can be resolved. The FIND is needed for automatic symbol resolution within a PHASE. The FIND can be removed from phases D, E, F, G, H, and I since no FIND modules are included in them.



```
TI TXDS   TXSLNK V1          05/02/77  16:05:04                     PAGE   1
COMMAND LIST

NOSYMT
TASK TICTAC
FORMAT COMPRESSED
INCLUDE DSC2:TICTAC/OBJ
FIND  :TXLOBJ/LIB
END



TI TXDS   TXSLNK V1          05/02/77  16:05:04                     PAGE   2
LINK MAP

CONTROL FILE = LOG

LINKED OUTPUT FILE = DSC2:PROG/OBJ

LIST FILE = LP

OUTPUT FORMAT = COMPRESSED
```

Figure 4-6. TXDS Example (Sheet 1 of 3)

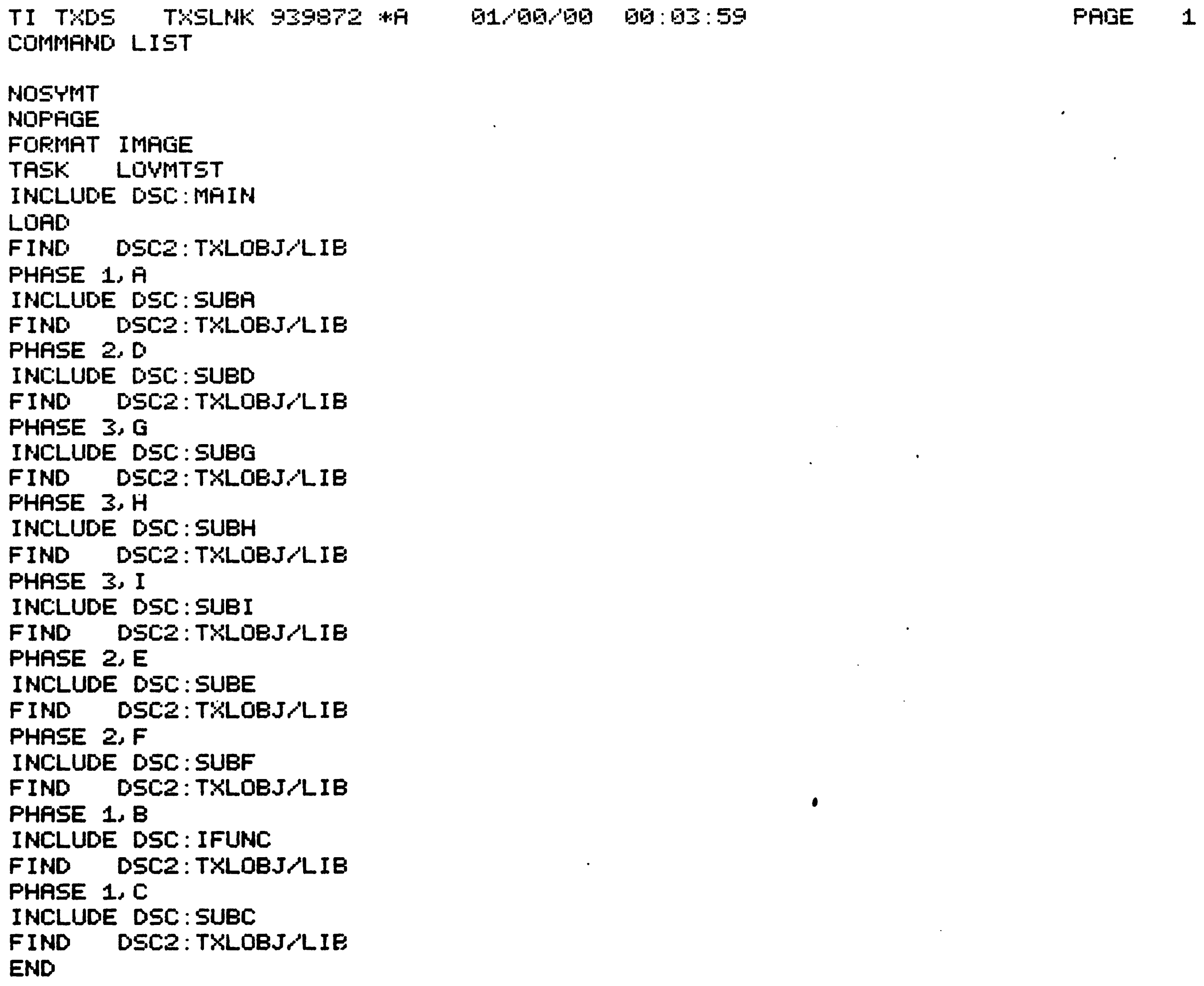


```
TI TXDS   TXSLNK V1         05/02/77  16:05:04                   PAGE   3


FHASE 0, TICTAC   ORIGIN = 0000  LENGTH = 22C0

MODULE     NO   ORIGIN    LENGTH     TYPE        DATE       TIME       CREATOR

$MAIN      1     0000      0530    INCLUDE     04/21/77    15:46:13    FTN990
$DATA      1     21DE      00E2
F$XPRE     2     0530      004A    FIND        04/13/77    09:20:13    SDSLNK
F$XLOG     3     057A      0093    FIND        04/13/77    09:20:38    SDSLNK
F$RINP     4     060E      0074    FIND        04/13/77    09:22:25    SDSLNK
F$RFZ      5     0682      0356    FIND        04/13/77    09:22:55    SDSLNK
F$RFTS     6     09D8      006A    FIND        04/13/77    09:24:10    SDSLNK
F$FINP     7     0A42      0B06    FIND        04/13/77    09:24:37    SDSLNK
F$XIOF     8     1548      0123    FIND        04/13/77    09:30:32    SDSLNK
F$REVP     9     166C      000C    FIND        04/13/77    09:36:02    SDSLNK
NRRST      10    1678      0061    FIND        04/13/77    09:36:27    SDSLNK
F$XVFB     11    16DA      0237    FIND        04/13/77    09:36:53    SDSLNK
F$XLIO     12    1912      001A    FIND        04/13/77    09:37:25    SDSLNK
F$RPRE     13    192C      009A    FIND        04/13/77    09:37:50    SDSLNK
F$XFCB     14    19C6      00C8    FIND        04/12/77    13:05:50    SDSLNK
F$RCGO     15    1A8E      0032    FIND        04/13/77    09:38:56    SDSLNK
F$RBUF     16    1AC0      008C    FIND        04/13/77    09:47:47    SDSLNK
F$RMSG     17    1B4C      015B    FIND        04/13/77    09:48:13    SDSLNK
F$XLWS     18    1CA8      00B6    FIND        04/13/77    09:48:41    SDSLNK
F$XTBL     19    1D5E      01E6    FIND        04/13/77    09:49:08    SDSLNK
F$ERRC     20    1F44      013C    FIND        04/13/77    10:20:48    SDSLNK
F$XERR     21    2080      000A    FIND        04/13/77    10:21:15    SDSLNK
F$XFTL     22    208A      0005    FIND        04/13/77    10:21:41    SDSLNK
F$XRST     23    2090      0002    FIND        04/13/77    10:22:06    SDSLNK
F$RWRK     24    2092      014C    FIND        04/13/77    10:22:32    SDSLNK


                              D E F I N I T I O N S

 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

*$MAIN    0000  1      A$BBUF   0124* 24     A$BFCB   0118* 24     A$BPRB   0120* 24
*A$BTCA   011C* 24     A$BWK1   2092  24     A$EFCB   011A* 24     A$EPRB   0122* 24
*A$ETCA   011E* 24     F$ASAD   0006* 13     F$ERRC   1F44  20     F$ERRS   1F4A  20
*F$ERST   1FCE  20    *F$FACC   0A5A  7     *F$FACD   0A5E  7      F$FCBE   000A* 13
 F$FCOL   0D06  7      F$FCUS   0C9A  7      F$FDEN   0A74  7     *F$FDIS   0A42  7
*F$FDIT   0A46  7      F$FDOL   0CF8  7      F$FDUS   0C8C  7      F$FENN   0A6A  7
 F$FEOL   0CCA  7      F$FEUS   0C5E  7      F$FFOL   0CD8  7      F$FFUS   0C6C  7
 F$FIOL   0CBC  7      F$FIUS   0C50  7      F$FLAG   0005* 13     F$FLOL   0D1A  7
 F$FLUS   0CAE  7      F$FRE    0AA4  7      F$FREB   0A9E  7      F$FRED   0A98  7
 F$FRER   0A92  7      F$FRF    0ABC  7      F$FRFB   0AB6  7      F$FRFD   0AB0  7
 F$FRFR   0AAA  7      F$FROL   0CEA  7      F$FRUS   0C7E  7      F$FSIO   0D88  7
 F$FWE    0A80  7      F$FWER   0A7A  7      F$FWF    0A8C  7      F$FWFR   0A86  7
 F$ILOG   21DA  24     F$LSTA   0001* 13     F$LUNO   0000* 13     F$NAME   0008* 13
 F$PRB    0002* 13     F$R10A   0014* 24     F$R10B   013A* 24     F$RBUF   1AC4  16
 F$RCGO   1A8E  15    *F$RCOL   067A  4     *F$RCUS   065E  4     *F$RDEN   0612  4
*F$RDOL   0672  4     *F$RDUS   0656  4     *F$RENN   060E  4     *F$REOL   0666  4
*F$REUS   064A  4      F$REVP   166C  9      F$RFFD   136C  7      F$RFFQ   0A06  6
 F$RFI    0682  5      F$RFL    07A6  5     *F$RFOL   066A  4      F$RFRW   12F4  7
 F$RFSI   0D98  7      F$RFSR   1406  7      F$RFSW   13A2  7      F$RFTS   09D8  6
*F$RFUS   064E  4      F$RFWB   0A2E  6      F$RFWD   12E4  7      F$RFZ    080C  5
```

Figure 4-6. TXDS Example (Sheet 2 of 3)

