As you are now the owner of this document which should have come to you for free, please consider making a donation of £1 or more for the upkeep of the (Radar) website which holds this document. I give my time for free, but it costs me money to bring this document to you. You can donate here https://blunham.com/Misc/Texas

Many thanks.

Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Misc/Texas). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you.

Colin Hinson
In the village of Blunham, Bedfordshire.

SPDU37C

Link Editor

User's Guide

TEXAS INSTRUMENTS

Link Editor User's Guide

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes in the devices or the device specifications identified in this publication without notice. TI advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current.

TI warrants performance of its semiconductor products, including SNJ and SMJ devices, to current specifications in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed.

In the absence of written agreement to the contrary, TI assumes no liability for TI applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Contents

Section		*Page*
1	**Introduction**	**1-1**
1.1	Description	1-2
1.2	Program Definition	1-2
2	**Link Editor Files**	**2-1**
2.1	Link Control File	2-2
2.2	Object Modules	2-2
2.3	Libraries	2-2
2.4	Linked Output File	2-3
2.5	Listing File	2-3
3	**Linker Commands**	**3-1**
3.1	Entering a Command	3-2
3.2	Linker Command Set	3-2
3.3	Individual Command Descriptions	3-5
4	**Linking Examples**	**4-1**
4.1	Simple Linking	4-2
4.2	ROM/RAM Partitioning	4-5
4.3	Partial Linking	4-6
4.4	Library Creation	4-10
5	**Link Editor Error Messages**	**5-1**
A	**Glossary**	**A-1**

Illustrations

Figure *Page*

4-1. Source for Module Alpha 4-1
4-2. Source for Module Beta 4-2
4-3. Source for Module Gamma 4-2
4-4. Listing File for a Simple Link 4-4
4-5. Listing File for ROM/RAM Partitioning 4-6
4-6. Listing and Object Files for a Partial Link 4-8
4-7. Listing and Object Files for Relinking the Partial Link Output 4-9
4-8. Source File for Sequential Library Creation 4-11

Tables

Table *Page*

3-1. Linker Syntax Symbols 3-2
3-2. Linker Command Set Summary 3-3

1. Introduction

The Link Editor combines separately generated object modules with associated procedures and overlays to form a single, linked, relocatable object module that can be installed and executed on various computer systems. The object code is generated by assemblers supplied with the TMS7000, TMS99000, or TMS320 software development systems. The link editor is currently available for the TI990 (DX10), VAX (VMS and Berkeley UNIX 4.1 and 4.2), and TI/IBM PC (MS/PC-DOS) operating systems.

This manual describes the Link Editor, its files and control commands, and gives examples of various linking procedures. Included in this document are the following major topics:

- Introduction (Section 1)
  - Description
  - Program definition (phase and task)
- Link Editor Files (Section 2)
  - Link control file
  - Object modules
  - Libraries
  - Linked output file
  - Listing file
- Linker Commands (Section 3)
  - Entering a command
  - Command set summary (listed according to function)
  - Individual command descriptions (alphabetized)
- Linking Examples (Section 4)
  - Simple link
  - RAM/ROM partitioning
  - Partial link
  - Library creation
- Link Editor Error Messages (Section 5)
- Glossary (Appendix A)

1.1 Description

The Link Editor provides symbol resolution for external references and definitions created by the REF and DEF assembler directives. Without this function, all modules would have to be compiled or assembled at once, and modules written in different languages could not be mixed.

The Link Editor builds a list of symbols from the REF tags in the object modules that are included in the linking process. The Link Editor then resolves references by matching DEF tag symbols with the REF tags and inserting the correct values for these symbols in the linked object code.

The Link Editor can position the three defined segments (program, data, and common) to prescribed boundaries for eventual ROM/RAM partitioning. Program, data and common segments are defined by the PSEG, DSEG, and CSEG assembler directives, respectively. If these directives are not used, the entire object module is tagged as a program segment.

When PSEG, DSEG, and CSEG tags are encountered in the included modules, the Link Editor reorganizes segments from each module into three segments in the linked output. The first segment contains the PSEGs of all included modules, the second segment contains the DSEGs, and the third segment the CSEGs of all included modules. The beginning location for each segment can be user-defined.

The Link Editor also allows overlays and procedure/task segmentation. However, if the system being used loads only one module at a time, procedure/task segmentation and overlays cannot be used because they produce multiple output modules.

1.2 Program Definition

To use the Link Editor, each program must be defined as a phase or a task. Below are the definitions of each.

Phase The smallest functional unit that can be loaded as a logical entity during execution in an overlay structure.

Each phase is identified by a name and a level number. The root phase is at level 0 and is that portion of the program that must remain memory resident. Other phases (level 1 and above) that do not have to be simultaneously memory-resident can overlay each other.

Task A complete program containing both variable data and executable code or the variable data portion of a program (for procedure/task segmentation).

2. Link Editor Files

Executing the Link Editor utility begins by accessing the Linker and then responding to prompts for the link control file, linked output file, and listing file. The Link Editor utility uses the following five files in the linking process:

- Link control file
- Object modules
- Libraries
- Linked output file
- Listing file

Each file is given a pathname so that when that pathname is entered, the Link Editor can search for that file. The pathnames for the link control file, object modules, declared libraries, the linked output file, and the listing file are in the listing file. An example of pathname structure (default value) for the link control file is given for four of the operating systems currently available for the Link Editor.

Pathname	System
DS01.SIMPLE.CTL	990/DX10
[PROJECT.MACK]SEGMENT.CON	VAX/VMS
\UFR\PROJECT\MACK\SEGMENT.CON	VAX/Berkeley UNIX
A:PARTIAL.CTL	TI/IBM PC (MS/PC-DOS)

Each of the link editor files is described in the succeeding subsections.

2.1 Link Control File

The link control file is an input file that controls the operation of the Link Editor. This file contains a set of link control commands called a control stream which defines the modules to be linked and how they are to be linked. The Link Editor links the object modules in the order specified by the linker commands. See Table 3-2 for a summary of all the linker commands.

The link control file must be created ahead of time. Entering a pathname instructs the editor to look for a file containing the necessary control commands.

2.2 Object Modules

Object modules are the input programs that are to be linked together. They are contained in files and must consist of either ASCII or compressed 990-tagged object code. The ASCII 990-tagged object code is the type of code generated by assemblers supplied with the TMS7000, TMS99000, and TMS320 Software Development Systems. The object code consists of ASCII tags followed by data fields. The following documents contain a description of object code format:

- Texas Instruments TMS7000 Assembly Language Programmer's Guide (SPNU002B)
- TMS99000 Assembly Language Programmer's Guide (SPOU001B)
- TMS32010 Assembly Language Programmer's Guide (SPRU002B)
- TMS32020 User's Guide (SPRU003)

As the Link Editor finishes writing out an object module, it names the module and gives the number of object records it contains. When the link terminates normally, the last line written reads '*** LINKING COMPLETED'. The date and time at the end of the link are printed on the last line. The date and time captured at the beginning of the link are printed at the top of every page and on the last card of every module in the linked object.

Object modules can be explicitly user-defined with the INCLUDE command in the control file, or automatically included by the Link Editor as a result of a search for unresolved references.

2.3 Libraries

Libraries are directories or files containing collections of object modules. An object library may be either random or sequential. A random library is a directory of object modules in separate files, whereas a sequential library is a file containing one or more object modules concatenated together. See Section 4.4 for examples of library creation.

Libraries are used to automatically resolve the REF and DEF tag symbols between object modules specified in INCLUDE commands.

Two types of symbol resolution are implemented:

- Automatic symbol resolution by default (the AUTO command) when the END command is detected in the control file unless the NOAUTO command has been used.

- Symbol resolution at a user-defined point in the linking process when a SEARCH or FIND command is used. The SEARCH command is used with random libraries and the FIND command with sequential libraries.

Libraries defined by the LIBRARY command are searched in the same order they are defined. Any additional unresolved references created by modules to satisfy references are also resolved automatically. Automatic symbol resolution still occurs at the end of the linking process for any remaining unresolved references unless a NOAUTO command is in the control file.

2.4 Linked Output File

The linked output file is an 80-character output file containing the 990-tagged object format load module in the "LINKED OUTPUT" file. This load module appears in ASCII or compressed format, depending on the use of the FORMAT command in the object link control file. The response to the linked output file name specifies the destination of the load module.

2.5 Listing File

The listing file consists of a listing that includes the control stream and a link map that lists the modules with their origins and lengths. The link map consists of the following four sections:

1) Individual constituent object modules
2) Common segments
3) Symbols (external)
4) Unresolved references (identified even if the NOMAP option has been selected).

The response to the listing file access name specifies the destination of the listing generated during the link edit. The pathnames for the control file, the listing file, the linked object file, and declared libraries are in the listing file. Messages are listed for detected errors in the listing file.

The Link Editor creates two temporary files on the work file disk. Therefore, sufficient space for two disk or diskette files must be available.

3. Linker Commands

Link control commands define the modules to be linked and how they are linked. This section gives some rules for entering a command in the link control file.

A command set summary of all the linker commands, arranged according to function, is provided for easy reference. Each command in the summary table is next described individually. Linker syntax and example(s) are also given for each command. The commands are listed in alphabetical order.

3.1 Entering a Command

When entering a command in the control file, these rules should be followed:

- Either the entire command or only the first four characters may be specified.
- At least one space must separate the command from its parameters.
- Comments may be entered either on a separate line or following the command parameters.
- All comments must be preceded by a semicolon (;).
- The command must be contained within the first 72 characters of the line.

3.2 Linker Command Set

Table 3-1 lists the symbols used in the syntax definitions of the linker commands.

Table 3-1. Linker Syntax Symbols

SYMBOL	MEANING
< >	User-defined parameters.
[ ]	Optional parameters. They may be omitted.
{ }	Alternative parameters, one of which must be entered.
...	The preceding parameter may be repeated.
( )	Indicates "contents of".
<acnm>	An access name for a file or library must be entered for the parameter.
<base>	The starting location of a segment, expressed as either a decimal or hexadecimal number up to five digits in length.
<level>	The level of a phase.
<name>	The name of a specified area. Consists of one to eight alphanumeric characters, the first of which must be alphabetic.
(<name>)	The name of a member in a library.
<value>	The number of lines, between 16 and 60, to be printed on a page. Replaces the default value of 60.
>	Represents hexadecimal, as does also a leading zero.
	Words shown in capital letters and special characters not listed here must be entered as shown.

The link command set summary of Table 3-2 is arranged according to function and alphabetized within each functional grouping. Of the four groups, the first group consists of basic commands that are required to perform basic Link Editor functions. The second group consists of ROM/RAM partitioning commands. The third group includes those miscellaneous commands that perform auxiliary link editor functions, such as specifying default conditions and procedure/task segmentation. The fourth group consists of the partial link commands.