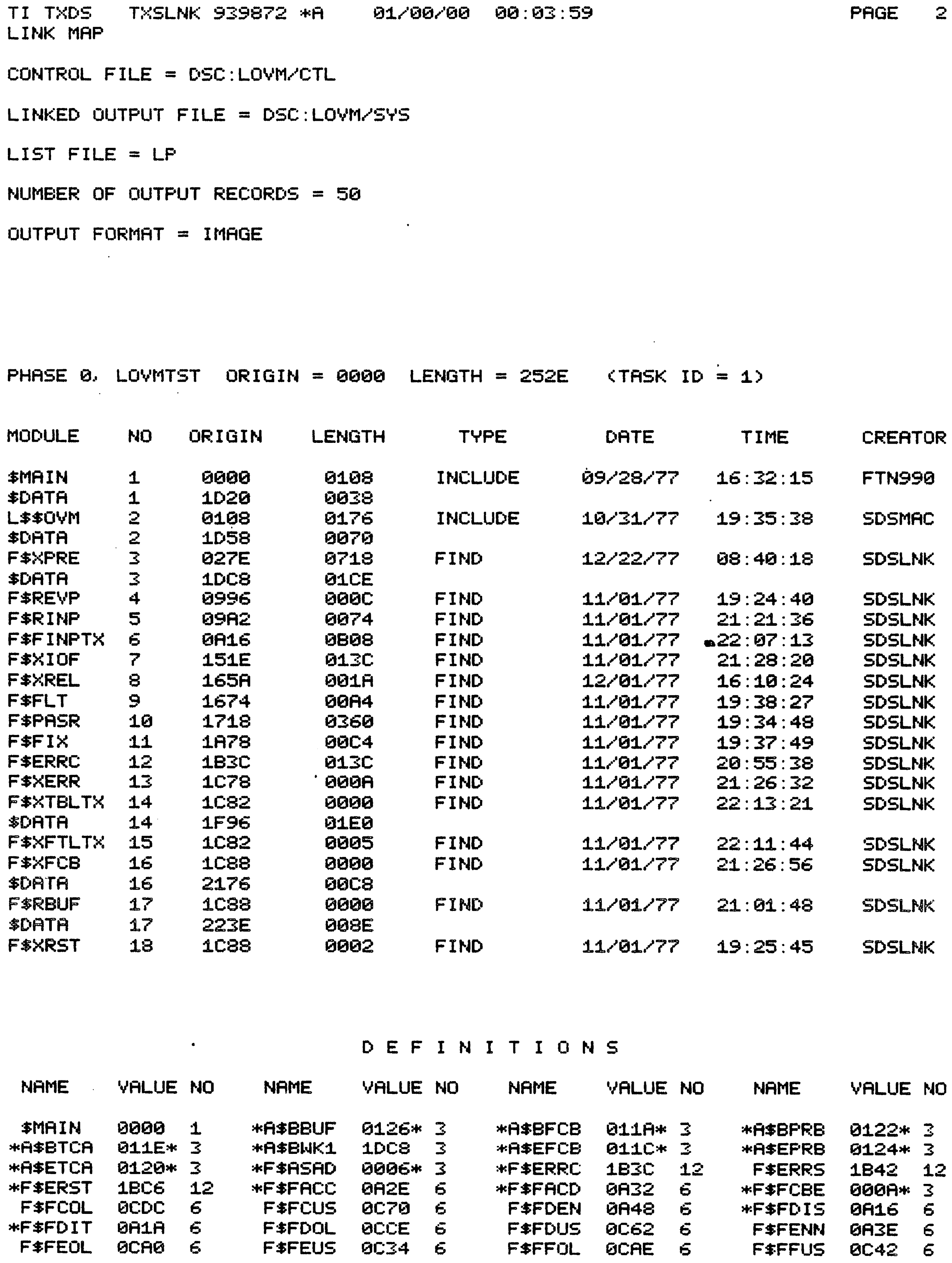
```
TI TXDS    TXSLNK V1               05/02/77   16:05:04                            PAGE    4
 NAME      VALUE NO     NAME       VALUE NO     NAME       VALUE NO     NAME       VALUE NO

 F$RIOL    0662   4      F$RIUS    0646   4      F$RLOG    0146*  24   *F$RLOL     0676   4
 F$RLP2    0148*  24    *F$RLUS    065A   4      F$RMSZ    00B4*  18    F$RPRE     192C   13
 F$RPRM    01E0*  19    *F$RRE     0632   4     *F$RREB    062E   4    *F$RRED     062A   4
*F$RRER    0626   4     *F$RRF     0642   4     *F$RRFB    063E   4    *F$RRFD     063A   4
 F$RRFR    0636   4     *F$RROL    066E   4     *F$RRUS    0652   4     F$RSIO     067E   4
 F$RTFG    20BA   24     F$RVFB    0028*  24     F$RVP2    002A*  24   *F$RWE      061A   4
*F$RWER    0616   4      F$RWF     0622   4     *F$RWFR    061E   4     F$RWRK     2094   24
 F$STAT    0004*  13    *F$XASN    0020*  18     F$XBCH    006C*  18   *F$XBCS     010A*  24
 F$XBFS    0088*  16    *F$XBUI    1618   8     *F$XBUO    1604   8    *F$XCAL     1F43   19
 F$XCPX    006E*  18     F$XCPY    0070%  18    *F$XCRR    0021*  18   *F$XEOF     15FA   8
 F$XERR    2080   21     F$XFCB    19C6   14     F$XFCE    1A8E   14   *F$XFND     1600   8
*F$XFOP    0022*  18     F$XFTL    208A   22     F$XLIO    1912   12    F$XLOG     057A   3
 F$XLWS    1CA8   18     F$XMES    0073*  18    *F$XOPN    0032*  18    F$XPER     0060*  18
 F$XPRE    0530   2     *F$XPSE    19B4   13     F$XRED    1548   8     F$XRST     2090   23
*F$XRWD    15E2   8     *F$XSA1    0111*  24    *F$XSA2    0112*  24   *F$XSA3     0113*  24
*F$XSA4    0114*  24    *F$XSA5    0115*  24    *F$XSA6    0116*  24   *F$XSA7     0117*  24
*F$XSER    003E*  18    *F$XSTC    010C*  24    *F$XSTL    010E*  24   *F$XSTP     19BC   13
*F$XSVC    0110*  24     F$XTBE    1F3E   19     F$XTBL    1D5E   19   *F$XTID     1F41   19
*F$XTRA    15DA   8     *F$XTRM    002C*  18    *F$XVBF    0100*  24   *F$XVCC     00FB*  24
*F$XVCH    0104*  24    *F$XVCL    00FF*  24    *F$XVCO    00FC*  24    F$XVFB     16DE   11
*F$XVRC    0102*  24    *F$XVRO    00FE*  24    *F$XVST    00FD*  24   *F$XVSV     00FA*  24
 F$XVWS    21BA   24     F$XWRT    1566   8      G$XE01    1B4C   17   *G$XE02     1B5F   17
*G$XE03    1B74   17    *G$XE04    1B86   17    *G$XE05    1B97   17   *G$XE06     1BA8   17
 G$XE08    1BBA   17     G$XE09    1BDC   17    *G$XE10    1C09   17    G$XE11     1C24   17
 G$XE12    1C3F   17     G$XE13    1C57   17     G$XE14    1C95   17    N$COLS     0106*  24
 N$LINS    0108*  24     NERRST    1678   10     P$ABUF    0006*  13    P$CCNT     000A*  13
 P$ERR     0001*  13     P$LACN    0016*  13    *P$LFIL    0011*  13    P$LIBF     0010*  13
 P$LLRL    0012*  13     P$LPRL    0014*  13     P$LUN     0003*  13    P$OP       0002*  13
 P$PFCB    0018*  13    *P$PRB     0000*  13     P$PRBE    001A*  13   *P$REC1     000D*  13
*P$REC2    000E*  13     P$RECL    0008*  13    *P$RES     000C*  13   *P$SFLG     0004*  13
*P$SVC0    0000*  13     P$UFLG    0005*  13    *S$APRB    183C   11   *S$OPEN     187A   11


****  LINKING COMPLETED
```

Figure 4-6. TXDS Example (Sheet 3 of 3)



```
TI TXDS    TXSLNK 939872 *A     01/00/00  00:03:59                        PAGE    1
COMMAND LIST

NOSYMT
NOPAGE
FORMAT IMAGE
TASK    LOVMTST
INCLUDE DSC:MAIN
LOAD
FIND    DSC2:TXLOBJ/LIB
PHASE 1,A
INCLUDE DSC:SUBA
FIND    DSC2:TXLOBJ/LIB
PHASE 2,D
INCLUDE DSC:SUBD
FIND    DSC2:TXLOBJ/LIB
PHASE 3,G
INCLUDE DSC:SUBG
FIND    DSC2:TXLOBJ/LIB
PHASE 3,H
INCLUDE DSC:SUBH
FIND    DSC2:TXLOBJ/LIB
PHASE 3,I
INCLUDE DSC:SUBI
FIND    DSC2:TXLOBJ/LIB
PHASE 2,E
INCLUDE DSC:SUBE
FIND    DSC2:TXLOBJ/LIB
PHASE 2,F
INCLUDE DSC:SUBF
FIND    DSC2:TXLOBJ/LIB
PHASE 1,B
INCLUDE DSC:IFUNC
FIND    DSC2:TXLOBJ/LIB
PHASE 1,C
INCLUDE DSC:SUBC
FIND    DSC2:TXLOBJ/LIB
END
```

Figure 4-7. TXDS Examples With Overlays (Sheet 1 of 7)

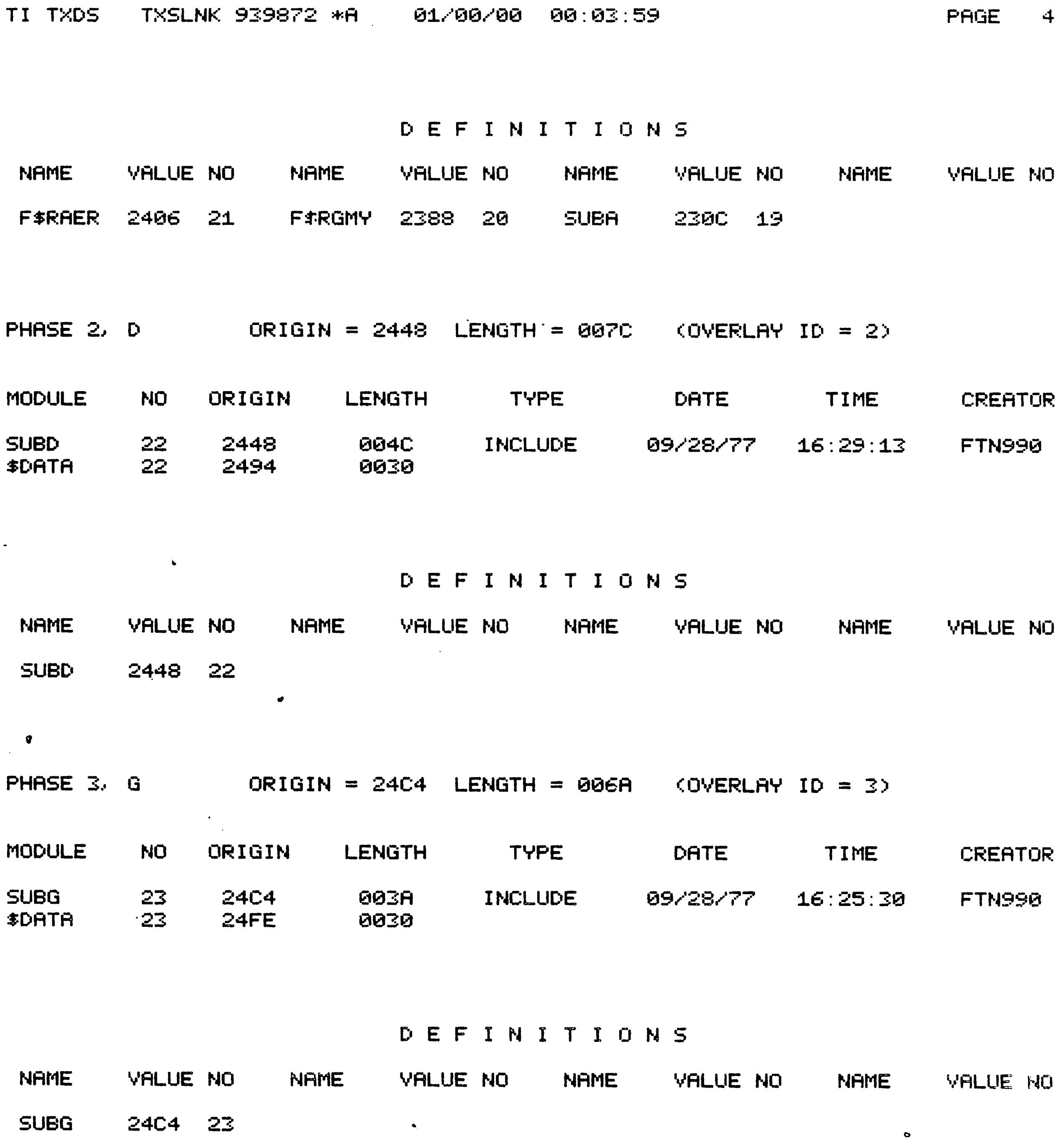


```
TI TXDS   TXSLNK 939872 *A    01/00/00  00:03:59                    PAGE   2
LINK MAP

CONTROL FILE = DSC:LOVM/CTL

LINKED OUTPUT FILE = DSC:LOVM/SYS

LIST FILE = LP

NUMBER OF OUTPUT RECORDS = 50

OUTPUT FORMAT = IMAGE




PHASE 0, LOVMTST  ORIGIN = 0000  LENGTH = 252E   (TASK ID = 1)

MODULE     NO   ORIGIN    LENGTH     TYPE        DATE       TIME      CREATOR

$MAIN      1     0000      0108    INCLUDE    09/28/77   16:32:15   FTN990
$DATA      1     1D20      0038
L$$OVM     2     0108      0176    INCLUDE    10/31/77   19:35:38   SDSMAC
$DATA      2     1D58      0070
F$XPRE     3     027E      0718    FIND       12/22/77   08:40:18   SDSLNK
$DATA      3     1DC8      01CE
F$REVP     4     0996      000C    FIND       11/01/77   19:24:40   SDSLNK
F$RINP     5     09A2      0074    FIND       11/01/77   21:21:36   SDSLNK
F$FINPTX   6     0A16      0B08    FIND       11/01/77   22:07:13   SDSLNK
F$XIOF     7     151E      013C    FIND       11/01/77   21:28:20   SDSLNK
F$XREL     8     165A      001A    FIND       12/01/77   16:10:24   SDSLNK
F$FLT      9     1674      00A4    FIND       11/01/77   19:38:27   SDSLNK
F$PASR     10    1718      0360    FIND       11/01/77   19:34:48   SDSLNK
F$FIX      11    1A78      00C4    FIND       11/01/77   19:37:49   SDSLNK
F$ERRC     12    1B3C      013C    FIND       11/01/77   20:55:38   SDSLNK
F$XERR     13    1C78      000A    FIND       11/01/77   21:26:32   SDSLNK
F$XTBLTX   14    1C82      0000    FIND       11/01/77   22:13:21   SDSLNK
$DATA      14    1F96      01E0
F$XFTLTX   15    1C82      0005    FIND       11/01/77   22:11:44   SDSLNK
F$XFCB     16    1C88      0000    FIND       11/01/77   21:26:56   SDSLNK
$DATA      16    2176      00C8
F$RBUF     17    1C88      0000    FIND       11/01/77   21:01:48   SDSLNK
$DATA      17    223E      008E
F$XRST     18    1C88      0002    FIND       11/01/77   19:25:45   SDSLNK



                           D E F I N I T I O N S

  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  $MAIN    0000  1    *A$BBUF   0126* 3    *A$BFCB   011A* 3    *A$BPRB   0122* 3
*A$BTCA    011E* 3    *A$BWK1   1DC8  3    *A$EFCB   011C* 3    *A$EPRB   0124* 3
*A$ETCA    0120* 3    *F$ASAD   0006* 3    *F$ERRC   1B3C  12    F$ERRS   1B42  12
*F$ERST    1BC6  12   *F$FACC   0A2E  6    *F$FACD   0A32  6    *F$FCBE   000A* 3
 F$FCOL    0CDC  6     F$FCUS   0C70  6     F$FDEN   0A48  6    *F$FDIS   0A16  6
*F$FDIT    0A1A  6     F$FDOL   0CCE  6     F$FDUS   0C62  6     F$FENN   0A3E  6
 F$FEOL    0CA0  6     F$FEUS   0C34  6     F$FFOL   0CAE  6     F$FFUS   0C42  6
```

Figure 4-7. TXDS Examples With Overlays (Sheet 2 of 7)

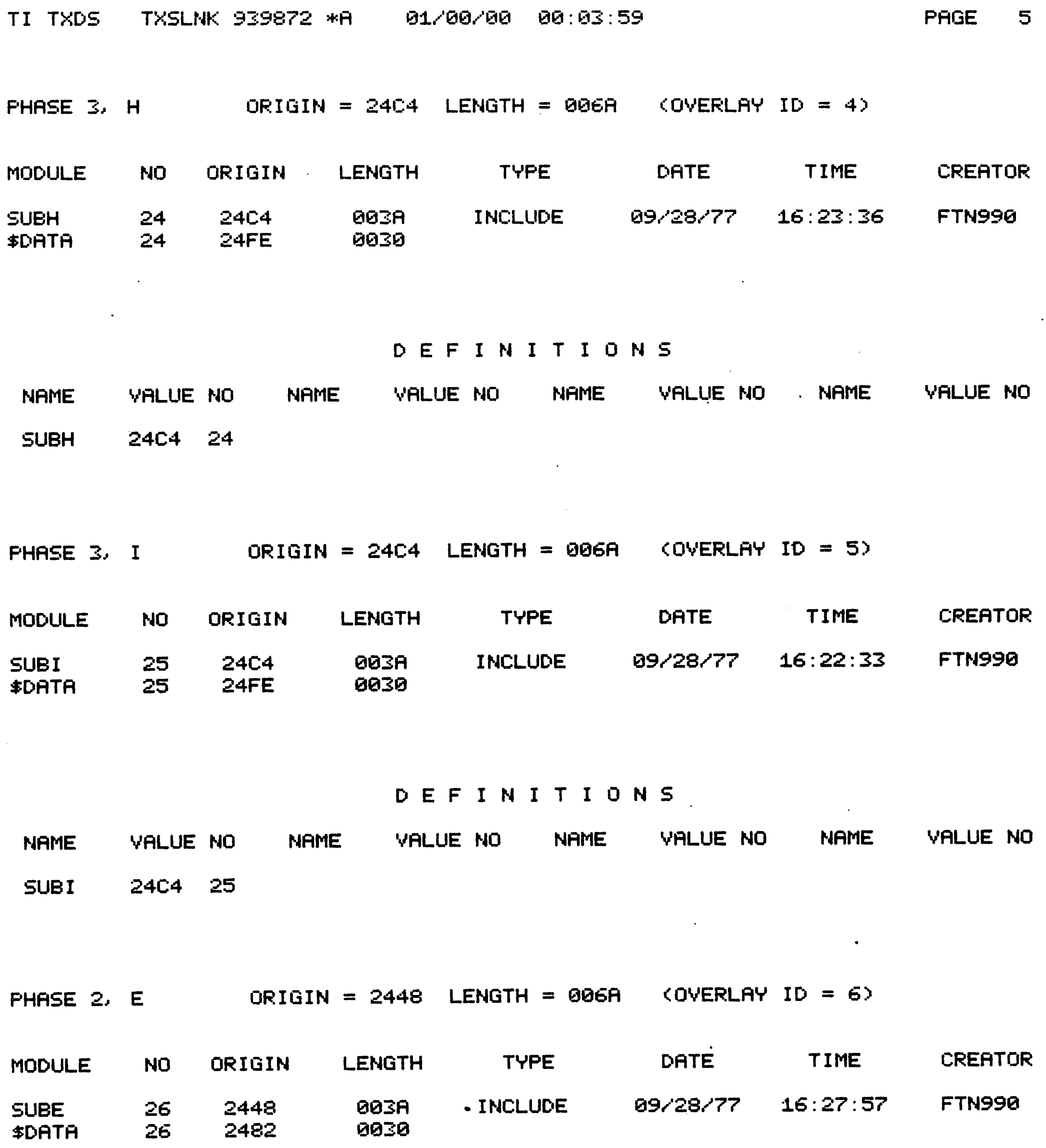
```
TI TXDS   TXSLNK 939872 *A    01/00/00   00:03:59                        PAGE    3
 NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

 F$FIOL   0C92   6     F$FIUS   0C26   6    *F$FLAG   0005*  3     F$FLOL   0CF0   6
 F$FLUS   0C84   6     F$FRE    0A78   6     F$FREB   0A72   6     F$FRED   0A6C   6
 F$FRER   0A66   6     F$FRF    0A90   6     F$FRFB   0A8A   6     F$FRFD   0A84   6
 F$FRFR   0A7E   6     F$FROL   0CC0   6     F$FRUS   0C54   6     F$FSIO   0D5E   6
 F$FWE    0A54   6     F$FWER   0A4E   6     F$FWF    0A60   6     F$FWFR   0A5A   6
 F$ILOG   1F12   3    *F$LSTA   0001*  3    *F$LUNO   0000*  3    *F$NAME   0008*  3
*F$PRB    0002*  3    *F$R10A   0014*  3    *F$R10B   013C*  3     F$RBUF   2242   17
*F$RCOL   0A0E   5    *F$RCUS   09F2   5    *F$RDEN   09A6   5    *F$RDOL   0A06   5
*F$RDUS   09EA   5    *F$RENN   09A2   5    *F$REOL   09FA   5    *F$REUS   09DE   5
 F$REVP   0996   4    *F$RFFD   1342   6    *F$RFOL   09FE   5    *F$RFRW   12CA   6
*F$RFSI   0D6E   6    *F$RFSR   13DC   6    *F$RFSW   1378   6    *F$RFUS   09E2   5
*F$RFWD   12BA   6    *F$RIOL   09F6   5    *F$RIUS   09DA   5     F$RLOG   0148*  3
*F$RLOL   0A0A   5    *F$RLP2   014A*  3    *F$RLUS   09EE   5    *F$RPAU   02CC   3
*F$RPRE   037C   3    *F$RRE    09C6   5    *F$RREB   09C2   5    *F$RRED   09BE   5
*F$RREL   165A   8    *F$RRER   09BA   5    *F$RRF    09D6   5    *F$RRFB   09D2   5
*F$RRFD   09CE   5    *F$RRFR   09CA   5    *F$RROL   0A02   5    *F$RRUS   09E6   5
 F$RSIO   0A12   5     F$RSTO   02DE   3    *F$RTFG   1DF0   3     F$RVFB   0028*  3
*F$RVP2   002A*  3    *F$RWE    09AE   5    *F$RWER   09AA   5     F$RWF    09B6   5
*F$RWFR   09B2   5     F$RWRK   1DCA   3    *F$STAT   0004*  3    *F$XAR    1722   10
*F$XBCS   010A*  3     F$XBFS   008A*  17   *F$XBUI   1604   7    *F$XBUO   15F0   7
*F$XBUT   0868   3    *F$XCDE   1A84   11   *F$XCDI   1A80   11   *F$XCED   1680   9
*F$XCER   167C   9    *F$XCID   1678   9    *F$XCIR   1674   9    *F$XCLS   04E4   3
*F$XCRE   1A7C   11   *F$XCRI   1A78   11   *F$XDR    1922   10   *F$XEOF   15E6   7
 F$XERR   1C78   13    F$XFCB   2176   16    F$XFCE   223E   16   *F$XFND   15EC   7
 F$XFTL   1C82   15   *F$XLIO   084E   3    *F$XLOG   079A   3    *F$XLR    165A   8
*F$XLWS   1F16   3    *F$XMR    187E   10   *F$XNGR   1666   8     F$XPRE   027E   3
*F$XPSE   0480   3     F$XRED   151E   7     F$XRST   1C88   18   *F$XRWD   15CE   7
*F$XSA1   0111*  3    *F$XSA2   0112*  3    *F$XSR    171A   10   *F$XSTC   010C*  3
*F$XSTL   010E*  3    *F$XSTP   0488   3    *F$XSTR   1660   8    *F$XSVC   0110*  3
 F$XTBE   2176   14    F$XTBL   1F96   14   *F$XTID   1EE3   3    *F$XTRA   15C6   7
*F$XVBF   0100*  3    *F$XVCC   00FB*  3    *F$XVCH   0104*  3    *F$XVCL   00FF*  3
*F$XVCO   00FC*  3    *F$XVFB   056E   3    *F$XVRC   0102*  3    *F$XVRO   00FE*  3
*F$XVST   00FD*  3    *F$XVSV   00FA*  3    *F$XVWS   1EF2   3     F$XWRT   153C   7
*G$XE01   0896   3    *G$XE08   08A9   3    *G$XE09   08CB   3    *G$XE10   08F8   3
*G$XE11   0913   3    *G$XE12   092E   3    *G$XE13   0946   3     G$XE14   0984   3
 L$$OBM   22CC   2    *L$$OVM   0108   2    *N$COLS   0106*  3    *N$LINS   0108*  3
*N$TID    0119*  3    *P$ABUF   0006*  3    *P$CCNT   000A*  3    *P$ERR    0001*  3
*P$LACN   0016*  3    *P$LFIL   0011*  3    *P$LIBF   0010*  3    *P$LLRL   0012*  3
*P$LPRL   0014*  3    *P$LUN    0003*  3    *P$OP     0002*  3     P$PFCB   0018*  3
*P$PRB    0000*  3    *P$PRBE   001A*  3    *P$REC1   000D*  3    *P$REC2   000E*  3
*P$RECL   0008*  3    *P$RES    000C*  3    *P$SFLG   0004*  3    *P$SVC0   0000*  3
*P$UFLG   0005*  3    *S$APRB   06C4   3



PHASE 1, A         ORIGIN = 230C   LENGTH = 013C   (OVERLAY ID = 1)


MODULE     NO    ORIGIN    LENGTH      TYPE        DATE        TIME       CREATOR

SUBA       19     230C      004C     INCLUDE     09/28/77    16:34:35    FTN990
$DATA      19     2358      0030
F$RGMY     20     2388      007E     FIND        11/01/77    19:51:38    SDSLNK
F$RAER     21     2406      0041     FIND        11/01/77    19:41:55    SDSLNK
```