Table 3-2. Linker Command Set Summary

BASIC COMMANDS	
Command	**Function**
END	Indicates the end of the control stream. This is a required command.
FIND	Specifies a search of only sequential libraries for unresolved references at this point in the control stream.
FORMAT	Defines the format of the linked output module as ASCII or COMPRESSED code. The default is ASCII object code.
INCLUDE	Defines one or more modules to be included in the linking process. At least one INCLUDE command is required in each control stream.
LIBRARY	Defines a random library directory.
PHASE	Defines the level and name of a phase in a program. Either the PHASE or the TASK command must appear in each control stream. Multiple phases are allowed when overlays are used.
SEARCH	Specifies a search of defined random libraries for unresolved references at this point in the control stream.
TASK	Defines a phase to be installed and executed as a task or stand-alone program. A name is assigned the task.
ROM/RAM PARTITIONING COMMANDS	
Command	**Function**
ALLOCATE	Controls the relative positioning of the program, data, and common segments (PSEG, DSEG, and CSEG assembler directives, respectively).
COMMON	Specifies the starting location of the common segment in the linked output.
DATA	Specifies the starting location of the data segment in the linked output.
PROGRAM	Specifies the starting location of the program segment in the linked output.

Table 3-2. Linker Command Set Summary (Concluded)

AUXILIARY FUNCTION COMMANDS	
Command	**Function**
ADJUST	Aligns a phase or a module within a phase on a specified boundary.
AUTO	Specifies automatic symbol resolution at the end of the control stream (default condition).
DUMMY	Suppresses generation of the linked output file. Useful for error identification or when only a listing file is required.
ENTRY	Specifies a symbol for an entry tag to be produced.
NOAUTO	Inhibits automatic symbol resolution, allowing the user to explicitly control library searching for unresolved references.
NOMAP	Suppresses the output of the link map listing by omitting the module, common, and symbol maps from the listing.
NOPAGE	Inhibits page ejects between the link maps of each phase.
NOSYMT	Omits symbol tables from included modules in the linked output file (default condition).
PAGE	Causes page ejects between link maps for each phase (default condition).
PROCEDURE	Defines a phase of the link edit structure which can be installed as a procedure. Used for procedure/task segmentation only. An alternate version of this command can be used to support levels 1 and 2.
REPLACE	Replaces one external symbol name for another in the next object file read in.
SYMT	Includes symbol tables in linked output files.
PARTIAL LINK COMMANDS	
Command	**Function**
ALLGLOBAL	Declares all external definitions in included modules as global symbols for subsequent relinking (default condition).
GLOBAL	Identifies the symbols defined in included modules to be processed as global symbols for subsequent relinking.
NOTGLOBAL	Declares either specified externally defined symbols or all externally defined symbols in included modules as local symbols.
PARTIAL	Performs a partial link and outputs either ASCII or compressed object code. The output of a partial link must be linked again without the PARTIAL command before the program can be loaded and executed.

3.3 Individual Command Descriptions

Each command in the linker command set summary is described in the following pages. Information, such as linker syntax, a description, and example(s), is given for each command. The commands are listed in alphabetical order.

Syntax

ADJUST [<n>]

where <n> = a decimal number less than 16 specifying a power-of-two bytes. A value greater than 15 causes an error. When the parameter is omitted or equal to zero, alignment is on the next word boundary.

Description

The ADJUST command specifies the alignment of a phase or of a module within a phase on a specified boundary.

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.

If the ADJUST command follows a PHASE command but precedes all INCLUDE commands in the phase, the effect is the same as above. 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.

Syntax

ALLGLOBAL

Description

The ALLGLOBAL (partial linking) command declares all external definitions in included modules as global symbols. ALLGLOBAL is a default condition.

Global symbols are externally defined in the linked output module and therefore may be re-linked in a subsequent linking process.

Syntax

ALLOCATE

Description

The ALLOCATE command controls the relative positioning of program, data, and common segments (PSEG, DSEG, and CSEG directives, respectively). ALLOCATE has no parameters.

ALLOCATE directs the Link Editor to reserve space for all outstanding data and common segments as if no more object modules were to be included in the link. The primary purpose of the ALLOCATE command is to aid the user in sharing non-reentrant procedures between different tasks.

The ALLOCATE command only works if all read/write data is contained in data segments (DSEGs) or common segments (CSEGs).

Syntax

AUTO

Description

The AUTO command specifies automatic symbol resolution using defined libraries at the end of the linking process. The AUTO command has no parameters and is optional. It is the default condition.

Syntax

COMMON {<base>[,<name>] [,<name>]...}

where <base> = the starting location of the common segment. It can be expressed as either a decimal or hexadecimal number up to five digits in length.

<name> = the name of the common segment. Any unnamed common segment begins after the last data area encountered. The commons are allocated in the order in which the definitions appear in the object module.

Description

The COMMON command defines the starting address for the specified common segment (CSEG). Commons that are loaded at the specified address must be specifically identified within this command. The COMMON command is only valid when used with the PROGRAM command and is ignored if used alone.

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 address. The COMMON command cannot be used in partial links.

Example 1

```
COMMON    01000,COMA        Begin common COMA at location
                            >1000.
```

Example 2

```
COMM      >1000,COMA        Results are the same as the
                            preceding example.
```

Example 3

```
COMMON    COMA,COMB         Begin common COMB immediately
                            following COMA.
```

Example 4

```
COMM      4096,COMA,COMB    Results are the same as the
                            two preceding examples.
```

Syntax

DATA <base>

where <base> = the starting location of the data segment. It can be expressed as either a decimal or hexadecimal number up to five digits in length.

Description

The DATA command defines the absolute starting address for the data segment (DSEG) in the linked output. The DATA command is only valid when used with the PROGRAM command and is ignored if used alone.

The DATA command may appear more than once in the control stream, but 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 of the corresponding program area.

The DATA command cannot be used in partial links.

Example 1

```
DATA   01000   Begin data segment at location >1000.
```

Example 2

```
DATA   4096    Same as the preceding example.
```

Syntax DUMMY

Description The DUMMY command supresses generation of the linked output file. This command is useful for error identification or when only a listing file is needed. DUMMY has no parameters.

Syntax

END

Description

The END command specifies the end of the control stream. The command is required in every control stream.

Syntax

FORMAT {ASCII,COMPRESSED}

Description

The FORMAT command defines the format of the linked output module.

The format specified may be either ASCII or COMPRESSED object code. In ASCII format, each integer value in the object is represented as a four-byte character string. ASCII format is also called 990-tagged object format and is the default condition. Compressed format is more efficient to use since each integer value is represented as a two-byte word.

Syntax

GLOBAL [symbolname][,symbolname]...

where symbolname = a symbol which is to be processed as a global symbol. It is defined at assembly time and consists of six characters or less, the first of which must be alphabetic.

Description

The GLOBAL command is a partial linking command, identifying the symbols defined in included modules to be processed as global symbols. Global symbols are externally defined in the output module that may be relinked.

Each parameter specifies a symbol that is to be processed as a global symbol. The command may include several parameters and may appear more than once in the command stream. If no parameters are specified, the command functions as an ALLGLOBAL command.

Symbols defined by the GLOBAL command are not affected by the NOTGLOBAL command (no parameters) that declares all symbols to be local.

Syntax

INCLUDE {<acnm>[,<acnm>]...,(<name>)[,(<name>)]...}

where <acnm> = the access name of a file containing the object module(s) to be included in the linking process.

(<name>) = a member in a library.

Description

The INCLUDE command specifies modules to be included in the linking process. This command is required in the control stream. More than one INCLUDE command may be used as needed.

A PROCEDURE, TASK, or PHASE command must precede the first INCLUDE command.

If the <name> parameter is used, enclose only the file name or module name of the object modules (rather than the entire access name) in parentheses. The specified <name> must be of a file contained in a defined random library. The Link Editor searches the defined libraries for the specified module.

If no parameters are given, in-line text format (not compressed) object code is assumed. The in-line object (see Example 3) is delimited by either end-of-file or by a record with '/*' in columns one and two. This method is suitable, for example, when the control file is read in from a card reader (in which case, end-of-file is denoted by a '/*' card).

Example 1

```
INCLUDE  (X)          Search defined random libraries for
                      a file named X and include the
                      module(s) in that file.
```

Example 2

```
INCLUDE  TEST.MPO     Include the module TEST.MPO from the
                      default directory on a PC/MS-DOS
                      system.
```

Example 3

```
INCLUDE
K006CCARTMOND50020LBL2B150021LBL2B240000LBL2C240000LBL2D2A00207F1E7F
BCE26BCE26BFE80E00000000BFE80E00010000BCE1BB0201B0388B4802B48A97F1E1F
BFFEEBCE27BCE50BCE04BCE05BCE01B567BB568CBCE00BCE0FBCE1FB807AB81A87F048F
:       CARTMOND   4/10/85  10:53:14    ASM32020 PC 1.0 85.092
/*
```

Syntax

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

where <acnm> = the access name of the directory that is to be defined as a library.

Description

The LIBRARY command defines random library directories. Random libraries must consist of a directory, and the files in the directory must contain 990-tagged object modules. Sequential libraries, consisting of a sequential file of object modules, are indicated using the FIND command.

Example 1

```
LIBR    VOL1.AOBJ,VOL1.BOBJ     Define directories AOBJ and
                                 BOBJ as random libraries
                                 using DX10 pathnames.
```

Example 2

```
LIBR    A:*.EXT                  Define drive A: as a random
                                 library of files with
                                 extension .EXT on a
                                 PC/MS-DOS system.
```

Syntax

NOAUTO

Description

The NOAUTO command inhibits automatic symbol resolution at the end of the linking process. This command allows the user to explicitly control library searching for unresolved references through use of the SEARCH and FIND commands. NOAUTO has no parameters.

Syntax

NOMAP

Description

The NOMAP command specifies that the module, common, and symbol maps are to be omitted from the listing. This gives some improvement in terms of speed and number of symbols that can be processed. The following information is still printed on the listing file:

- Length of task and procedure(s)
- Unresolved references
- Release number of the Link Editor

NOMAP must appear before any PHASE or TASK commands are used.

Syntax

NOPAGE

Description

The NOPAGE command sets no page ejects between the link maps for each phase. New pages are started for the listing of the first phase and when the number of lines per page has been exceeded.

Syntax

NOSYMT

Description

The NOSYMT command omits symbol tables from included modules in the linked output file. This provides for more compact object code but does not allow symbolic debugging.

The NOSYMT command may appear anywhere in the control file. However, if an overlay structure is used, the NOSYMT command must appear in the root phase (phase 0).

NOSYMT is the default option and is the inverse of SYMT.

Syntax

NOTGLOBAL [symbolname][,symbolname]...

where symbolname = a symbol which is to be processed as a local symbol. It is defined at assembly time and consists of six characters or less, the first of which must be alphabetic.

Description

The NOTGLOBAL command is a partial linking command, declaring that either specified externally defined symbols or all externally defined symbols in the included modules are to be processed as local (not global) symbols.

Local symbols are not externally defined in the partially linked output module and thus can only be referenced by modules included in the current partial link.

The command may include several parameters and may appear more than once in the command stream. If no parameters are specified, all symbols are processed as local, except those specified in the GLOBAL command.

Syntax

PAGE [value]

where value = the number of lines to be printed on a page, replacing the default value of 60. The value parameter is optional, but when present, the value must be between 16 and 60.