Figure 4-7. TXDS Examples With Overlays (Sheet 3 of 7)



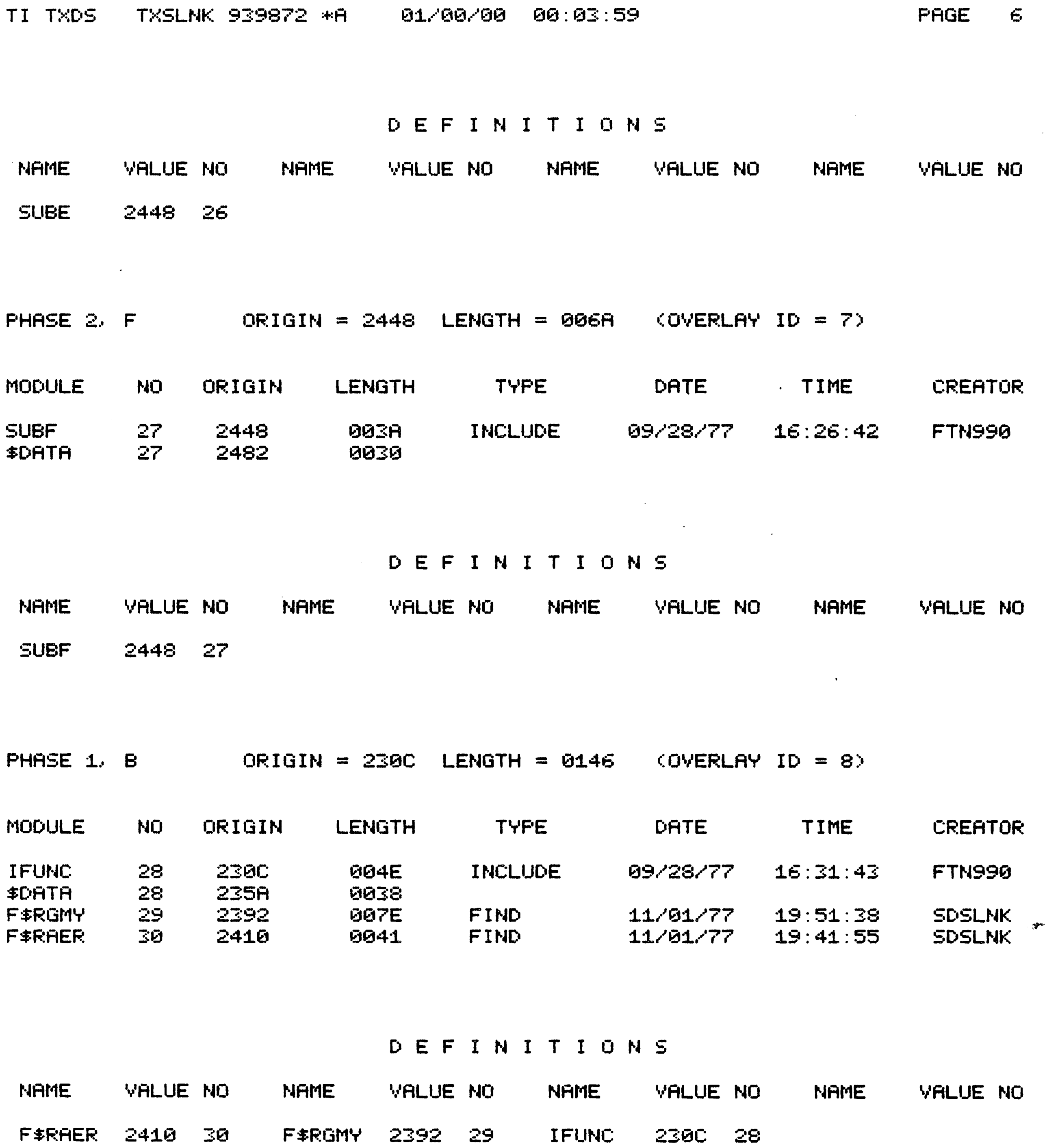
```
TI TXDS   TXSLNK 939872 *A    01/00/00  00:03:59                      PAGE   4


                                   D E F I N I T I O N S

 NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO

 F$RAER    2406  21      F$RGMY    2388  20      SUBA      230C  19


PHASE 2, D          ORIGIN = 2448  LENGTH = 007C   (OVERLAY ID = 2)

MODULE      NO    ORIGIN     LENGTH        TYPE          DATE          TIME        CREATOR

SUBD        22     2448       004C       INCLUDE      09/28/77     16:29:13      FTN990
$DATA       22     2494       0030


                                   D E F I N I T I O N S

 NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO

 SUBD      2448  22


PHASE 3, G          ORIGIN = 24C4  LENGTH = 006A   (OVERLAY ID = 3)

MODULE      NO    ORIGIN     LENGTH        TYPE          DATE          TIME        CREATOR

SUBG        23     24C4       003A       INCLUDE      09/28/77     16:25:30      FTN990
$DATA       23     24FE       0030


                                   D E F I N I T I O N S

 NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO      NAME      VALUE NO

 SUBG      24C4  23
```

Figure 4-7. TXDS Examples With Overlays (Sheet 4 of 7)



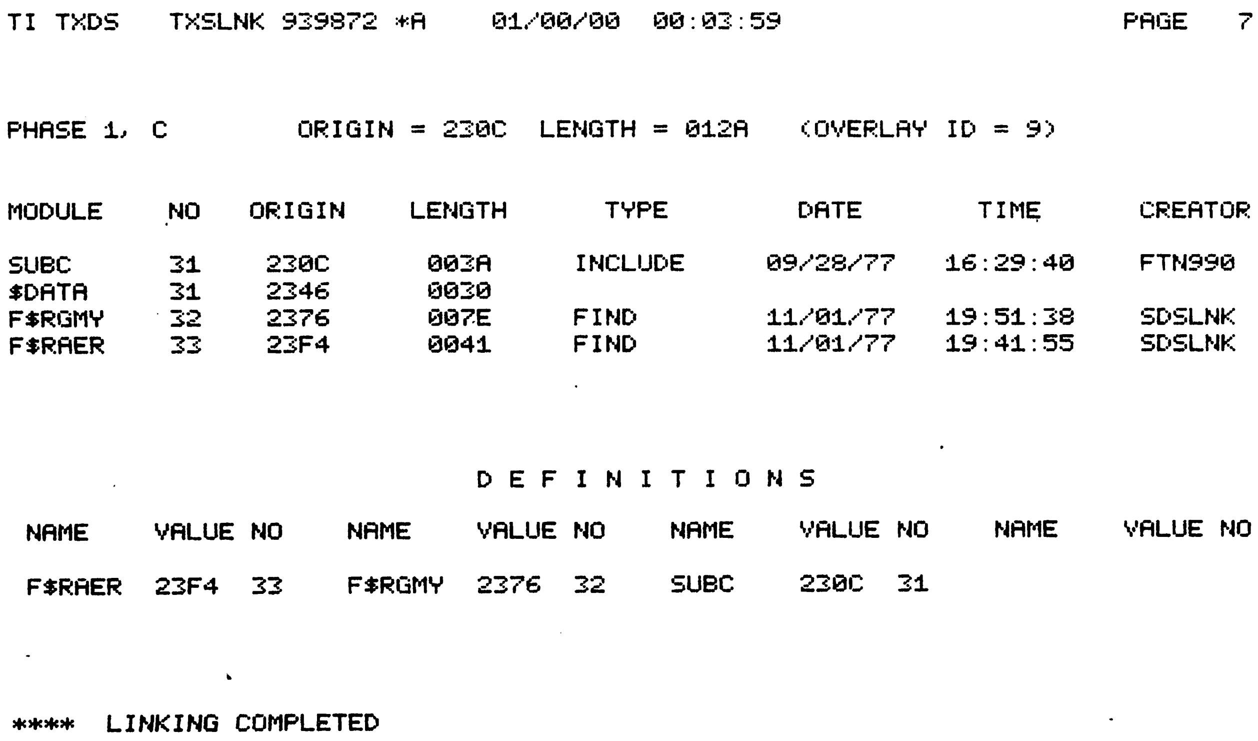
```
TI TXDS   TXSLNK 939872 *A    01/00/00  00:03:59                  PAGE   5


PHASE 3, H         ORIGIN = 24C4  LENGTH = 006A   (OVERLAY ID = 4)

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE       TIME      CREATOR

SUBH       24    24C4      003A     INCLUDE     09/28/77   16:23:36    FTN990
$DATA      24    24FE      0030


                              D E F I N I T I O N S

  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  SUBH     24C4  24


PHASE 3, I         ORIGIN = 24C4  LENGTH = 006A   (OVERLAY ID = 5)

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE       TIME      CREATOR

SUBI       25    24C4      003A     INCLUDE     09/28/77   16:22:33    FTN990
$DATA      25    24FE      0030


                              D E F I N I T I O N S

  NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO     NAME     VALUE NO

  SUBI     24C4  25


PHASE 2, E         ORIGIN = 2448  LENGTH = 006A   (OVERLAY ID = 6)

MODULE     NO   ORIGIN    LENGTH      TYPE        DATE       TIME      CREATOR

SUBE       26    2448      003A     INCLUDE     09/28/77   16:27:57    FTN990
$DATA      26    2482      0030
```

Figure 4-7. TXDS Examples With Overlays (Sheet 5 of 7)



```
TI TXDS    TXSLNK 939872 *A     01/00/00  00:03:59                          PAGE    6


                                        D E F I N I T I O N S

 NAME       VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 SUBE       2448  26



PHASE 2, F          ORIGIN = 2448  LENGTH = 006A   (OVERLAY ID = 7)

MODULE      NO    ORIGIN    LENGTH       TYPE         DATE         TIME        CREATOR

SUBF        27     2448      003A      INCLUDE      09/28/77     16:26:42      FTN990
$DATA       27     2482      0030


                                        D E F I N I T I O N S

 NAME       VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 SUBF       2448  27



PHASE 1, B          ORIGIN = 230C  LENGTH = 0146   (OVERLAY ID = 8)

MODULE      NO    ORIGIN    LENGTH       TYPE         DATE         TIME        CREATOR

IFUNC       28     230C      004E      INCLUDE      09/28/77     16:31:43      FTN990
$DATA       28     235A      0038
F$RGMY      29     2392      007E      FIND         11/01/77     19:51:38      SDSLNK
F$RAER      30     2410      0041      FIND         11/01/77     19:41:55      SDSLNK


                                        D E F I N I T I O N S

 NAME       VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

 F$RAER     2410  30     F$RGMY    2392  29     IFUNC     230C  28
```

Figure 4-7. TXDS Examples With Overlays (Sheet 6 of 7)



```
TI TXDS   TXSLNK 939872 *A    01/00/00  00:03:59                         PAGE   7


PHASE 1, C        ORIGIN = 230C  LENGTH = 012A   (OVERLAY ID = 9)

MODULE     NO   ORIGIN    LENGTH      TYPE         DATE         TIME       CREATOR

SUBC       31    230C      003A     INCLUDE      09/28/77    16:29:40     FTN990
$DATA      31    2346      0030
F$RGMY     32    2376      007E     FIND         11/01/77    19:51:38     SDSLNK
F$RAER     33    23F4      0041     FIND         11/01/77    19:41:55     SDSLNK


                                   DEFINITIONS

  NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO     NAME      VALUE NO

  F$RAER    23F4  33     F$RGMY    2376  32     SUBC      230C  31


****  LINKING COMPLETED
```

Figure 4-7. TXDS Examples With Overlays (Sheet 7 of 7)

SECTION V

LINK EDITOR USE ON DX10

5.1 SUPPORTED FEATURES

The disc based operating system for the Model 990/10 Minicomputer, DX10, is a multitasking operating system, and it supports all of the features of the Link Editor. Being a disc-based system, DX10 is well suited for overlay structured programs, and it also supports the Automatic Overlay Loading feature described in Section II of this manual. Complete information on DX10 can be found in the five volume DX10 Operating System Reference Manuals, Manual numbers 946250-9701, 9702, 9703, 9704, and 9705.