Description

The PAGE command causes page ejects to separate the beginning of each link map for each phase. This is the default condition.

Syntax

PARTIAL

Description

The PARTIAL command performs a partial link and outputs either ASCII or compressed object code. The output of a partial link is not executable and must be linked again without the PARTIAL directive before the program can be loaded and executed.

The PARTIAL command causes the Link Editor to do the following:

1) Resolve all external references defined by any module included in the partial link.

2) Retain all entry points in the partial link as an entry in the output (subject to GLOBAL, NOTGLOBAL, ALLGLOBAL commands).

3) Retain the common tags and update common numbers.

4) Output one data section that is the total of all input data sections.

Partial linking is allowed for single phases only, and the control stream must contain either a TASK or PHASE 0 command. If partial linking of overlays is required, each phase must be partially linked separately as a phase 0. The phase level and name may be redefined in subsequent links. The following commands are invalid with partial links: ALLOCATE, PROGRAM, DATA, COMMON, and DUMMY.

Syntax

PHASE <level>,<name>

where <level> = the level of the phase. Levels specified greater than zero can be used for overlay structures only. Level 0 defines the root (memory-resident) phase. Each subsequent PHASE command defines the level and name of an overlay.

<name> = the name of the phase. It consists of one to eight alphanumeric characters, the first of which must be alphabetic. The name supplied becomes the IDT name, placed on the last card of the object module produced and on the identification fields of ASCII-formatted object records.

Description

The PHASE command defines the level and name of a phase in a program.

PHASE 0 and TASK commands are logically identical; one and only one of these two commands must appear in each control stream.

The Link Editor produces an output module for each phase of the program. PHASE commands are followed by INCLUDE commands that define the modules included in the phase. Multiple phases are allowed when overlays are used.

Example 1

```
PHASE    0,MAIN          Define phase MAIN at level 0.
```

Example 2

```
PHAS     2,DISK          Define phase DISK at level 2.
```

Syntax

PROCEDURE {<name>,<level,name>}

where <name> = the identifier of the procedure to be used. The parameter consists of one to eight alphanumeric characters, the first of which must be alphabetic.

<level> = the level of the phase.

Description

The PROCEDURE command provides procedure/task segmentation by defining a phase of the link edit structure which can be installed as a procedure (a re-entrant procedure may be shared among several tasks). The name supplied becomes the IDT name, placed on the last record of the object module produced and on the identification field of ASCII-formatted object records. This command was designed for the DX10, but can be used on other systems, particularly in ROM/RAM partitioning for generating load modules with a level of root phase 0.

When used, the PROCEDURE command must precede the TASK command, all PHASE commands, and the INCLUDE command that defines the procedure module.

The PROCEDURE command is used with the INCLUDE commmand 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. Procedures contain the program segment (PSEG), which may be the entire program but is usually only the executable code and read-only data.

A generalization of the standard PROCEDURE command is supported for levels 1 and 2. In place of a single first procedure, any number of other level-one procedures can be defined, any of which can be resident in memory at a given time under the user's control. The length of the first procedure area is the maximum of the lengths of the individual level-one modules. Analogous properties apply to second-level PROCEDURES. Modules brought in by automatic call to satisfy references in any procedure module will be placed in the root.

Example 1

```
PROCEDURE   FORLIB      Define procedure FORLIB.
```

Example 2

```
PROC        RUNLIB      Define procedure RUNLIB.
```

Example 3

```
PROCEDURE   2,FILEMG    Define a procedure FILMG at level 2.
```

Syntax

PROGRAM <base>

where <base> = the starting location of the program segment. It can be expressed as a decimal or hexadecimal number up to five digits in length.

Description

The PROGRAM command defines the absolute starting address for the program segment (PSEG) in the linked output.

The PROGRAM command may be used more than once. The first PROGRAM command must appear before the first INCLUDE command. Use of the PROGRAM command by itself or with the DATA and COMMON commands causes the linked output to be loaded at the specified address (base).

Example 1

```
PROGRAM     01F00       Begin program segment at location
                        >1F00.
```

Example 2

```
PROG        >1F00       Same as the preceding example.
```

Example 3

```
PROG        7936        Begin program segment at location
                        7936 (>1F00).
```

Syntax

REPLACE <oldsym(newsym)>[,<oldsym(newsym)>]...

where oldsym = the currently existing external symbol representing a reference, definition, or common name.

(newsym) = the new external symbol to replace the oldsym.

Description

The REPLACE command specifies that in the next file read in, each occurrence of 'oldsym' as an external symbol is replaced by 'newsym'. The command applies to every module in a file containing multiple modules. It applies only to the first file in an INCLUDE command list. If the command immediately precedes a FIND, SEARCH, or END command, it still applies to the next single file read in.

If 'oldsym' is $DATA and an affected module contains a DSEG, the linker converts the DSEG to a common with the name 'newsym'. This means that no data segment is identified in the listing, and if other instances of the common name occur, the common may be extended in length or promoted (moved up to a lower-numbered phase).

Note that data segments can be shared by using the REPLACE command to convert them to a common. Appropriately used, this permits a module to share a data segment in an ancestor phase and places no restrictions on the order of definition of segments with different lengths.

Syntax

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

where <acnm> = the access name of random libraries to be searched. The order of these access names determines the order of the search. If no <acnm>s are specified, the libraries defined by the LIBRARY commands define the search ordering.

Description

The SEARCH command directs the Link Editor to search for unresolved references at any point in the control stream.

If a SEARCH command is given in a phase other than the TASK or PHASE 0 phase, searching is performed only for symbols that are unresolved in that phase. Unresolved references that were established in or promoted to other phases are ignored.

A SEARCH command in a TASK phase causes searching to be done for every phase (for the given phase and all its descendant and previous phases). The only way the SEARCH command can be applied to more than one phase is by re-entering a phase defined earlier. This is permitted only for the task phase and for the purpose of doing SEARCHes and FINDs.

Example 1

```
SEARCH                  Search defined libraries for
                        unresolved references.
```

Example 2

```
SEARCH    A:*.EXE       Search drive A: as a library of
                        files with extension .EXE on a
                        PC/MS-DOS system.
```

Syntax

SYMT

Description

The SYMT command causes the Link Editor to include symbol tables in the linked output file when the linker input files contain such symbols. These symbols were provided in the assembler as a result of selecting the SYMLST option (see the OPTION directive in the appropriate Assembly Language Programmer's Guide). Although symbol tables make the linked module larger, they are useful for symbolic debugging.

SYMT is the inverse of the NOSYMT option.

Syntax

TASK [<name>]

where <name> = the identifier of the task module. The <name> can have up to eight characters. The name supplied becomes the IDT name, placed on the last record of the object module produced and on the ID fields of ASCII-formatted records. If the parameter is omitted, the IDT name of the first included module is used as the task name.

Description

The TASK command defines a phase that can be installed and executed as a task or standalone program, and assigns a name to the task.

A task is either a complete program, containing both variable data and executable code, or it is the variable data portion of a program (procedure/task segmentation). The TASK and PHASE 0 commands are logically identical; one and only one of these two commands must appear in each control stream.

When task/procedure segmentation is used, the TASK command must follow all PROCEDURE commands and precede all PHASE and INCLUDE commands that define the task module. The TASK command can be given after overlays have been defined (to re-enter the root phase).

Example 1

```
TASK      FORPRG      Define task named FORPRG.
```

Example 2

```
TASK                  Define task and assign it the IDT name
                      of the first included module.
```

4. Linking Examples

Examples showing how and when to use the link control commands are provided in this section. Among the examples are a simple link (Section 4.1), ROM/RAM partitioning (Section 4.2), and a partial link (Section 4.3). In addition, examples are given for creating random and sequential libraries (Section 4.4).

Three separately assembled modules, ALPHA, BETA, and GAMMA, are to be linked together. Figure 4-1, Figure 4-2, and Figure 4-3 contain the source for each module. The first example assumes that each module is contained in a separate file named ALPHA, BETA, and GAMMA, respectively, and that the three files are listed under a user directory named OBJ on the diskette in drive DS01 on a DX10 operating system. The second and third examples are similar, but base their file access on the VAX/VMS and TI/IBM PC (MS/PC-DOS) operating systems, respectively.

TMS99000 assembly language source is used in the three figures, but the code could easily have been TMS7000 or TMS320 assembly language. All three assemblers produce 990-tagged object code that the Link Editor requires as input.

```
            IDT     'ALPHA'
*
            REF     BETA,GAMMA
            PSEG
ALPHA       EQU     $
            SETO    R0
LOOP        MOV     R0,@CVAR
            BL      @BETA
            BL      @GAMMA
            DEC     R0
            JNE     LOOP
            IDLE
            PEND
*
            DSEG
WRKSPC      BSS     16*2
            DEND
*
            CSEG    'COMDAT'
CVAR        BSS     2
            CEND
            END
```

Figure 4-1. Source for Module Alpha

```
            IDT       'BETA'
*
            DEF       BETA
            PSEG
BETA        EQU       $
            MOV       @CVAR,@MYVAR
            RT
            PEND
*
            DSEG
MYVAR       BSS       2
            DEND
*
            CSEG      'COMDAT'
CVAR        BSS       2
            CEND
*
            END
```

Figure 4-2. Source for Module Beta

```
            IDT       'GAMMA'
*
            DEF       GAMMA
            PSEG
GAMMA       EQU       $
            MOV       @CVAR,@MYVAR
            RT
            PEND
*
            DSEG
MYVAR       BSS       2
            DEND
*
            CSEG      'COMDAT'
CVAR        BSS       2
            CEND
*
            END
```

Figure 4-3. Source for Module Gamma

4.1 Simple Linking

Every control stream must contain either a TASK or PHASE 0 command to define the name of the program being linked. In addition, the control stream must contain one or more INCLUDE commands to define modules that are being linked. The control stream is terminated with an END command. The following is an example control stream on the DX10 operating system for linking the three example object modules generated for ALPHA, BETA, and GAMMA.

```
PHASE     0,SIMPLE
INCLUDE   DS01.OBJ.ALPHA
INCLUDE   DS01.OBJ.BETA
INCLUDE   DS01.OBJ.GAMMA
END
```

The three modules may be specified in one INCLUDE command rather than with three separate commands. The directory containing the modules (DS01.OBJ) may also be defined as a library. When this is done, only the file name (enclosed in parentheses) need be specified. The Link Editor searches the defined library for the required files. An example of a control stream using the INCLUDE command is as follows:

```
PHASE     0,SIMPLE
LIBRARY   DS01.OBJ
INCLUDE   (ALPHA),(BETA),(GAMMA)
END
```

Since ALPHA references BETA and GAMMA, and the directory containing these modules has been defined as a library, ALPHA is the only module that must be specified in the INCLUDE command, as shown in the following example:

```
PHASE     0,SIMPLE
LIBR      DS01.OBJ
INCL      (ALPHA)
END
```

At the end of the control stream, the Link Editor automatically searches the defined library for unresolved references and includes the modules that satisfy the references in the linking process.

The Link Editor produces a listing of the linking process and writes it to a specified file. Figure 4-4 is an example of the listing file produced. The listing file for this example consists of three pages. The first page contains a copy of the link control stream. The second page lists the parameters used when the Link Editor was initialized (access names of the control file, linked output file, and listing file) and the format of the linked output. Since the FORMAT command was not included in the control stream, the default, ASCII, is used.

The third page contains the link map, which is generated to facilitate debugging. The link map lists the origins and lengths of the phase being linked, the modules included in the link, and any common segments. The origins are relative to the beginning of the phase. The order in which the included modules are linked is indicated by the number listed next to the module name. The link map also lists the symbols defined in the included modules, indicating the module in which the symbol is defined (number) and the resolved location of the symbol (value). An asterisk (*) preceding the symbol name indicates that the symbol is not referenced in the included modules. An asterisk to the right means the symbolic value is absolute.

```
DX10/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:15:21 PAGE 1
COMMAND LIST

       PHASE     0,SIMPLE
       INCLUDE   DS01.OBJ.ALPHA
       INCLUDE   DS01.OBJ.BETA
       INCLUDE   DS01.OBJ.GAMMA
       END

DX10/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:15:21 PAGE 2
LINK MAP
CONTROL FILE = DS01.SIMPLE.CON

LINKED OUTPUT FILE = DS01.SIMPLE.LOD

LIST FILE = DS01.SIMPLE.MAP

OUTPUT FORMAT = ASCII

DX10/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:15:21 PAGE 3

PHASE 0, SIMPLE   ORIGIN = 0000  LENGTH = 004A

MODULE  NO   ORIGIN   LENGTH     TYPE     DATE      TIME    CREATOR

ALPHA    1     0000     0014    INCLUDE  5/20/85   11:15:21  TMS990
$DATA    1     0024     0020
BETA     2     0014     0008    INCLUDE  5/20/85   11:15:21  TMS990
$DATA    2     0044     0002
GAMMA    3     001C     0008    INCLUDE  5/20/85   11:15:21  TMS990
$DATA    3     0046     0002

COMMON              NO          ORIGIN    LENGTH

COMDAT              3           0048      0002

                      D E F I N I T I O N S

NAME         VALUE NO   NAME     VALUE NO

*BETA        0014  2    *GAMMA   001C  3

****  LINKING COMPLETED     5/20/85   11:15:29
```

Figure 4-4. Listing File for a Simple Link

4.2 ROM/RAM Partitioning

Each example module has a program segment defined by the PSEG assembler directive, a data segment defined by the DSEG directive, and a common defined by the CSEG directive. Program segments generally contain instructions and nonvariable data (read only). Data segments generally contain variable data (read/write) and are labeled by the Link Editor as $DATA. Common segments contain variable data that may be shared by more than one module.

The Link Editor automatically reorganizes the output so that all the program segments of the included modules are together, followed by the data segments and then the common segments. The link control commands PROGRAM, DATA, and COMMON can be used to specify the beginning location of each output segment. These commands cannot be used with a PROCEDURE command or a PHASE command with a level greater than zero.

The following is an example of the control stream for a VAX/VMS operating system, which is used to partition the program and data segments into potential ROM and RAM locations.

```
PHASE       0,SEGMENT
PROGRAM     >1000
DATA        >2000
COMMON      >3000,COMDAT
INCLUDE     [PROJECT.MACK]ALPHA.MPO
INCLUDE     [PROJECT.MACK]BETA.MPO
INCLUDE     [PROJECT.MACK]GAMMA.MPO
END
```

The example assumes that location >1000 is in ROM and locations >2000 and >3000 are in RAM. This control stream causes the program segment of ALPHA to begin at location >1000, followed by the program segment of BETA, and then GAMMA. The data segment of ALPHA begins at location >2000 and is followed by the data segments of BETA and GAMMA. The common segment that is to be shared by all three modules begins at location >3000. Note that if the common segment is not specifically named in the COMMON command, the segment begins immediately following the last data segment.

Figure 4-5 contains the listing produced by this link. Use of the PROGRAM, DATA, and COMMON commands causes the phase length to be listed as zero and the origins to be listed as absolute locations. An asterisk(*) preceding the symbol name indicates that the symbol is not referenced in the included modules. An asterisk following the value of a symbol name indicates an absolute location. Linking absolute code generated by the assembler (AORG assembler directive) also causes the phase length to be listed as zero and the origins to be absolute locations.

```
VAX/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:24:46 PAGE 1
COMMAND LIST
        PHASE       0,SEGMENT
        PROGRAM     >1000
        DATA        >2000
        COMMON      >3000,COMDAT
        INCLUDE     [PROJECT.MACK]ALPHA.MPO
        INCLUDE     [PROJECT.MACK]BETA.MPO
        INCLUDE     [PROJECT.MACK]GAMMA.MPO
        END

VAX/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:24:46 PAGE 2
LINK MAP
CONTROL FILE = [PROJECT.MACK]SEGMENT.CON

LINKED OUTPUT FILE = [PROJECT.MACK]SEGMENT.LOD

LIST FILE = [PROJECT.MACK]SEGMENT.MAP

OUTPUT FORMAT = ASCII

VAX/99000 LINKER VERSION v.2.4 85.122 5/20/85 11:24:46 PAGE 3

PHASE 0, SEGMENT  ORIGIN = 0000  LENGTH = 0000

MODULE  NO  ORIGIN  LENGTH    TYPE     DATE      TIME     CREATOR

ALPHA    1    1000    0014   INCLUDE  5/20/85  11:24:46  ASM700
$DATA    1    2000    0020
BETA     2    1014    0008   INCLUDE  5/20/85  11:24:46  ASM700
$DATA    2    2020    0002
GAMMA    3    101C    0008   INCLUDE  5/20/85  11:24:46  ASM700
$DATA    3    2022    0002

COMMON              NO          ORIGIN    LENGTH

COMDAT               3          3000      0002

                    D E F I N I T I O N S

NAME         VALUE NO   NAME     VALUE NO

BETA         0014  2    *GAMMA   101C* 3

**** LINKING COMPLETED     5/20/85   11:24:54
```

Figure 4-5. Listing File for ROM/RAM Partitioning

4.3 Partial Linking

This section shows how to generate a partial link and then include the output of the partial link in a subsequent link. Only ASCII object code can be used in partial linking on the TMS320 and TMS7000 devices; both ASCII and compressed code can be used on the TMS99000 devices.

The PARTIAL command is used in the control stream to specify a partial link. In this example, modules BETA and GAMMA are to be linked together in a partial link. The output of the partial link is not executable and must be linked again without the PARTIAL command so that the output of this partial link will then be linked with

module ALPHA to produce an executable module. The following is the control stream for the partial link, using the TI/IBM PC (PC/MS-DOS) operating systems:

```
PARTIAL
PHASE     0,PARTIAL
INCLUDE   A:BETA.MPO
INCLUDE   A:GAMMA.MPO
END
```

All commands pertaining to partial links must be issued before any INCLUDE, SEARCH, and FIND commands. The PARTIAL command must be given before the first INCLUDE command in the control stream. In a partial link, only one phase is allowed and must be defined by the PHASE 0 or TASK command.

The ALLGLOBAL, GLOBAL, and NOTGLOBAL commands are used with the PARTIAL command to define the scope of DEF tags in modules included in the partial link. These symbols are specified as either global or local. All externally-defined symbols are processed as global symbols. Global symbols are externally defined in the partially linked output modules and may be referenced in a subsequent link. Local symbols are not externally defined in the partially linked output module; therefore, they may be referenced in the current partial link. Since none of these commands are included in the control stream, the default, ALLGLOBAL, is used.

The output of the partial link can now be linked with module ALPHA to produce an executable module, using the following control stream:

```
PHASE     0,MAIN
INCLUDE   B:ALPHA.MPO
INCLUDE   B:PARTIAL.MPO
END
```

The listing and object modules from a partial link using the PARTIAL command are given in Figure 4-6.

The second part of the link, in which the output of the partial link is relinked without using the PARTIAL command, is performed next. The listing and object files for relinking the output of the partial link are shown in Figure 4-7.

```
PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 1
COMMAND LIST
        PARTIAL
        PHASE       0,PARTIAL
        INCLUDE     A:BETA.MPO
        INCLUDE     A:GAMMA.MPO
        END

PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 2
LINK MAP
CONTROL FILE = A:PARTIAL.CON

LINKED OUTPUT FILE = A:PARTIAL.MPO

LIST FILE = A:PARTIAL.MAP

OUTPUT FORMAT = ASCII

PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 3

PHASE 0  PARTIAL MODULE  ORIGIN = 0000  LENGTH = 001A

MODULE  NO   ORIGIN  LENGTH    TYPE    DATE     TIME    CREATOR

BETA     1    0000    000A    INCLUDE  5/20/85  11:34:46  ASM320
$DATA    1    0000    0002
GAMMA    2    000A    000A    INCLUDE  5/20/85  11:34:46  ASM320
$DATA    2    0002    0002

COMMON              NO          ORIGIN    LENGTH

COMDAT               1          0000      0002

                      D E F I N I T I O N S

NAME        VALUE NO   NAME     VALUE NO

BETA        0000  1    *GAMMA   000A  2

LENGTH OF REGION FOR TASk             = 001A

NUMBER OF RECORDS FOR MODULE PARTIAL =    3

TOTAL RECORDS WRITTEN                 =    3

**** LINKING COMPLETED    5/20/85   11:34:54
```

a. Listing File for a Partial Link

```
K0014PARTIAL M0004$DATA 0000M0002COMDAT00025000BETA   5000AGAMMA 7F115F
A0000B0700BC820N00000002T0000B045BA000AB0700BC820N00000002T00027F27FF
B045BI0000BETA     I000AGAMMA  7F92AF
:     PARTIAL      5/20/85     11:36:48   LINK990    v2.4    85.122
```

b. Object File for a Partial Link

Figure 4-6. Listing and Object Files for a Partial Link

```
PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 1
COMMAND LIST

          PHASE     0,TEST
          INCLUDE   A:ALPHA.MPO
          INCLUDE   A:PARTIAL.MPO
          END

PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 2
LINK MAP

CONTROL FILE = A:TEST.CON

LINKED OUTPUT FILE = A:TEST.LOD

LIST FILE = A:TEST.MAP

OUTPUT FORMAT = ASCII

PC/CrossWare Family Linker v.2.4 85.122 5/20/85 11:34:46 PAGE 3

PHASE 0  MAIN   MODULE  ORIGIN = 0000      LENGTH = 004E
MODULE  NO   ORIGIN  LENGTH    TYPE    DATE     TIME    CREATOR

ALPHA    1     0000    0014    INCLUDE  5/20/85  11:44:46  ASM320
$DATA    1     0028    0020
PARTIAL  2     0014    0014    INCLUDE  5/20/85  11:44:46  ASM320
$DATA    2     0048    0004

COMMON              NO          ORIGIN    LENGTH

COMDAT              1           004C      0002


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

*BETA        0014  2    *GAMMA   001E 2

LENGTH OF REGION FOR TASK             = 004E

NUMBER OF RECORDS FOR MODULE PARTIAL =    3

TOTAL RECORDS WRITTEN                 =    3

**** LINKING COMPLETED     5/20/85   11:44:54
```

a. Listing File for Relinking the Partial Link Output

```
K004EMAIN     A0000B0700BC800C004CB06A0C0014B06A0C001EB0600B16F87F258F
B0340A0014B0700BC820C004CC0048B045BB0700BC820C004CC004AB045B7F2D5F
I0000ALPHA     I0014PARTIAL  I001BETA    I001EGAMMA    7F433F
:    MAIN       5/20/85      11:46:48   LINK990     v2.4    85.122
```

b. Object File for Relinking the Partial Link Output

Figure 4-7. Listing and Object Files for Relinking the Partial Link Output

4.4 Library Creation

The linker can accommodate two object library types: random and sequential.