The following Link Editor features are supported by DX10:

- Automatic Overlay Loading
- Random Libraries
- Sequential Libraries
- COBOL program linking
- FORTRAN program linking
- Image format.

5.2 LINK EDITOR OPERATION WITH DX10

The first step in performing a Link Edit run is to develop a control file that defines the Link Edit functions. The Control File can be developed using the DX10 Text Editor (defined in the DX10 Operating System Developmental Operation Guide, manual 946250-9704), or it can be developed as a card, tape, or cassette file. The control file contains commands as described in Section III of the manual and can also contain object modules.

The Link Editor is called from a station by entering the command XLE. Note that the station must be in the command mode prior to entering the command (to activate the command mode, press the CMD key on the 911 VDT, the HELP key on the 913 VDT, or by simultaneously pressing the CONTROL and X keys on a hard-copy device). When XLE is entered, the following display is presented at a VDT (on a hard copy device, the prompts are printed one at a time).

```
XLE

EXECUTE LINKAGE EDITOR

          CONTROL ACCESS NAME:

    LINKED OUTPUT ACCESS NAME:   DUMY

          LISTING ACCESS NAME:   DUMY

                  PRINT WIDTH:   80
```

In response to the CONTROL ACCESS NAME: prompt, the user must enter the pathname of the device or file from which the control stream is to be read. The control file can be on a sequential disc file, or any sequential device such as a tape unit, cassette unit, or cards. The following is an example of the pathname entry for a sequential disc file:

```
CONTROL ACCESS NAME: VOL2. EDITOR.CONFILE
```

Note that there is no default for the CONTROL ACCESS NAME:, DX10 does not allow the user to TAB out of the field.

In response to the LINKED OUTPUT ACCESS NAME prompt, the user enters the access name of the sequential device or file to which the output of the Link Editor is to be written. Note that the user can enter the value DUMY, which causes no output to be generated. Use of the DUMY value allows for a trial run to ensure that no errors occur. The following is an example of an access name entry for a sequential disc file:

```
LINKED OUTPUT ACCESS NAME: VOL2.LINK.OUT1
```

If the FORMAT Link Editor command specifies the IMAGE option, the entry made in response to the LINKED OUTPUT ACCESS NAME prompt must be a DX10 Program file or a System Image file.

In response to the LISTING ACCESS NAME: prompt, the user enters the access name of the file or device to which the load map listing is to be written. If DUMY is used no listing is created. The value entered in response to the prompt can be any valid DX10 pathname, synonym, or device name. The following example causes the listing to be written to a line printer.

```
LISTING ACCESS NAME: LP01
```

An example and description of the Load Map listing is provided in Section IV.

The last prompt, PRINT WIDTH: allows the user to either specify the width of the print line, or to accept the default value - 80 characters. If a different line width is specified, the print format is altered to correspond to the line width.

The following example shows the responses for the prompt when the control file is on VOL1.LINK.OUT1, the listing device is line printer one (LP01), and the LINE WIDTH default value is accepted:

```
XLE

EXECUTE LINKAGE EDITOR

        CONTROL ACCESS NAME:   VOL2.EDITOR.CONFILE

  LINKED OUTPUT ACCESS NAME:   VOL2.LINK.OUT1

        LISTING ACCESS NAME:   LP01

               PRINT WIDTH:   80
```

Before calling the Link Editor, the user should be aware that the maximum memory available for a Link Edit is 64K bytes minus the size of the Link Editor. To determine the amount of memory required, the following guidelines should be used.

Allow	16 bytes for each external reference
+	40 bytes for each included module
+	10% of the sum of the above.

SECTION VI

LINK EDITOR USE ON TXDS

6.1 SUPPORTED FEATURES

The Terminal Executive Development System (TXDS) is a memory-resident operating system for the Model 990/4 and Model 990/10 minicomputers. For complete information on TXDS, refer to the *Model 990 Computer Terminal Executive Development System (TXDS) Programmer's Guide*, manual 946258-9701. The Link Editor is a utility program on TXDS and is named TXSLNK.

Being a memory based operating system, TXDS does not support certain of the Link Editor features, particularly those concerned with random libraries. TXDS does not support the following Link Editor features:

- SYMT output option
- LIBRARY/SEARCH commands

6.2 OPERATION

To activate the Link Editor, TXDS itself must be activated by following the procedure given in Section II of the TXDS Programmer's Guide. Once TXDS is activated, the following message is presented at the system console:

```
TXDS *A ddd/yy hh:mm

PROGRAM:
```

In the preceding message, ddd/yy represents the date, displayed in the Julian format with a three digit day of the year (ddd) and the two digit year (yy). The second line of the message, PROGRAM: is a prompt that requests entry of the pathname of the program to be executed. To activate the Link Editor, the following response is entered:

```
PROGRAM:   :TXSLNK/SYS
```

which defines the file that contains the TXSLNK object module. Once this entry is completed, the prompt:

```
INPUT:
```

is presented at the system console. In response to this prompt, the user enters the pathname of the file or device from which the control file is to be read. Examples are:

INPUT: LOG	Control file is to be entered from the system console
INPUT: DSC2:FILEA/CTL	Control file is to be read from :FILEA/CTL on diskette unit two

The next prompt presented is:

OUTPUT:

which requests that the following entries be made:

$$\text{OUTPUT:}\quad \text{<object>} \left\{ \begin{matrix} \text{,<load map>} \\ \underline{\text{, sys default printer}} \end{matrix} \right\} \qquad \left\{ \begin{matrix} \text{,<scratch>} \\ \underline{\text{,TXSLNK diskette}} \end{matrix} \right\}$$

The parameters have the following meanings:

object — specifies the object output pathname

load map — specifies the output listing pathname. If no entry is made, system default printer is used.

scratch — specifies the diskette unit upon which the scratch files will be created. If no value is entered, the scratch files are created on the same diskette unit from which the Link Editor was loaded.

NOTE

There can be no blanks within a parameter line.

After the OUTPUT: prompt parameters, the following prompt is presented:

OPTIONS:

In response to this prompt, the user can either enter the number of bytes (in decimal) of memory available to the Link Editor for the table area, or make no entry and accept the default of 8192 bytes. The following considerations apply to calculating memory requirements:

Allow 16 bytes per external definition

+ 40 bytes per included module

+ 10% of the sum of the above.

Any entry made by the user in response to the OPTIONS: prompt is preceded by the letter M. An example of a request for 3000 bytes of memory is:

OPTIONS: M3000

6.3 PATHNAME DEFAULTS

Fields of a pathname that are not entered by the user are defaulted according to table 6-1.

Table 6-1. TXDS Pathname Defaults

	DEV	FILE	EXT
INPUT	System Default Diskette Name	None	CTL
OBJECT	System Default Diskette Name	None	OBJ
LISTING	System Default Diskette Name	None	LST

NOTE

If the Link Editor is operating in IMAGE format, the object default extension is SYS.

6.4 EXAMPLES

The following are examples of the responses to the prompts presented by TXDS:

Example 1:

```
PROGRAM: :TXSLNK/
  INPUT: :LINK/CTL
 OUTPUT: DSC2:PROG/,LP
OPTIONS: M10000
```

Example 2:

```
PROGRAM: :TXSLNK/
  INPUT: :FINLNK/
 OUTPUT: DSC2:FORTGO/
OPTIONS: M12000
```

Example one causes the control file to be read from a file identified as LINK with the extension being CTL, on the default diskette. The output is written to file PROG, with the default extension OBJ. on the second diskette (DSC2). The listing is written to the line printer. The scratch files are written to the diskette that TXSLNK was loaded from. The memory default is overridden with a specification of 10,000 bytes.

Example two reads the control file from the default diskette from a file named FINLNK with the default extension being CTL. The output is written to the second diskette (DSC2) in file FORTGO and the default extension is OBJ. The load map is written to the system default printer. The scratch files are written to the same diskette that TXSLNK has loaded from. The memory specified is 12,000 bytes.

SECTION VII

ERROR REPORTING

7.1 INTRODUCTION

Errors that occur during Link Editor processing are reported on either the terminal or the listing file. Table 7-1 gives the errors that are reported on the listing file and table 7-2 gives the erros that are reported on the terminal.

The use of the ERROR/NOERROR Link Editor commands affects the manner in which the Link Editor responds to errors. The ERROR command allows continuation of processing after an error occurs, whereas the NOERROR command terminates processing when an error occurs. See Section III for detailed information on the use of these commands.

Table 7-1. Link Editor Errors — List File

Message Text	Explanation
SYNTAX	A rule of syntax for Link Editor commands has been violated. Consult Section III for the correct syntax.[1]
COMMAND SEQUENCE	The command is not in the proper sequence in the command stream.[1]
DUPLICATE NAME	The same name has been used needlessly or ambiguously two or more times. The rest of the command record is ignored.[1]
COMMAND EOF	An unexpected end of file was encountered in the command stream.[1]
SIZE	Too many characters in a LIBRARY or INCLUDE name.[1]
UNABLE TO READ CONTROL FILE	Check the I/O error code shown on the terminal.
UNABLE TO OPEN CONTROL FILE	Check control file access name. If correct, check the I/O error code shown on the terminal.
ILLEGAL LIBRARY NAME ACCESS NAME = name	Check the access name for a legal random library name.
NO TASK COMMAND	Control file does not contain a TASK command.

[1] When any of these errors occur, a dollar sign ($) appears on the listing immediately under the error.

Table 7-1. Link Editor Errors — List File (Continued)

Message Text	Explanation
ILLEGAL DUMMY COMMAND, phase name	DUMMY command appeared after a procedure which was not dummied.
UNABLE TO OPEN INCLUDE FILE, ACCESS NAME = name	The file named does not exist or cannot be opened.
NO FIRST INPUT RECORD, ACCESS NAME = name	The file name existed, but an end of file was encountered on the first read.
ILLEGAL TAG ENCOUNTERED ON RECORD XX, ACCESS NAME = name	The object input was bad.
PREMATURE END OF FILE OCCURRED, ACCESS NAME = name	No colon record was found in the object input.
CHECKSUM ERROR ENCOUNTERED ON RECORD XX, ACCESS NAME = name	The object input was bad.
SYMBOL MULTIPLY DEFINED	A list of the multiply defined symbols, and the modules they occured in, follows. The first value assigned is used.
ILLEGAL COMMON REFERENCE ENCOUNTERED ON RECORD XX ACCESS NAME =	Bad object input.
ILLEGAL BACK CHAIN ACCESS NAME = acnm SYMBOL = external name	Bad object input.
UNABLE TO BACKSPACE INPUT FILE ACCESS NAME = name	Check input access name for a legal sequential library name. If correct, check the I/O error code shown on the the terminal.
ADDRESS SPACE OVERFLOW	The Link Edit has exceeded the 32K word maximum. A program cannot be larger than 32K.
ILLEGAL OVERLAY SEGMENT	Bad object file.
UNABLE TO OPEN OUTPUT FILE, ACCESS NAME = name	The output file cannot be opened, probably because of access rights violations. Check the I/O error code shown on the terminal.
UNABLE TO CLOSE OUTPUT FILE ACCESS NAME = name	Check the I/O error code shown on the terminal.
UNABLE TO WRITE OUTPUT RECORD ACCESS NAME = name	Check the I/O error code shown on the terminal.

[1] When any of these errors occur, a dollar sign ($) appears on the listing immediately under the error.

Table 7-1. Link Editor Errors — List File (Continued)

Message Text	Explanation
UNABLE TO OPEN OUTPUT FILE ACCESS NAME = acnm	Check the access name. If correct, check the I/O error code shown on the terminal.
UNABLE TO WRITE WORK RECORD	Link Editor scratch file error. Check I/O error code shown on terminal.
UNABLE TO READ OVERFLOW RECORD	Link Editor scratch file error. Check I/O error code shown on terminal.
UNABLE TO WRITE OVERFLOW RECORD	Link Editor scratch file error. Check I/O error code shown on terminal.
UNABLE TO READ WORK RECORD	Link Editor scratch file error. Check the I/O error code shown on the terminal.
ROLL POINTER OVERFLOW (TXDS) ROLL MEMORY OVERFLOW (TXDS)	User did not specify adequate memory.
CAN'T GET COMMON (TXDS)	The Link Editor cannot obtain System Common.
INTERNAL LINKER BUG, ENCOUNTERED AT LINKER LOCATION XXXX	A bug in the link editor has caused processing to terminate. Communicate the problem to the customer support line.
UNABLE TO INSERT PROCEDURE	Image format only. Replace option not selected and a procedure of the same name already exists.
UNABLE TO INSERT TASK	Image format only. Replace option not selected and a task of the same name already exists.
UNABLE TO INSERT OVERLAY	Image format only. Replace option not selected and an overlay of the same name already exists.
UNABLE TO FIND PROCEDURE name (DX10)	Image format only. A procedure that was 'dummied' does not exist on the program file. DX10 only.
UNABLE TO ASSIGN OVERLAY ID	Image format only. Indicates that no overlay IDs are available in the program file directory.
UNABLE TO LOAD REQUIRED MODULE MODULE NAME =	A module specified by a U-tag in the object module cannot be loaded.

Table 7-1. Link Editors — List File (Continued)

Message Text	Explanation
WARNING MESSAGES	
ADDRESS SPACE OVERFLOW	Program Counter (PC) in the linked output exceeded $FFFF_{16}$.
MULTIPLE SYMBOL DEFINITION	The listed symbol has been defined more than once. It is assigned the value of the last occurrence.
UNABLE TO LOAD nnnnnnn	Unable to perform a forced load of the listed module (nnnnnnn).
SHARE SPACE	When two or more modules share a data area (see the SHARE command), the first module included must have the largest data area. This message warns that a subsequent data area is larger.

Table 7-2. Link Editor Errors — Terminal

Message Text	Explanation
INPUT FILE I/O ERROR, CODE = XXXX	Unable to read or open the input file[1]. See listing for more information.
ILLEGAL TAG	Object code contained an illegal tag. See listing for more information.
SYNTAX ERROR	See listing for more information.
BAD OBJECT FORMAT	See listing for more information.
OUTPUT FILE I/O ERROR, CODE = XXXX	See listing for more information. Check output file access name[1].
LIST FILE I/O ERROR, CODE = XXXX	Unable to open or write the list file.[1]
UNABLE TO LOAD OVERLAY	One of the overlays of the Link Editor could not be loaded (see listing).
MISSING OR MISPLACED COMMANDS	Check your control file (see listing).
PROCEDURE IMAGE ERROR, CODE = XXXX	Unable to locate or install a procedure on the program file (see listing).[1]
CANNOT OPEN TCA	The system file TCAFIL cannot be opened (DX10).

[1] The CODE = XXXX is defined in table 7-3.

Table 7-2. Link Editor Errors — Terminal (Continued)

Message Text	Explanation
CANNOT READ TCA	An error occurred while reading the system file TCAFIL (DX10).
CANNOT CLOSE TCA	The system file TCAFIL cannot be closed (DX10).
UNABLE TO GET MEMORY (DX10) CAN'T GET MEMORY (TXDS)	The Link Editor cannot obtain sufficient memory. Try again after other tasks have terminated.
LINK EDITOR BUG	Error occurred within the link editor. See listing, and notify the Customer Support Line.
NO FIRST INPUT RECORD	Input file has an EOF and no data. See listing for more information.
PREMATURE END OF FILE	No colon record on input file. See listing for more information.
CHECKSUM ERROR	Checksum did not verify on an input record. See listing for more information.
TASK IMAGE ERROR, CODE = XXXX	Unable to locate or install a task on the program file (see listing).[1]
OVERLAY IMAGE ERROR, CODE = XXXX	Unable to locate or install an overlay on the program file (see listing).[1]
WORK FILE I/O ERROR, CODE = XXXX OVERFLOW FILE I/O ERROR, CODE = XXXX	I/O errors on Link Editor Scratch Files[1].
CONTROL FILE I/O ERROR, CODE = XXXX	Check control file access name (see listing)[1].
INVALID LIBRARY NAME	Check access names on library commands (see listing).
CAN'T GET COMMON	Insufficient common area in the system (TXDS).
MAXIMUM TABLE SIZE EXCEEDED	Too many references (REFs) and definitions (DEFs) in the Link Edit.
INSUFFICIENT MEMORY REQUESTED	

[1] The CODE = XXXX is defined in table 7-3.

Table 7-3. Error Codes

CODE = XX	MEANING
00XX	DX10 I/O error code. Refer to the *Model 990 Computer, DX10 Operation System Release 3 Reference Manual, Voluem VI Error Reporting and Recovery.*
80XX	TX990 I/O error codes. Refer to the *Model 990 Computer TX990 Operating System (Release 2) Programmer's Guide.*
8190	File is not relative record (TXDS).
8191	Record length of file is not 256 (TXDS).
8192	Attempted to use function that is not available on TXSLNK (TXDS).
8193	Too many overlays; more that 255 (TXDS).

APPENDIX A

OVERLAY MANAGER
(IMAGE FORMAT ONLY)

APPENDIX A

OVERLAY MANAGER

(IMAGE FORMAT ONLY)

A.1 OVERLAY MANAGER

The overlay manager is a table-driven program, with the two required tables being built by the Link Editor when the linked output is produced. The tables generated are the Overlay Entry Vector (OEV) table, defined in table A-1, and the Overlay Phase Directory (OPD), defined in table A-2. The OEV table is a read-only table that can be included in the procedure portion of a program. It contains an entry for each forward reference in the overlay structure, with each entry having a pointer to an entry in the OPD table. Each entry in the OPD corresponds to a PHASE command in the Link Editor Control File. The OPD is divided into two portions — a read-only part and a read/write part. The read-only part of the OPD can be included as a part of a shared procedure. The read/write part of the OPD, which consists of a flag that indicates whether the overlay is currently in memory, is included as a part of the task section of the linked object output module.

Table A-1. OEV Entry Format

Word	Description
0	Address of workspace for overlay manager.
1	Address of the new PC value for a BLWP instruction (equal to current address + 2ʃ.
2	A Branch and Link (BL) instruction to transfer control to the overlay manager. Equivalent to BL *R1.
3	Address of the transfer vectors (WP and PC) in the overlay.
4	Address of the entry in the OPD that describes the overlay to be loaded for this entry.

Table A-2. OPD Entry Format

Byte	Description
0-1	Overlay ID.[1]
2-3	Address of the OPD entry for the first overlay on the same level.[1]
4-5	Overlay load address.

Bit map to indicate whether the overlay is in memory (1 = yes, 0 = no).[2]

Notes:

[1] Read only information

[2] Read/write information, not contiguous with the rest of the OPD. Kept as a bit map indexed by the overlay ID.

APPENDIX B

TXDS LINKING LOADER

APPENDIX B

TXDS LINKING LOADER

B.1 INTRODUCTION

This appendix describes the TXDS Linking Loader (LNKLDR). The Linking Loader allows the user to link FORTRAN programs with the runtime package, to load the program and to execute it all in one operation. All LUNOs used by the program to be executed must be assigned prior to LNKLDR execution.

NOTE

> The LUNOs 2, $A5_{16}$, and $A6_{16}$ cannot be used by programs linked and loaded by the Linking Loader, as they are used by the Linking Loader.

B.2 USER INTERFACE

When TXDS is activated, following the procedures given in Section II of the TXDS Programmer's Reference Manual, the following prompts are presented:

```
PROGRAM:

  INPUT:

 OUTPUT:

OPTIONS:
```

In response to the PROGRAM: prompt, enter the pathname of the Linking Loader object. The following is an example of the response:

```
PROGRAM: :LNKLDR/SYS
```

which defines the Linking Loader as being on the default disc, with the file name LNKLDR and the extension being SYS.

Respond to the INPUT: prompt with a carriage return as the Linking Loader does not use this parameter. In response to the OUTPUT: prompt, enter the device name or pathname to which the load map is to be written. The Load Map is the only output of the Linking Loader and it may be directed to any file or device on the system. If a device name is entered and the name is illegal, an error message is printed and the Linking Loader terminates. The following is a valid device name causing the load map to be written to the line printer:

```
OUTPUT: LP
```

If the output is directed to a file and the file does not exist, a sequential file is created by TXDS using the entered pathname as the file identifier. Certain defaults are applied by the system to the pathname. The pathname defaults are as follows:

Field	Defaults
DEV	Default Diskette Name
FILE	None
EXT	LST

The following example causes the load map to be written to a file:

```
OUTPUT: DSC2:LOADMP/LST
```

If no entry is made in response to the OUTPUT: prompt, the system default printer is used as the output device.

The only response accepted to the OPTIONS: prompt is a memory size. If no entry is made, the default value, 8K bytes is used. The syntax of the entry is as follows:

```
MNNNNN
```

where N is a decimal digit. The following example specifies 15K bytes of memory:

```
OPTIONS: M15000
```

After all of the preceding prompts have been answered, the Linking Loader is loaded into memory and the following prompt is displayed:

```
COMMAND,PATHNAME:
```

The commands available and the syntax for each are as follows:

Command	Description
I, pathname	Include modules in the specified file.
F, pathname	Search the specified sequential library for unresolved symbol resolution.
G	Print load map and begin execution.