A random library can be created almost automatically. Whenever one or more object files are placed in the same directory or sub-directory, that directory becomes a random library. Some examples of random libraries are as follows:

PROJ.A2710.OBJ (DX10), where the example indicates a directory containing object file members named (e.g.,, SIN, COS, SUB2, HEAPSORT, etc.).

DUA0:[USER07.PROJ2710.PARTS] (VAX), where the example indicates a directory containing object file members named (e.g., PART1.OBJ, INITIAL.OBJ, CLEANUP.OBJ, etc.).

A:*.MPO (MS-DOS), where the name indicates a drive and all files with the extension .MPO (e.g., QUICK.MPO, BTREE.MPO, SHELL.MPO, etc.).

The creation of sequential libraries is more involved. Since sequential libraries offer no advantages over random libraries, their use will probably be restricted to those users of systems not supporting random libraries, i.e., not supporting multilevel directories or "wild-card" file specifications.

A sequential library is a single file and consists of a "dictionary," followed by one or more concatinated object modules. The user must order the elements in a sequential library so that no object segment contains an external reference to a preceding segment. The concatinated object files may be created by assembling a source file created by concatinating the source files of several proposed members of the sequential library. Such a source file might appear as shown in Figure 4-8.

```
          IDT    'TRESRT'
          TITL   'THIS IS THE FIRST LIB MEMBER'
          DEF    TRESRT,QUICK
*
QUICK     EQU    $
           .
           .
           .
TRESRT    EQU    $
           .
          END
*
          IDT    'ELEM'
          TITL   'STILL IN SEQ LIB'
          DEF    ELEM
           .
           .
           .
          END
*
          IDT    'LASTPROG'
          TITL   'ET CETERA'
           .
           .
           .
          END
```

Figure 4-8. Source File for Sequential Library Creation

The assembler output contains all modules within the same file, yet distinct. The use of a text editor allows the creation of such a file by appending the object files that result from independent assembly of the proposed library members. The dictionary structure must be created by use of the text editor. It precedes the first object module and must match the following pattern:

First line, the library 'IDT' record: 0llllaaaaaaaaF

where 0 = tag
llll = length of the dictionary
aaaaaaaa = library name
F = end-of-record tag
user-defined information out to 80th character

Example: 00000SEQLIB01F 04/10/85 08:15:00

One entry for each object module included in the library:
FllllIaaaaaaaa;

where F = tag
I = IDT marker
llll = length of PSEG of module
aaaaaaaa = IDT of the module
';' = record end marker

Example: FI00CEFASTSORT;

For each DEF within a module: FEtIIIIaaaaaa{,tIIIIaaaaaa};

where F = tag
E = tag
t = type of DEF ("A" = absolute, "R" = relative)
IIII = value of the DEF'd symbol
aaaaaa = 6-character name of symbol DEF'd in this module
, = if more (up to five DEFs allowed per record
; = if no more

Example: FEA0050PACK ,R00ACUNPAK ,A0E08ENCODE;

For each REF within the module: FR]]]]]aaaaaa{,]]]]]aaaaaa}

where F = tag
R = tag
']]]]]' = five blanks
aaaaaa = 6-character name of symbol REF'd
, = if more (up to five REFs allowed per record)
; = if no more

Example: FR]]]]]EXREF ,]]]]]ESYM ,]]]]]X]]]]]

For each COMMON segment contained within the module:
FCIIIIaaaaaa{,IIIIaaaaaa};

where F = tag
C = tag
IIII = length of COMMON segment
aaaaaa = 6-character COMMON segment name

Example: FC0140COMNAM,0266$BLANK;

For each DATA segment defined within the module:
FDIIII]]]]]]{,IIII]]]]]]};

where F = tag
D = tag
IIII = length of DATA segment
']]]]]]' = six blanks
up to five DSEGs allowed, separated by commas
terminated by ;

Example: FD0010]]]]]],0032]]]]]]

This set of records is repeated for each object module in the library. The final record, just prior to the first record of the first library member, must contain a colon in the first character position. The remainder of this colon record is not specified. It can be used for date, time, and other user-defined information.

5. Link Editor Error Messages

Messages are listed for detected errors in the listing file. Linker error messages are named and described below.

When the error-message description indicates that a malfunction of the link editor has occurred, please contact the Texas Instruments Customer Response Center (CRC) hotline number, 1-800-232-3200, extension 2171, for assistance.

'(' EXPECTED: The REPLACE command expects a parenthesis.

ADDRESS SPACE HAS OVERFLOWED IN THIS MODULE: The maximum address required to represent this module is >10000 or greater. No valid object module can be produced for this phase. The linker continues to produce the map, but with increased likelihood that it will abort from internal errors.

ADDRESS SPACE TRUNCATED FOR TAG = X IN THE SEGMENT STARTING AT YYYY: The 320-specific tags have a seven-bit address field that has overflowed.

ALIGNMENT VALUE MUST BE IN THE RANGE 0..15: The value in the ADJUST command is out of range.

AN ACTIVE BUFFER SHOULD HAVE BEEN CLOSED: A buffer that needs to be closed is still marked as active.

ATTEMPT MADE TO WRITE TO A NIL SEGMENT: The linker attempted to write to a nonexistent segment. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT MADE TO WRITE TO INACTIVE SEGMENT: The linker has attempted to write to an unopen segment. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT TO ACTIVATE AN ALREADY ACTIVE SEGMENT: The linker has attempted to open a segment that is already active. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT TO ACTIVATE NIL SEGMENT: The linker has attempted to open a nonexistent segment. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT TO ALLOCATE AFTER DSEG OF CSEG DEFINED: The ALLOCATE command has been given after data and/or common segments have been encountered.

ATTEMPT TO MOVE NON-COMMON SEGMENT: An attempt has been made to move a segment that is not a common. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT TO ORDER A NIL SEGMENT: A common that was never defined cannot be placed in the stream of commons. Indicates a malfunction of the link editor; call hotline immediately.

ATTEMPT TO REDEFINE COMMON ORIGIN: Directives to place a common origin provide information that conflicts with information that has been already determined.

BAD CHAIN FOR XXXX TO YYYY: In a partial link, the reference chain for XXXX points to the address YYYY that is outside the scope of the segment.

BAD INDEX - COMMAND FORMAT NOT RECOGNIZED: In a partial link, an error was detected in the control record such that the current index is negative.

BAD TAG IN CHAIN FOR XXXX AT YYYY: In a partial link, an invalid tag was encountered in the processing of the reference chain. If this error occurs, the object module has been damaged.

CANNOT ORDER COMMONS FROM DIFFERENT MODULES: Commons that are overlayed in procedures of the same phase level cannot be ordered together. This usually occurs in a partial link.

CAN'T ASSIGN LUNO TO LIBRARY: The DX10 system assigns a luno (logical unit number) to each library identified in a LIBRARY or SEARCH command. This message indicates that an error code was returned, and the library was not entered into the library table for later use. (Luno's are assigned so that later references to library members will be resolved more quickly by the operating system. In addition, this permits early identification of error when a library is incorrectly specified.) If using a DOS system, this message indicates that the limit has been exceeded in the number of files that can be opened at one time.

CAN'T OPEN FILE: A file that should be opened cannot be opened. The system return code is identified in the immediate preceding warning.

CAN'T OPEN LIBRARY: The DX10 system reads in the directory of each library specified in the LIBRARY commands, except the last. This message indicates that an error code was returned or an attempt made to open the directory. (A luno has already been assigned to the library. If this message occurs, it is probably because the luno has not been assigned to the directory file.) The library's directories are read in to permit the linker to minimize the number of abortive file-open operations during automatic call.

CHAIN TO UNINITIALIZED LOCATION FOR XXXX AT YYYY: The reference chain for value XXXX points from YYYY to an insignificant address. Processing of the chain is discontinued.

COMMAND NOT VALID WITH PARTIAL LINK: The specified command is not valid in producing a partial link.

COMMAND ONLY VALID WITH PARTIAL LINK: The GLOBAL, NOTGLOBAL, and ALLGLOBAL commands must follow a PARTIAL command in a control stream.

COMMON HAS NOT BEEN PLACED VALIDLY: An attempt has been made to place a common at two or more locations.

COMMON NAME TRUNCATED: The common name has been truncated to six characters. This is a trivial warning and processing proceeds with the truncated name.

COMMON NUMBER INVALID: The common number given an M tag is not in the valid range.

COMMON ORIGIN HAS ALREADY BEEN SET: An attempt has been made to define a common origin already set with a previous COMMON command.

COMMON ORIGIN INVALID: The origin for a common is not valid.

COMMON ORIGIN WAS NOT DEFINED: The first common name specified in a COMMON command was never defined in the link.

COMMON SEGMENT HAS NO SYMBOL DEFINED: A symbol has been given for a common segment, but the segment itself does not exist.

COMMON SYMBOL IS NOT VALID: The given common symbol is not legal.

COMMON SYMBOL WAS NEVER DEFINED: A common segment was not found corresponding to the common symbol.

COMPRESSED FORMAT NOT SUPPORTED FOR 320 & 7000 INPUT MODULES: The linker only recognizes ASCII-formatted input modules for these object codes.

CONFLICTING COMMON SYMBOL FOUND: A common symbol that is not consistent with previous commons has been detected.

CURRENT SEGMENT HAS NOT BEEN DEACTIVATED: The segment that should be closed has been left active.

DUPLICATE SYMBOL DEFINITION ENCOUNTERED: Two definitions have been encountered for the same external symbol.

ENTRY NAME TRUNCATED: The entry name has been truncated to six characters. This is a trivial warning and processing proceeds with the truncated name.

EXTERNAL REFERENCE INDEX OUT OF RANGE: The index number specified by an E tag in an input object module is not valid.

EXTERNAL SYMBOL TRUNCATED: A symbol specified in a GLOBAL or NOTGLOBAL command exceeds six characters in length.

FATAL ERROR DETECTED -- *LINKER ABORTING*: An error that the linker cannot recover from has been detected. The user should repeat the process. If the message occurs again, then either check the procedures used or call the hotline for assistance.

FIRST PHASE HAS ALREADY BEEN DEFINED: A NOMAP command has appeared after a TASK or a PHASE command has been issued in the control stream.

HEAP ERROR ENCOUNTERED IN PASS2: A heap error was detected while trying to allocate data space for the second pass. Indicates a malfunction of the link editor; call hotline immediately.

ILLEGAL INTERMEDIATE TAG ENCOUNTERED AT XXXX: An encoded tag was encountered that was not valid. Indicates a malfunction of the link editor; call hotline immediately.

ILLEGAL TAG FOUND IN INTERMEDIATE FILE; TAG=X: An invalid tag was found in the intermediate file. Indicates a malfunction of the link editor; call hotline immediately.

INTERMEDIATE FILE OVERFLOW: The maximum number of records for intermediate object representation has been exceeded. Obtain more file space by making individual object modules smaller or call the hotline for assistance.

INTERMEDIATE RECORD NUMBER INVALID: The record index for the intermediate storage is not in the legal range.

INTERNAL LINKER ERROR IN AUTOCALL: An error has occurred in the automatic-call algorithm. This error should never occur. If it does, unresolved references may be the result of modules not having been read in; in other respects, the object module produced should be good. Relink using specific INCLUDES for the missing modules or call the hotline for assistance.

INVALID ATTEMPT TO MOVE FIRST COMMON: The linker has attempted to move a common that was specified as the first common by a COMMON command. Indicates a malfunction of the link editor; call hotline immediately.

INVALID ATTEMPT TO READ BUFFER: The linker has attempted to activate a buffer that is not of the correct type. Indicates a malfunction of the link editor; call hotline immediately.

INVALID LEVEL FOR PHASE: The level argument to a PHASE command is not appropriate. The first phase established must be a TASK or a phase of level 0. If the current level is N, a new phase must have level $\leq N+1$.

INVALID PROCEDURE LEVEL: The level for procedures must be 1 or 2.

INVALID PROCEDURE SPECIFIED: An illegal procedure has been declared.

INVALID SYMBOL NAME FOR REPLACE: The REPLACE command has encountered an illegal symbol name.

INVALID VALUE FOR LINES PER PAGE: The argument to a page command is not recognized as a positive integer or is out of the range of 16 to 60 lines per page.

LAST COMMON IN LIST IS NIL: The last common in a list of ordered commons does not exist. Indicates a malfunction of the link editor; call hotline immediately.

LAST MODULE FOR PHASE IS NIL: The linker cannot find the information about the current phase. Indicates a malfunction of the link editor; call hotline immediately.

MAP RECORD INDEX IS OUT OF RANGE: The map record is full or the index has been changed to an invalid value. Indicates a malfunction of the link editor; call hotline immediately.

MEMBER NAME TOO LONG: The member name exceeds eight characters. The command is not processed.

MEMBER NAME TRUNCATED: The member name has been truncated to eight characters. This is a trivial warning and processing proceeds with the truncated name.

MINIMUM NUMBER OF LINES PER PAGE IS 16: A PAGE N command may not specify a value of N less than 16.

MODULE LENGTH IS ZERO: The length for the module has been incorrectly specified as zero.

MODULE ORIGIN IS NOT ZERO: The origin for a module must be zero before the relocation is applied.

NIL COMMON SEGMENT WAS ACTIVATED: An attempt was made to activate a common segment that does not exist.

NIL SEGMENT FOR M TAG SYMBOL: An M tag definition applies to a segment that does not exist. Indicates a malfunction of the link editor; call hotline immediately.

NO PHASE IS DEFINED: No PROCEDURE, TASK, or PHASE 0 has been defined. A command has been given which requires that object modules be read in or that some phase be active.

NO TASK PHASE IS DEFINED: No TASK or PHASE 0 has been defined. A valid set of linked object modules cannot be produced.

NOTGLOBAL MUST PRECEDE A GLOBAL COMMAND: The GLOBAL command is only valid if it is preceded by a NOTGLOBAL command with no parameter.

OBJECT CARD INDEX ERROR DETECTED: After writing an object record, the index into the record was not equal to one. Indicates a malfunction of the link editor; call hotline immediately.

ORIGIN CANNOT BE WRITTEN TO INACTIVE SEGMENT: The linker has attempted to write origin information to a segment that is not open. Indicates a malfunction of the link editor; call hotline immediately.

OVERWRITTEN BLOCKS FOR XXXX TO YYYY: Absolutely placed object code overlaps at the given address.

OVERWRITTEN SEGMENTS STARTING AT XXXX IN MODULE NNNNNNNN: Overlapping segments have been detected starting at location XXXX. The link map specifies which segment starts at that point. This is flagged as a warning.

PARTIAL COMMAND INVALID IN CONTEXT: The PARTIAL command was specified in the control stream after a command that is inconsistent with partial links (e.g., DUMMY, PROGRAM, DATA, COMMON, ALLOCATE, PROCEDURE, PHASE 1, etc.)

PHASE LEVEL EXPECTED: The level argument to a PHASE command must be a zero or a positive integer.

PHASE LEVEL SPECIFIED IS NOT VALID: The level specified in a PHASE command is not in the valid range.

PHASE NAME TRUNCATED: The phase name has been truncated to eight characters. This is a trivial warning, and processing proceeds with the truncated name.

PHASE SEQUENCE IS NOT VALID: The order in which the phases have been declared is not legal.

PREMATURE END OF CONTROL FILE: The control file has ended before an END command was encountered. No further processing is done.

PROCEDURE CANNOT HAVE BROTHERS: A procedure cannot have phases at the same level defined with it.

PROCEDURE NAME TRUNCATED: The procedure name has been truncated to eight characters. This is a trivial warning, and processing proceeds with the truncated name.

PROC 1 MUST BE DUMMIED TO DUMMY PROC 2: In order to dummy the second procedure, the first procedure must also be dummied.

PROC 1 SYMBOL NUMBER IS NOT ZERO: The symbol number for the procedure must be zero.

PROC 2 SYMBOL NUMBER IS NOT ZERO: The symbol number for the procedure must be zero.

RELOCATABLE ADDRESS IS NOT VALID;SEGMENTS SHOULD BE PLACED AT ABSOLUTE LOCATIONS: Certain TMS320-specific tags require that segments to which they refer be placed at absolute addresses.

SEGMENT BUFFER HAS BEEN DAMAGED: The current segment does not contain the expected information. Indicates a malfunction of the link editor; call hotline immediately.

SEGMENT ORIGIN IS ZERO: The origin for a segment has erroneously changed to zero. Indicates a malfunction of the link editor; call hotline immediately.

TASK OR PHASE 0 IS ALREADY DEFINED: A PROCEDURE command cannot be given once a task phase has been defined.

TASK OR PHASE 0 MUST BE DEFINED BEFORE OVERLAY: An overlay has been defined before the task or root phase.

THE CURRENT SEGMENT IS NIL: The segment that is being examined does not exist. Indicates a malfunction of the link editor; call hotline immediately.

THE INPUT OBJECT MODULE HAS BEEN DAMAGED: Unexpected or invalid tags and values have been encountered in the input object module.

THE MAP RECORD IS NIL: An attempt has been made to place information into the map record when the record does not exist. Indicates a malfunction of the link editor; call hotline immediately.

THE OVERWRITTEN BLOCKS ARE NOT COMPATIBLE: The types of the overlapping blocks are not the same, and a valid object module cannot be produced.

THE PHASE TYPE IS NOT TASK, OVLY, OR PROC: This error should never occur, because the only valid types are TASK, OVLY, and PROC.

THE SEGMENT TYPE IS NOT PSEG, DSEG, OR CSEG: The only valid segment types are PSEG, DSEG, and CSEG.

TOO MANY SYMBOLS HAVE BEEN DEFINED: The statically allocated arrays that contain the values for symbols (mostly external symbols and phase lengths, origins, etc.) have overflowed. The number of symbols allowed in a symbol table is 1110.

UNABLE TO PROPERLY ORDER COMMONS: The linker cannot order the commons as specified.

UNEXPECTED TAG: The input object module is not of the expected format. It may not really be an object module. Processing stops.

UNRECOGNIZED FORMAT: The argument to the FORMAT command is not recognized.

UNRECOGNIZED COMMAND: The command on the most recent line is not recognized as a linker command. The line is ignored.

UNSUPPORTED INTER-SEGMENT LINK FOR XXXX FROM YYYY TO ZZZZ: This message is printed by the second pass, and applies to the module in the linked output that is next identified. An external reference chain has pointed from one PSEG, DSEG, or CSEG into another. XXXX is the value of the external symbol to be filled in. (The second pass cannot identify it by symbol name. The name can often be found by examining the symbol definitions. A symbol with a value of zero may be an unresolved reference.) YYYY, the address where the chain starts, identifies the offending module. ZZZZ is the address to which the chain points. The only deficiency in the linked object is that incorrect values remain where the value of external symbol XXXX should have been inserted.

A. Glossary

This Glossary includes terms used in this manual or related documents. The definitions provided apply to link editors in general.

Absolute Address: An address that designates a specific physical location in memory.

Access Name: A uniquely identifying name by which a directory or file can be accessed. An access name may include a device name, a directory name, or a file name.

Address: An expression, usually numerical, that designates a location in a storage or memory device.

ASCII Object Code Format: The default format for object code produced by the assembler or link editor. The format consists of an ASCII tag character followed by one or two ASCII fields. The first field is numeric in value, and the optional second field contains a symbol. Refer to the appropriate Assembly Language Programmer's Guide for further information.

Assembler: A system utility that translates assembly language instructions (symbolic source code) into machine language instructions (binary object code) so they can then be executed by the hardware on a step-by-step basis.

Base: A reference value.

Base Address: The value to which a relative address is added to obtain an absolute address.

BNPF Formatted File: A file containing object code in binary form, where P denotes a logic 1 and N denotes a logic 0. Data for each word begins with a B, followed by the bit values and ending with an F.

Compressed Object Code Format: An optional format for object code where numeric fields are expressed in binary rather than ASCII to conserve space.

CRU: Communications Register Unit - A command-driven, bit-addressable, input/output interface. The processor instruction can set, reset, or test any bit in the CRU array or move data between the memory and CRU data field.

CSEG: An assembler directive that defines the beginning of a common data segment.

Debug: To detect, locate, and remove mistakes in software or malfunctions in the hardware.

DEF: An assembler directive that causes specified symbols to be tagged as external definitions in the object module. Defined symbols then are available to other modules and separately assembled modules can be linked.

Default: A value used for a parameter when no other value is supplied by the user.

Device: Physical equipment such as a diskette drive or VDT.

Device Name: A character string assigned by the system to specify a physical device (e.g., LP01 is the device name for a line printer).

Directory: An index file that contains the information necessary to locate other files listed in that directory and to describe the characteristics of those files. It does not contain user data.

Directory Name: A character string assigned by the user that identifies a directory. The directory name is part of the access name for all files indexed in that directory. The directory name for a diskette directory can be either the volume name assigned to the diskette or the device name of the drive in which the diskette is residing.

Diskette Directory: The directory that contains all user directories and all files not included in a user directory on a diskette.

Diskette Name: Either the volume name assigned to the diskette or the device name of the drive in which the diskette is residing.

DSEG: An assembler directive that defines the beginning of a data segment.

Entry Point: In a software program, any place to which control can be passed in a particular software module.

EPROM: Erasable programmable read-only memory - an MOS semiconductor memory storage element that can be programmed after packaging with a fixed (read-only) program. Programmed data can be erased by exposing the EPROM to ultraviolet light. The EPROM can then be reprogrammed.