There must be no blanks in the entered command and pathname. The following are examples of the preceding commands:

```
COMMAND,PATHNAME:    I,DSC2:TICTAC/OBJ

COMMAND,PATHNAME:    F,:TXLOBJ/LIB

COMMAND PATHNAME:    G
```

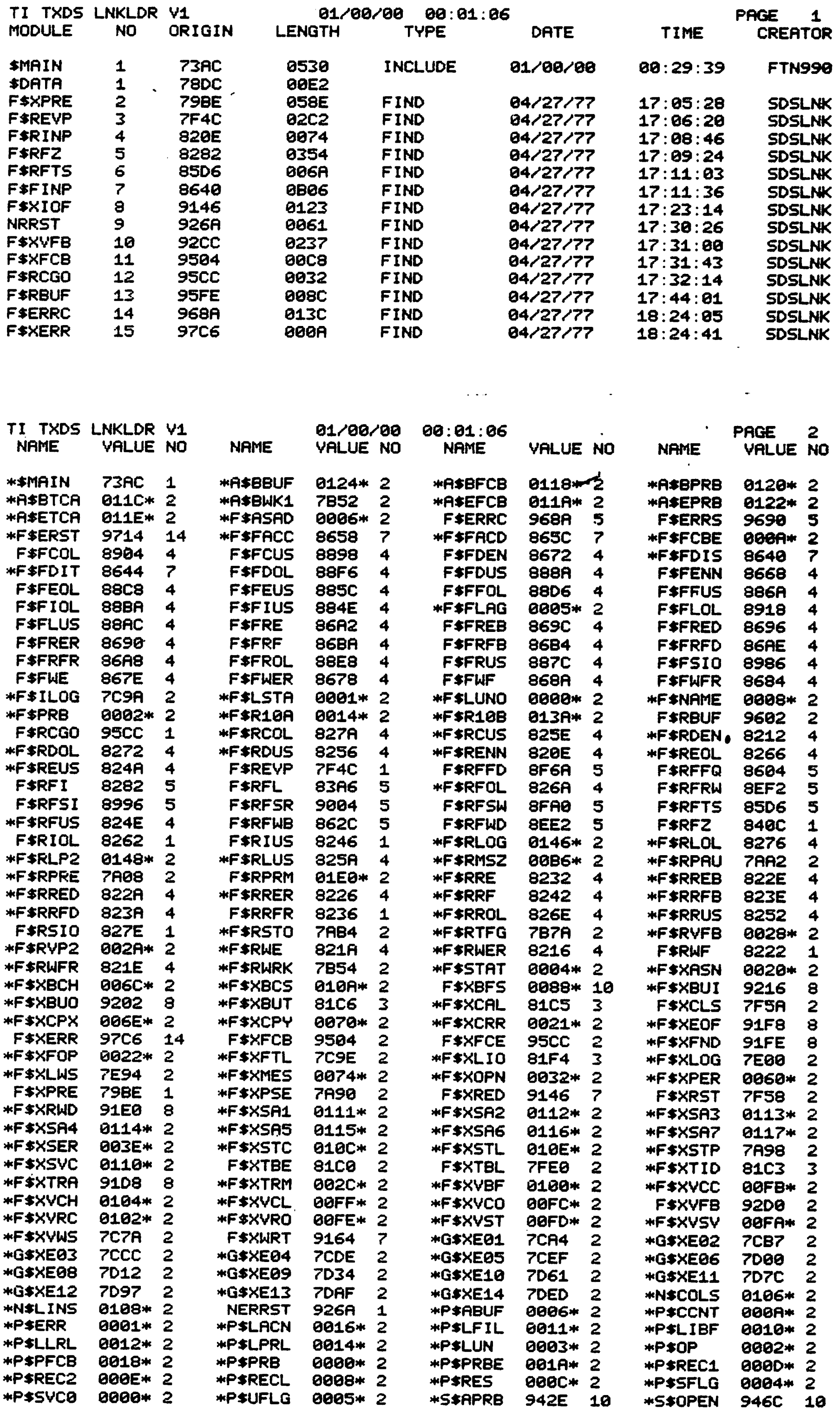
The COMMAND,PATHNAME: prompt is repeated until the G command is entered. More than one I command and/or F command may be entered.

Figure B-1 is the output load map of a sample run using the following parameters:

```
             PROGRAM:   :LNKLDR/SYS
               INPUT:
              OUTPUT:   LP
             OPTIONS:   M15000
    COMMAND,PATHNAME:   I,DSC2:TICTAC/OBJ
    COMMAND,PATHNAME:   F,:TXLOBJ/LIB
    COMMAND,PATHNAME:   G
```

B.3 TXDS LINKING LOADER ERRORS

The messages shown in table B-1 are those that can be printed on the log if an error occurs during execution of the Linking Loader.



```
TI TXDS LNKLDR V1            01/00/00  00:01:06                      PAGE  1
MODULE    NO   ORIGIN     LENGTH       TYPE        DATE          TIME     CREATOR

$MAIN     1     73AC       0530      INCLUDE     01/00/00      00:29:39     FTN990
$DATA     1     78DC       00E2
F$XPRE    2     79BE       058E      FIND        04/27/77      17:05:28     SDSLNK
F$REVP    3     7F4C       02C2      FIND        04/27/77      17:06:20     SDSLNK
F$RINP    4     820E       0074      FIND        04/27/77      17:08:46     SDSLNK
F$RFZ     5     8282       0354      FIND        04/27/77      17:09:24     SDSLNK
F$RFTS    6     85D6       006A      FIND        04/27/77      17:11:03     SDSLNK
F$FINP    7     8640       0B06      FIND        04/27/77      17:11:36     SDSLNK
F$XIOF    8     9146       0123      FIND        04/27/77      17:23:14     SDSLNK
NRRST     9     926A       0061      FIND        04/27/77      17:30:26     SDSLNK
F$XVFB    10    92CC       0237      FIND        04/27/77      17:31:00     SDSLNK
F$XFCB    11    9504       00C8      FIND        04/27/77      17:31:43     SDSLNK
F$RCGO    12    95CC       0032      FIND        04/27/77      17:32:14     SDSLNK
F$RBUF    13    95FE       008C      FIND        04/27/77      17:44:01     SDSLNK
F$ERRC    14    968A       013C      FIND        04/27/77      18:24:05     SDSLNK
F$XERR    15    97C6       000A      FIND        04/27/77      18:24:41     SDSLNK



TI TXDS LNKLDR V1            01/00/00  00:01:06                      PAGE  2
 NAME     VALUE NO      NAME     VALUE NO      NAME     VALUE NO      NAME     VALUE NO

**$MAIN   73AC  1      *A$BBUF   0124* 2      *A$BFCB   0118* 2      *A$BPRB   0120* 2
*A$BTCA   011C* 2      *A$BWK1   7B52  2      *A$EFCB   011A* 2      *A$EPRB   0122* 2
*A$ETCA   011E* 2      *F$ASAD   0006* 2       F$ERRC   968A  5       F$ERRS   9690  5
*F$ERST   9714  14     *F$FACC   8658  7      *F$FACD   865C  7      *F$FCBE   000A* 2
 F$FCOL   8904  4       F$FCUS   8898  4       F$FDEN   8672  4      *F$FDIS   8640  7
*F$FDIT   8644  7       F$FDOL   88F6  4       F$FDUS   888A  4       F$FENN   8668  4
 F$FEOL   88C8  4       F$FEUS   885C  4       F$FFOL   88D6  4       F$FFUS   886A  4
 F$FIOL   88BA  4       F$FIUS   884E  4      *F$FLAG   0005* 2       F$FLOL   8918  4
 F$FLUS   88AC  4       F$FRE    86A2  4       F$FREB   869C  4       F$FRED   8696  4
 F$FRER   8690  4       F$FRF    86BA  4       F$FRFB   86B4  4       F$FRFD   86AE  4
 F$FRFR   86A8  4       F$FROL   88E8  4       F$FRUS   887C  4       F$FSIO   8986  4
 F$FWE    867E  4       F$FWER   8678  4       F$FWF    868A  4       F$FWFR   8684  4
*F$ILOG   7C9A  2      *F$LSTA   0001* 2      *F$LUNO   0000* 2      *F$NAME   0008* 2
*F$PRB    0002* 2      *F$R10A   0014* 2      *F$R10B   013A* 2       F$RBUF   9602  2
 F$RCGO   95CC  1      *F$RCOL   827A  4      *F$RCUS   825E  4      *F$RDEN   8212  4
*F$RDOL   8272  4      *F$RDUS   8256  4      *F$RENN   820E  4      *F$REOL   8266  4
*F$REUS   824A  4       F$REVP   7F4C  1       F$RFFD   8F6A  5       F$RFFQ   8604  5
 F$RFI    8282  5       F$RFL    83A6  5      *F$RFOL   826A  4       F$RFRW   8EF2  5
 F$RFSI   8996  5       F$RFSR   9004  5       F$RFSW   8FA0  5       F$RFTS   85D6  5
*F$RFUS   824E  4       F$RFWB   862C  5       F$RFWD   8EE2  5       F$RFZ    840C  1
 F$RIOL   8262  1       F$RIUS   8246  1      *F$RLOG   0146* 2      *F$RLOL   8276  4
*F$RLP2   0148* 2      *F$RLUS   825A  4      *F$RMSZ   00B6* 2      *F$RPAU   7AA2  2
*F$RPRE   7A08  2       F$RPRM   01E0* 2      *F$RRE    8232  4      *F$RREB   822E  4
*F$RRED   822A  4      *F$RRER   8226  4      *F$RRF    8242  4      *F$RRFB   823E  4
*F$RRFD   823A  4       F$RRFR   8236  1      *F$RROL   826E  4      *F$RRUS   8252  4
 F$RSIO   827E  1      *F$RSTO   7AB4  2      *F$RTFG   7B7A  2      *F$RVFB   0028* 2
*F$RVP2   002A* 2      *F$RWE    821A  4      *F$RWER   8216  4       F$RWF    8222  1
*F$RWFR   821E  4      *F$RWRK   7B54  2      *F$STAT   0004* 2      *F$XASN   0020* 2
*F$XBCH   006C* 2      *F$XBCS   010A* 2       F$XBFS   0088* 10     *F$XBUI   9216  8
*F$XBUO   9202  8      *F$XBUT   81C6  3      *F$XCAL   81C5  3       F$XCLS   7F5A  2
*F$XCPX   006E* 2      *F$XCPY   0070* 2      *F$XCRR   0021* 2      *F$XEOF   91F8  8
 F$XERR   97C6  14      F$XFCB   9504  2       F$XFCE   95CC  2      *F$XFND   91FE  8
*F$XFOP   0022* 2      *F$XFTL   7C9E  2      *F$XLIO   81F4  3      *F$XLOG   7E00  2
*F$XLWS   7E94  2      *F$XMES   0074* 2      *F$XOPN   0032* 2      *F$XPER   0060* 2
 F$XPRE   79BE  1      *F$XPSE   7A90  2       F$XRED   9146  7       F$XRST   7F58  2
*F$XRWD   91E0  8      *F$XSA1   0111* 2      *F$XSA2   0112* 2      *F$XSA3   0113* 2
*F$XSA4   0114* 2      *F$XSA5   0115* 2      *F$XSA6   0116* 2      *F$XSA7   0117* 2
*F$XSER   003E* 2      *F$XSTC   010C* 2      *F$XSTL   010E* 2      *F$XSTP   7A98  2
*F$XSVC   0110* 2       F$XTBE   81C0  2       F$XTBL   7FE0  2      *F$XTID   81C3  3
*F$XTRA   91D8  8      *F$XTRM   002C* 2      *F$XVBF   0100* 2      *F$XVCC   00FB* 2
*F$XVCH   0104* 2      *F$XVCL   00FF* 2      *F$XVCO   00FC* 2       F$XVFB   92D0  2
*F$XVRC   0102* 2      *F$XVRO   00FE* 2      *F$XVST   00FD* 2      *F$XVSV   00FA* 2
*F$XVWS   7C7A  2       F$XWRT   9164  7      *G$XE01   7CA4  2      *G$XE02   7CB7  2
*G$XE03   7CCC  2      *G$XE04   7CDE  2      *G$XE05   7CEF  2      *G$XE06   7D00  2
*G$XE08   7D12  2      *G$XE09   7D34  2      *G$XE10   7D61  2      *G$XE11   7D7C  2
*G$XE12   7D97  2      *G$XE13   7DAF  2      *G$XE14   7DED  2      *N$COLS   0106* 2
*N$LINS   0108* 2       NERRST   926A  1      *P$ABUF   0006* 2      *P$CCNT   000A* 2
*P$ERR    0001* 2      *P$LACN   0016* 2      *P$LFIL   0011* 2      *P$LIBF   0010* 2
*P$LLRL   0012* 2      *P$LPRL   0014* 2      *P$LUN    0003* 2      *P$OP     0002* 2
*P$PFCB   0018* 2      *P$PRB    0000* 2      *P$PRBE   001A* 2      *P$REC1   000D* 2
*P$REC2   000E* 2      *P$RECL   0008* 2      *P$RES    000C* 2      *P$SFLG   0004* 2
*P$SVC0   0000* 2      *P$UFLG   0005* 2      *S$APRB   942E  10     *S$OPEN   946C  10
```