File: A collection of data generated by the user or the system.

File Access Name: An access name that specifies a text or data file.

File Management: Process of organizing data on a storage device and managing transfer of that data between main memory and the storage device.

File Manager: A set of file management utilities.

File Name: A character string assigned by the user to identify a file. The file name is part of the access name for the file.

File-Oriented Device: A device that can be used for processing or storing files, such as a diskette drive or a line printer.

Host System: The system that is monitoring and controlling operation of the target system.

Initial Value: The value initially displayed or supplied by the system for a parameter.

I/O: Input/Output.

Library: A directory or file used by the link editor, consisting of object modules that may be included in the linking process.

Link Editor: A system utility used to link object modules that have been separately assembled or compiled to produce an object module that can be loaded into the target memory and executed.

Listing File: A file generated by the show directory utility, assembler, and link editor, which contains information pertaining to execution of the utility or the results.

Memory Word: Data consisting of 16 bits (2 bytes).

Microprocessor: An integrated circuit that can be programmed with stored instructions to perform a wide variety of functions.

Object Code: Output from an assembler or compiler that is itself executable machine code or is suitable for processing (i.e., linking) to produce executable machine code.

Operating System: Software that controls the execution of all programs, routines, tasks, etc. that execute in a computer.

Operator Interface: The means by which the user communicates with the system.

Overlay: Parts (phases) of a program which share memory and are loaded into executable memory during program execution.

PROM: Programmable read-only memory - a bipolar semiconductor memory storage element that can be programmed after packaging with a fixed (read-only) program.

PROM Personality Card: An interchangeable part that snaps onto the front of the PROM programmer console. There are four types of personality cards, each of which is used to program or read a particular class of PROM or EPROM devices.

PROM Utility: A system utility used in conjunction with the PROM programming hardware to store data in or read data from a PROM or EPROM.

PROM Word: A group of data in a PROM or EPROM that can be accessed by one address. The number of bits per each PROM word depends on the type of PROM or EPROM being used.

Prompt: A display on a terminal that requests the user to enter some information.

Prompt Response: The value entered for a prompt. This value is passed as a parameter to the utility that has been invoked.

PSEG: An assembler directive that defines the beginning of a program segment.

RAM: Random-access memory; a memory storage element from which data can be read or to which data can be written as required by the user application.

Random Library: A directory, used by the link editor, which lists files that contain object modules that may be included in the linking process.

REF: An assembler directive that causes specified sysmbols to be tagged as external references in the object code. The link editor resolves external references by matching REF tags with DEF tags from other modules.

Relative Address: An address that is relative to the beginning of a program. The value of a relative address is added to the base address to obtain an absolute address.

Relocatable Code: A program or routine that can be moved from one portion of memory to another and have the necessary address references adjusted so that it can be executed from its new location.

Response Field: The area immediately following a command prompt in which the user may enter a parameter.

ROM: Read-only memory; a fixed program semiconductor storage element that has been hard-coded by the manufacturer with a permanent program.

Sequential File: A file consisting of variable-length logical records that must be accessed sequentially.

Sequential Library: A sequential file used by the link editor, consisting of concatenated object modules that were generated by previous partial links and may be included in the current linking process.

Source code: Code input to a compiler or assembler in order to generate machine executable code. Source code can be written in assembly language or in some high-level language, such as Pascal.

Source Module: A file generated by using the text editor, which contains source statements (i.e., assembly language or high-level language statements) for all or part of a software program.

Symbol Resolution: A function of the link editor utility, consisting of matching REF and DEF tag symbols to form different modules and inserting the correct locations for these symbols in the linked object code.

Target System: The system under development.

Text Editor: A system utility used to create and edit text and data files.

Text Formatter: A system utility used to format text files for purposes of readability and printing.

User Directory: A directory created by the user and listed under the diskette directory.

Utility: A software program executed by the operating system to perform a special function.

VDT: Video display terminal

Volume Name: A character string assigned by the user to a particular storage device.

Index

A

ADJU
  Specify Alignment of Phase Command 3-6
ALLG
  Declare Global Symbols Command 3-7
ALLO
  Allocate Relative Positioning of Segments Command 3-8
AUTO
  Automatic Symbol Resolution Command 3-9

C

COMM
  Set Starting Location Counter for CSEG Command 3-10
commands (see linker commands)

D

DATA
  Set Starting Location Counter for DSEG Command 3-11
description 1-2
DUMM
  Suppress Generation of Linked Output File Command 3-12

E

END
  Specify End of Control Stream Command 3-13
entering a command 3-2
ENTR
  Specify a Symbol for an Entry Tag Command 3-14
error messages 5-1
examples 4-1
  library creation 4-10
  partial linking 4-6
  ROM/RAM partitioning 4-5
  simple linking 4-2

F

FIND
  Search Sequential Libraries for Unresolved References Command 3-15
FORM
  Define Format of Linked Output Module Command 3-16

G

GLOB
  Identify Global Symbols Command 3-17

I

INCL
  Specify Modules To Be Included in Link Command 3-18

L

LIBR
  Define Random Library Directories Command 3-19
libraries 2-2
link control file 2-2
link editor files 2-1
  libraries 2-2
    symbol resolution 2-2
  link control file 2-2
  linked output file 2-3
  listing file 2-3
    link map 2-3
  object modules 2-2
link map 2-3
linked output file 2-3
linker command set summary 3-2
linker commands 3-1

ADJUST (Specify Alignment of Phase) 3-6
ALLGLOBAL (Declare Global Symbols) 3-7
ALLOCATE (Allocate Relative Positioning of Segments) 3-8
AUTO (Automatic Symbol Resolution) 3-9
command set summary 3-2
COMMON (Set Starting Counter for CSEG) 3-10
DATA (Set Starting Counter for DSEG) 3-11
DUMMY (Supress Generation of Linked Output File) 3-12
END (Specify End of Control Stream) 3-13
entering a command 3-2
ENTRY (Specify a Symbol for an Entry Tag) 3-14
FIND (Search Sequential Libraries for Unresolved References) 3-15
FORMAT (Define Format of Linked Output Module) 3-16
GLOBAL (Identify Global Symbols) 3-17
INCLUDE (Specify Modules To Be Included in Link) 3-18
LIBRARY (Define Random Library Directories) 3-19
NOAUTO (Inhibit Automatic Symbol Resolution) 3-20
NOMAP (Omit Module, Common, and Symbol Maps from Listing) 3-21
NOPAGE (Set No Page Ejects Between Link Maps) 3-22
NOSYMT (Omit Symbol Table from Modules) 3-23
NOTGLOBAL (Define Local Symbols) 3-24
PAGE (Set Page Eject to Separate Link Maps) 3-25
PARTIAL (Perform Partial Link) 3-26
PHASE (Define Phase Level and Name) 3-27
PROCEDURE (Define Phase as Procedure) 3-28
PROGRAM (Define Absolute Counter for PSEG) 3-29
REPLACE (Relate Oldsym with Newsym) 3-30
SEARCH (Search for Unresolved References) 3-31
SYMT (Include Symbol Tables in Linked Output File) 3-32
TASK (Define Phase as Task) 3-33
listing file 2-3

N

NOAU
Inhibit Automatic Symbol Resolution Command 3-20
NOMA
Omit Module, Common, and Symbol Mapsfrom Listing Command 3-21
NOPA
Set No Page Ejects Between Link Maps Command 3-22
NOSY
Omit Symbol Table from Modules Command 3-23
NOTG
Define Local Symbols Command 3-24

O

object modules 2-2

P

PAGE
Set Page Eject to Separate Link Maps Command 3-25
PART
Perform Partial Link Command 3-26
partial linking 4-6
PHAS
Define Phase Level and Name Command 3-27
PROC
Define Phase as Procedure Command 3-28
procedure/task segmentation 1-2, 3-28, 3-33
PROG
Define Absolute Starting Counter for PSEG Command 3-29
program definition 1-2

R

random libraries
creation 4-10
definition 2-2, 3-19
search using SEARCH command 3-31
REPL
Replace Oldsym with Newsym Command 3-30
ROM/RAM partitioning 4-5

S

SEAR
Search for Unresolved References Command 3-31
segment
positioning 1-2, 3-8
sequential libraries
creation 4-10
definition 2-2
search using FIND command 3-15
simple linking 4-2
symbol resolution
automatic resolution 2-2
inhibit resolution 3-20
user-defined resolution 2-3
SYMT
Include Symbol Tables in Linked Output File Command 3-32

T

TASK
Define Phase as Task Command 3-33

TI Worldwide Sales Offices

ALABAMA: Huntsville: 500 Wynn Drive, Suite 514, Huntsville, AL 35805, (205) 837-7530.

ARIZONA: Phoenix: 8825 N. 23rd Ave., Phoenix, AZ 85021, (602) 995-1007.

CALIFORNIA: Irvine: 17891 Cartwright Rd., Irvine, CA 92714, (714) 660-1200; **Sacramento:** 1900 Point West Way, Suite 171, Sacramento, CA 95815, (916) 929-1521; **San Diego:** 4333 View Ridge Ave., Suite B., San Diego, CA 92123, (619) 278-9601; **Santa Clara:** 5353 Betsy Ross Dr., Santa Clara, CA 95054, (408) 980-9000; **Torrance:** 19505 Hamilton St., Bldg. A, Suite 1, Torrance, CA 90502, (213) 217-7010; **Woodland Hills:** 21220 Erwin St., Woodland Hills, CA 91367, (213) 704-7759.

COLORADO: Aurora: 1400 S. Potomac Ave., Suite 101, Aurora, CO 80012, (303) 368-8000.

CONNECTICUT: Wallingford: 9 Barnes Industrial Park Rd., Barnes Industrial Park, Wallingford, CT 06492, (203) 269-0074.

FLORIDA: Ft. Lauderdale: 2765 N.W. 62nd St., Ft. Lauderdale, FL 33309, (305) 973-8502; **Maitland:** 2601 Maitland Center Parkway, Maitland, FL 32751, (305) 660-4600; **Tampa:** 5010 W. Kennedy Blvd., Suite 101, Tampa, FL 33609, (813) 870-6420.

GEORGIA: Norcross: 5515 Spalding Drive, Norcross, GA 30092, (404) 662-7900

ILLINOIS: Arlington Heights: 515 W. Algonquin, Arlington Heights, IL 60005. (312) 640-2925.

INDIANA: Ft. Wayne: 2020 Inwood Dr., Ft. Wayne, IN 46815, (219) 424-5174; **Indianapolis:** 2346 S. Lynhurst, Suite J-400, Indianapolis, IN 46241, (317) 248-8555.

IOWA: Cedar Rapids: 373 Collins Rd. NE, Suite 200, Cedar Rapids, IA 52402, (319) 395-9550.

MARYLAND: Baltimore: 1 Rutherford Pl., 7133 Rutherford Rd., Baltimore, MD 21207, (301) 944-8600.

MASSACHUSETTS: Waltham: 504 Totten Pond Rd., Waltham, MA 02154, (617) 895-9100.

MICHIGAN: Farmington Hills: 33737 W. 12 Mile Rd., Farmington Hills, MI 48018, (313) 553-1500.

MINNESOTA: Eden Prairie: 11000 W. 78th St., Eden Prairie, MN 55344 (612) 828-9300.

MISSOURI: Kansas City: 8080 Ward Pkwy., Kansas City, MO 64114, (816) 523-2500; **St. Louis:** 11861 Westline Industrial Drive, St. Louis, MO 63141, (314) 569-7600.

NEW JERSEY: Iselin: 485E U.S. Route 1 South, Parkway Towers, Iselin, NJ 08830 (201) 750-1050

NEW MEXICO: Albuquerque: 2820-D Broadbent Pkwy NE, Albuquerque, NM 87107, (505) 345-2555.

NEW YORK: East Syracuse: 6365 Collamer Dr., East Syracuse, NY 13057, (315) 463-9291; **Endicott:** 112 Nanticoke Ave., P.O. Box 618, Endicott, NY 13760, (607) 754-3900; **Melville:** 1 Huntington Quadrangle, Suite 3C10, P.O. Box 2936, Melville, NY 11747, (516) 454-6600; **Pittsford:** 2851 Clover St., Pittsford, NY 14534, (716) 385-6770; **Poughkeepsie:** 385 South Rd., Poughkeepsie, NY 12601, (914) 473-2900.

NORTH CAROLINA: Charlotte: 8 Woodlawn Green, Woodlawn Rd., Charlotte, NC 28210, (704) 527-0930; **Raleigh:** 2809 Highwoods Blvd., Suite 100, Raleigh, NC 27625, (919) 876-2725.

OHIO: Beachwood: 23408 Commerce Park Rd., Beachwood, OH 44122, (216) 464-6100; **Dayton:** Kingsley Bldg., 4124 Linden Ave., Dayton, OH 45432, (513) 258-3877.

OKLAHOMA: Tulsa: 7615 East 63rd Place, 3 Memorial Place, Tulsa, OK 74133, (918) 250-0633.

OREGON: Beaverton: 6700 SW 105th St., Suite 110, Beaverton, OR 97005, (503) 643-6758.

PENNSYLVANIA: Ft. Washington: 260 New York Dr., Ft. Washington, PA 19034, (215) 643-6450; **Coraopolis:** 420 Rouser Rd., 3 Airport Office Park, Coraopolis, PA 15108, (412) 771-8550.

PUERTO RICO: Hato Rey: Mercantil Plaza Bldg., Suite 505, Hato Rey, PR 00919, (809) 753-8700.

TEXAS: Austin: 12501 Research Blvd., P.O. Box 2909, Austin, TX 78723, (512) 250-7655; **Richardson:** 1001 E. Campbell Rd., Richardson, TX 75080, (214) 680-5082; **Houston:** 9100 Southwest Frwy., Suite 237, Houston, TX 77036, (713) 778-6592; **San Antonio:** 1000 Central Parkway South, San Antonio, TX 78232, (512) 496-1779.

UTAH: Murray: 5201 South Green SE, Suite 200, Murray, UT 84107, (801) 266-8972.

VIRGINIA: Fairfax: 3001 Prosperity, Fairfax, VA 22031, (703) 849-1400.

WASHINGTON: Redmond: 5010 148th NE, Bldg B. Suite 107, Redmond, WA 98052, (206) 881-3080.

WISCONSIN: Brookfield: 450 N. Sunny Slope, Suite 150, Brookfield, WI 53005, (414) 785-7140.

CANADA: Nepean: 301 Moodie Drive, Mallorn Center, Nepean, Ontario, Canada, K2H9C4, (613) 726-1970. **Richmond Hill:** 280 Centre St. E., Richmond Hill L4C1B1, Ontario, Canada (416) 884-9181; **St. Laurent:** Ville St. Laurent Quebec, 9460 Trans Canada Hwy., St. Laurent, Quebec, Canada H4S1R7. (514) 334-3635.

ARGENTINA: Texas Instruments Argentina S.A.I.C.F.: Esmeralda 130, 15th Floor, 1035 Buenos Aires, Argentina, 1 + 394-3008.

AUSTRALIA (& NEW ZEALAND): Texas Instruments Australia Ltd.: 6-10 Talavera Rd., North Ryde (Sydney), New South Wales, Australia 2113, 2 + 887-1122; 5th Floor, 418 St. Kilda Road, Melbourne, Victoria, Australia 3004, 3 + 267-4677; 171 Philip Highway, Elizabeth, South Australia 5112, 8 + 255-2066.

AUSTRIA: Texas Instruments Ges.m.b.H.: Industriestrabe B/16, A-2345 Brunn/Gebirge, 2236-846210.

BELGIUM: Texas Instruments N.V. Belgium S.A.: Mercure Centre, Raketstraat 100, Rue de la Fusee, 1130 Brussels, Belgium, 2/720.80.00.

BRAZIL: Texas Instruments Electronicos do Brasil Ltda.: Rua Paes Leme, 524-7 Andar Pinheiros, 05424 Sao Paulo, Brazil, 0815-6166.

DENMARK: Texas Instruments A/S, Mairelundvej 46E, DK-2730 Herlev, Denmark, 2 - 91 74 00.

FINLAND: Texas Instruments Finland OY: Teollisuuskatu 19D 00511 Helsinki 51, Finland, (90) 701-3133.

FRANCE: Texas Instruments France: Headquarters and Prod. Plant, BP 05, 06270 Villeneuve-Loubet, (93) 20-01-01; Paris Office, BP 67 8-10 Avenue Morane-Saulnier, 78141 Velizy-Villacoublay, (3) 946-97-12; Lyon Sales Office, L'Oree D'Ecully, Batiment B, Chemin de la Forestiere, 69130 Ecully, (7) 833-04-40; Strasbourg Sales Office, Le Sebastopol 3, Quai Kleber, 67055 Strasbourg Cedex, (88) 22-12-66; Rennes, 23-25 Rue du Puits Mauger, 35100 Rennes, (99) 31-54-86; Toulouse Sales Office, Le Peripole—2, Chemin du Pigeonnier de la Cepiere, 31100 Toulouse, (61) 44-18-19; Marseille Sales Office, Noilly Paradis—146 Rue Paradis, 13006 Marseille, (91) 37-25-30.

TEXAS INSTRUMENTS

Creating useful products and services for you

GERMANY (Fed. Republic of Germany): Texas Instruments Deutschland GmbH: Haggertystrasse 1, D-8050 Freising, 8161 + 80-4591; Kurfuerstendamm 195/196, D-1000 Berlin 15, 30 + 882-7365; III, Hagen 43/Kibbelstrasse, .19, D-4300 Essen, 201-24250; Frankfurter Allee 6-8, D-6236 Eschborm 1, 06196 + 8070; Hamburgerstrasse 11, D-2000 Hamburg 76, 040 + 220-1154, Kirchhorsterstrasse 2, D-3000 Hannover 51, 511 + 648021; Maybachstrabe 11, D-7302 Ostfildern 2-Nelingen, 711 + 547001; Mixikoring 19, D-2000 Hamburg 60, 40 + 637 + 0061; Postfach 1309, Roonstrasse 16, D-5400 Koblenz, 261 + 35044.

HONG KONG (+ PEOPLES REPUBLIC OF CHINA): Texas Instruments Asia Ltd., 8th Floor, World Shipping Ctr., Harbour City, 7 Canton Rd., Kowloon, Hong Kong, 3 + 722-1223.

IRELAND: Texas Instruments (Ireland) Limited: Brewery Rd., Stillorgan, County Dublin, Eire, 1 831311.

ITALY: Texas Instruments Semiconduttori Italia Spa: Viale Delle Scienze, 1, 02015 Cittaducale (Rieti), Italy, 746 694.1; Via Salaria KM 24 (Palazzo Cosma), Monterotondo Scalo (Rome), Italy, 6 + 9003241; Viale Europa, 38-44, 20093 Cologno Monzese (Milano), 2 2532541; Corso Svizzera, 185, 10100 Torino, Italy, 11 774545; Via J. Barozzi 6, 40100 Bologna, Italy, 51 355851.

JAPAN: Texas Instruments Asia Ltd.: 4F Aoyama Fuji Bldg., 6-12, Kita Aoyama 3-Chome, Minato-ku, Tokyo, Japan 107, 3-498-2111; Osaka Branch, 5F, Nissho Iwai Bldg., 30 Imabashi 3- Chome, Higashi-ku, Osaka, Japan 541, 06-204-1881; Nagoya Branch, 7F Daini Toyota West Bldg., 10-27, Meieki 4-Chome, Nakamura-ku Nagoya, Japan 450, 52-583-8691.

KOREA: Texas Instruments Supply Co.: 3rd Floor, Samon Bldg., Yuksam-Dong, Gangnam-ku, 135 Seoul, Korea, 2 + 462-8001.

MEXICO: Texas Instruments de Mexico S.A.: Mexico City, AV Reforma No. 450 — 10th Floor, Mexico, D.F., 06600, 5 + 514-3003.

MIDDLE EAST: Texas Instruments: No. 13, 1st Floor Mannai Bldg., Diplomatic Area, P.O. Box 26335, Manama Bahrain, Arabian Gulf, 973 + 274681.

NETHERLANDS: Texas Instruments Holland B.V., P.O. Box 12995, (Bullewijk) 1100 CB Amsterdam, Zuid-Oost, Holland 20 + 5602911.

NORWAY: Texas Instruments Norway A/S: PB106, Refstad 131, Oslo 1, Norway, (2) 155090.

PHILIPPINES: Texas Instruments Asia Ltd.: 14th Floor, Ba- Lepanto Bldg., 8747 Paseo de Roxas, Makati, Metro Manila, Philippines, 2 + 8188987.

PORTUGAL: Texas Instruments Equipamento Electronico (Portugal), Lda.: Rua Eng. Frederico Ulrich, 2650 Moreira Da Maia, 4470 Maia, Portugal, 2-948-1003.

SINGAPORE (+ INDIA, INDONESIA, MALAYSIA, THAILAND): Texas Instruments Asia Ltd.: 12 Lorong Bakar Batu, Unit 01-02, Kolam Ayer Industrial Estate, Republic of Singapore, 747-2255.

SPAIN: Texas Instruments Espana, S.A.: C/Jose Lazaro Galdiano No. 6, Madrid 16, 1/458.14.58.

SWEDEN: Texas Instruments International Trade Corporation (Sverigefilialen): Box 39103, 10054 Stockholm, Sweden, 8 - 235480.

SWITZERLAND: Texas Instruments, Inc., Reidstrasse 6, CH-8953 Dietikon (Zuerich) Switzerland, 1-740 2220.

TAIWAN: Texas Instruments Supply Co.: Room 903, 205 Tun Hwan Rd., 71 Sung-Kiang Road, Taipei, Taiwan, Republic of China, 2 + 521-9321.

UNITED KINGDOM: Texas Instruments Limited: Manton Lane, Bedford, MK41 7PA, England, 0234 67466; St. James House, Wellington Road North, Stockport, SK4 2RT, England, 61 + 442-7162. BK