Figure B-1. Linking Loader Load Map

Table B-1. Link Loader Errors

Message	Description
MEMORY OVERFLOW	Insufficient memory.
INIT BLANK COMMON	Attempt to initialize blank common.
CHECKSUM ERROR	Checksum did not verify on an input record.
COMMON TOO BIG	Blank common larger than available space.
INPUT FILE ERROR, CODE = NNNN	Error reading the input file. Check the error code (NNNN) in the TX990 documentation.
PRINT FILE ERROR, CODE = NNNN	Error writing the load map. Check the error code (NNNN) in the TX990 documentation.
ILLEGAL BACK CHAIN	The external reference chain is bad.
ILLEGAL COMMON REF	A reference to a common segment is invalid.
ILLEGAL TAG	Object input contains an illegal tag.
NO FIRST INPUT RECORD	Input file has an End of File mark, but no data.
PREMATURE END OF FILE	No colon on the input file.
CAN'T GET COMMON	Cannot obtain TXDS system common area.
CAN'T GET MEMORY	Memory requested (in response to the OPTIONS prompt) cannot be obtained.
COMMAND/PATHNAME ERROR, CODE = NNNN	Command pathname or command is in error. Check the error code (NNNN) in the TX documentation. If the command is in error, the CODE = 0.
LINK LOAD ERROR	An error occurred within the Link Loader. Contact a Texas Instruments representative.

APPENDIX C

COMMAND SYNTAX

APPENDIX C

COMMAND SYNTAX

Table C-1 lists the commands provided by the Link Editor. The syntax of the command is given, and the paragraph in which the command is described is referenced.

Table C-1. Commands

Command	**Syntax**	**Paragraph**
ADJUST	ADJUST<n>	3.3.9
ALLGLOBAL	ALLGLOBAL	3.3.7.3
ALLOCATE	ALLOCATE	3.3.4
COMMON	COMMON <base>,[<name>] [,<name>]	3.6.3
DATA	DATA <base>	3.6.2
DUMMY	DUMMY	3.3.8
ERROR	ERROR	3.5.5
FIND	FIND <acnm>	3.2.5
FORMAT	FORMAT $\left\{\begin{array}{l}\text{ASCII}\\ \text{COMPRESSED}\\ \text{IMAGE [REPLACE]} \left\{\begin{array}{l}\text{<,priority>}\\ \underline{4}\end{array}\right\}\end{array}\right\}$	3.5.1
GLOBAL	GLOBAL [<symbol>] [,<symbol>] [, . . .]	3.3.7.2
INCLUDE	*INCLUDE ƀ<acnm>[,<acnm>] [, . . .]	3.2.1
LIBRARY[1]	LIBRARY <acnm>[,<acnm>] [. . . .]	3.2.2
LOAD	LOAD	3.3.4
MAP	MAP REFS NO <'string'> [,NO <'string'>] [, . . .]	3.5.2
NOAUTO[1]	NOAUTO	3.2.4
NOERROR	NOERROR	3.5.5
NOLOAD	NOLOAD	3.3.5
NOMAP	NOMAP	3.5.3
NOPAGE	NOPAGE	3.5.4
NOSYMT	NOSYMT	3.4.2
NOTGLOBAL	NOTGLOBAL [<symbol>] [,<symbol>] [, . . .]	3.3.7.4
PAGE	PAGE	3.5.4
PARTIAL	PARTIAL	3.3.7.1
PHASE	PHASE <level>,<name>	3.3.3
PROCEDURE	PROCEDURE <name>	3.3.1
PROGRAM	PROGRAM <base>	3.6.1
SEARCH[1]	SEARCH [<acnm>] [,<acnm>] [, . . .]	3.2.3
SHARE	SHARE <module name>,<module name>[, . . .]	3.3.6
SYMT[1]	SYMT	3.4.1
TASK	TASK <name>	3.3.2

Notes:

[1] These commands supported under DX10 only; they are not supported for TX systems.

APPENDIX D

OVERLAY LOADER ROUTINE (TXDS)

APPENDIX D

OVERLAY LOADER ROUTINE (TXDS)

D.1 DESCRIPTION

Whereas DX10 provides a Load Overlay supervisor call for user specified overlay loading, TXDS provides a user callable routine for overlay loading. The object for the overlay loader routine resides on the same diskette as the Link Editor and has the pathname DSCx:TXLOVL/SYS, where 'x' indicates the appropriate diskette drive. This routine must be included in the linked output by use of the following command:

```
INCLUDE DSCx:TXLOVL/SYS
```

The overlay loader routine loads overlays from the same program file as the task itself was loaded and performs the necessary relocation of the overlay. The overlay loader assumes that the LUNO equal to the task ID has been assigned to the program file (this is performed by the task loader). If the program file is not open, the overlay loader opens it.

The calling sequence within the user program is as follows:

When called:

R3 = Address at which the ovelay is to be loaded.

R4 = ID of the overlay to be loaded.

Call:

```
BLWP @L$$OLD
```

Returned Values:

R0 = Completion code. If zero, no errors. If not equal to zero, the rightmost byte contains a TX990 I/O error code.

APPENDIX E

LINK EDITOR CONDITION CODES UNDER DX10

APPENDIX E

LINK EDITOR CONDITION CODES UNDER DX10

E-1 DESCRIPTION

When the Link Editor is executed through an SCI batch stream or a command procedure, a condition code is returned as the value of synonym $$CC. The possible values of $$CC are interpreted as follows:

0	— No errors or warnings
4000	— One or more warnings
8000	— One or more errors
C000	— Link Editor aborted (I/O error, end action, syntax error, . . .)

For more information about condition codes, see the *DX10 Operating System Release 3 Reference Manual, Volume V, System Programming Guide,* part number 946260-9705.

APPENDIX F

OBJECT RECORD FORMAT AND TAGS

APPENDIX F

OBJECT RECORD FORMAT AND TAGS

F.1 DESCRIPTION

Table F-1 shows the object code tags and field values associated with each tag. Unless otherwise noted the size of each field is either four characters (ASCII object format) or four binary digits (compressed object format). Where noted by "(int)", the field is <int> number of characters in length (in both ASCII and compressed formats).

Table F-1. Object Record Format and Tags

TAG	FIELD 1	FIELD 2	FIELD 3
***	MODULE DEFINITION	–	–
0	PSEG LENGTH	PROGRAM ID (8)	–
M	DSEG LENGTH	$DATA	0000
M	BLANK COMMON LENGTH	$BLANK	0001
M	CSEG LENGTH	COMMON NAME (6)	COMMON #
M	CBSEG LENGTH	$CBSEG	CBSEG #
***	ENTRY POINT DEFINITION		
1	ABSOLUTE ADDRESS	–	–
2	P-R ADDRESS	–	–
***	LOAD ADDRESS		
9	ABSOLUTE ADDRESS	–	–
A	P-R ADDRESS	–	–
S	D-R ADDRESS	–	–
P	C-R ADDRESS	COMMON OR CBSEG #	–
***	DATA		
B	ABSOLUTE VALUE	–	–
C	P-R ADDRESS	–	–
T	D-R ADDRESS	–	–
N	C-R ADDRESS	COMMON OR CBSEG #	–
R . . .			
***	EXTERNAL DEFINITIONS		
6	ABSOLUTE VALUE	SYMBOL (6)	–
5	P-R ADDRESS	SYMBOL (6)	–
W	D-R/C-R ADDRESS	SYMBOL (6)	COMMON #
***	EXTERNAL REFERENCES		
3	P-R ADDRESS OF CHAIN	SYMBOL (6)	–
4	ABSOLUTE ADDRESS OF CHAIN	SYMBOL (6)	–
X	D-R/C-R ADDRESS OF CHAIN	SYMBOL (6)	COMMON #
E	SYMBOL INDEX NUMBER	ABSOLUTE OFFSET	

Table F-1. Object Record Format and Tags (Continued)

TAG	FIELD 1	FIELD 2	FIELD 3
***	SYMBOL DEFINITIONS		
G	P-R ADDRESS	SYMBOL (6)	–
H	ABSOLUTE VALUE	SYMBOL (6)	–
J	D-R/C-R ADDRESS	SYMBOL (6)	COMMON #
***	FORCE EXTERNAL LINK		
U	0000	SYMBOL (6)	–
***	SECONDARY EXTERNAL REFERENCE		
V	P-R ADDRESS OF CHAIN ENTRY	SYMBOL (6)	–
Y	ABSOLUTE ADDRESS OF CHAIN	SYMBOL (6)	–
Z	D-R/C-R ADDRESS OF CHAIN	SYMBOL (6)	COMMON #
***	CHECK SUM		
7	VALUE	–	–
***	IGNORE CHECK SUM		
8	ANY VALUE	–	–
***	LOAD BIAS		
D	ABSOLUTE ADDRESS	–	–
***	END OF RECORD		
F	–	–	–
***	REPEAT COUNT	Word-	
R	Abs. VALUE	REPEAT COUNT	–
***	PROGRAM ID (?)		
I	P-R ADDRESS	PROGRAM ID (8)	–
***	COBOL SEGMENT REFERENCE		
Q	RECORD OFFSET	CBSEG #	
***			

ALPHABETICAL INDEX

ALPHABETICAL INDEX

INTRODUCTION

The following index lists key words and concepts from the subject material of the manual together with the area(s) in the manual that supply major coverage of the listed concept. The numbers along the right side of the listing reference the following manual areas:

- Sections - References to Sections of the manual appear as "Section x" with the symbol x representing any numeric quantity.

- Appendixes - References to Appendixes of the manual appear as "Appendix y" with the symbol y representing any capital letter.

- Paragraphs - References to paragraphs of the manual appear as a series of alphanumeric or numeric characters punctuated with decimal points. Only the first character of the string may be a letter: all subsequent characters are numbers. The first character refers to the section or appendix of the manual in which the paragraph is found.

- Tables - References to tables in the manual are represented by the capital letter T followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the table). The second character is followed by a dash (-) and a number:

  Tx-yy

- Figures - References to figures in the manual are represented by the capital letter F followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the figure). The second character is followed by a dash (-) and a number:

  Fx-yy

- Other entries in the Index - References to other entries in the index are preceded by the word "See" followed by the referenced entry.

FOLD

FIRST CLASS
PERMIT NO. 7284
DALLAS, TEXAS

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

TEXAS INSTRUMENTS INCORPORATED
DIGITAL SYSTEMS DIVISION

P.O. BOX 2909 · AUSTIN, TEXAS 78769

ATTN: TECHNICAL PUBLICATIONS
MS 2146

FOLD

CUT ALONG LINE

USER'S RESPONSE SHEET

Manual Title: Model 990 Computer Link Editor Reference Manual (949617-9701)

Manual Date: 15 March 1978 Date of This Letter: ____________

User's Name: ____________ Telephone: ____________

Company: ____________ Office/Department: ____________

Street Address: ____________

City/State/Zip Code: ____________

Please list any discrepancy found in this manual by page, paragraph, figure, or table number in the following space. If there are any other suggestions that you wish to make, feel free to include them. Thank you.

Location in Manual Comment/Suggestion

NO POSTAGE NECESSARY IF MAILED IN U.S.A.
FOLD ON TWO LINES (LOCATED ON REVERSE SIDE), TAPE AND MAIL

TEXAS INSTRUMENTS
INCORPORATED
DIGITAL SYSTEMS DIVISION
POST OFFICE BOX 2909 AUSTIN, TEXAS 78769