Microsoft® Link

Linker Utility

for 8086 and 8088 Microprocessors

Microsoft Corporation

System Requirements

The Microsoft Linker Utility requires:

50K bytes of memory minimum:
- 40K bytes for code and data
- 10K bytes for run space

Disk drive(s):

1 disk drive if and only if output is sent to the same physical disk from which the input was taken. MS-LINK does not allow time to swap disks during operation on a one-drive configuration. Therefore, two disk drives is a more practical configuration.

Contents

CHAPTER 1

INTRODUCTION

The Microsoft Linker Utility (MS-LINK) is a relocatable linker designed to link separately produced modules of 8086 object code. The input to MS-LINK is a subset of the Intel object module format standard.

MS-LINK prompts you for all MS-LINK commands. Your answers to these prompts are the commands for MS-LINK.

The output file from MS-LINK (a Run file) is not bound to specific memory addresses and, therefore, can be loaded and executed at any convenient address by the operating system.

MS-LINK uses a dictionary-indexed library search method, which substantially reduces link time for sessions involving library searches.

MS-LINK is able to link files totaling 1 megabyte.

NOTE

This manual describes some of the technical information about MS-LINK. It is recommended that this manual be read in conjunction with Chapter 9, "The Linker Program (MS-LINK)," in the MS-DOS User's Guide.

1.1 OVERVIEW OF MS-LINK OPERATION

MS-LINK performs the following steps to combine object modules and produce a Run file:

1. Reads segments in object modules
2. Assigns addresses to segments
3. Assigns public symbol addresses
4. Reads data in segments
5. Reads all relocation references in object modules
6. Resolves references and determines relocation information
7. Outputs a Run file (executable image) and relocation information

As it combines modules, MS-LINK can search multiple library files for definitions of any external references left unresolved.

MS-LINK also produces a List file that shows external references resolved and any error messages.

MS-LINK uses available memory as much as possible. When available memory is exhausted, MS-LINK then creates a disk file (VM.TMP) to use as temporary memory.

The following figure illustrates the MS-LINK operation.

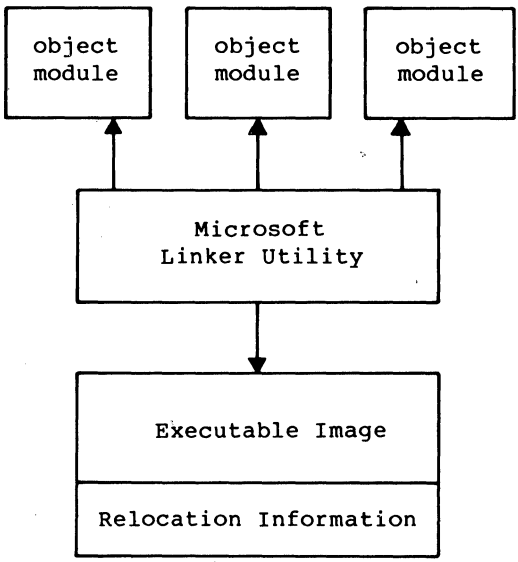


Figure 1. MS-LINK Operation

The executable image contains the concatenated object modules that make the Run file. The relocation information is a list of long addresses that must change when the executable image is relocated in memory. Refer to Section 1.7.4, "Long References," for an explanation of long addresses.

1.2 DEFINITIONS

The following terms describe the functioning of MS-LINK. An understanding of the concepts that define these terms will provide a basic understanding of the way MS-LINK works. Refer to the MS-DOS User's Guide for more information on these definitions.

1. Segment

   A segment is a contiguous area of memory up to 64K bytes in length. A segment may be located anywhere in 8086 memory. The contents of a segment are addressed by a canonical frame address and offset within that frame. Refer to Section 1.5, "Segment Addresses," for further discussion of canonical frames.

2. Group

   A group is a collection of segments that fit within 64K bytes of memory. The segments are named to the group by the assembler, by the compiler, or by you. You give the group name in the assembly language program. For the high-level languages (BASIC, FORTRAN, COBOL, Pascal), the naming is carried out by the compiler.

   The group is used for addressing segments in memory. Each group is addressed by a common canonical frame. This frame is the lowest canonical frame of the segments that belong to the group. It is a usual practice in assembler and higher languages for the canonical frame address to be contained in a segment register. MS-LINK checks to see that the object modules of a group meet the 64K-byte constraint.

3. Class

   A class is a collection of segments. The naming of segments to a class controls the order and relative placement of segments in memory. You give the class name in the assembly language program. For the high-level languages (BASIC, FORTRAN, COBOL, Pascal), the naming is carried out by the compiler. The segments are named to a class at compile time or assembly time.

   The segments of a class are loaded into memory contiguously. The segments are ordered within a class in the order the Linker encounters the segments in the object files. One class precedes another in memory only if a segment

for the first class precedes all segments for the second class in the input to MS-LINK. Classes may be loaded across 64K-byte boundaries. Groups may span classes.

4. Alignment

Alignment refers to certain segment boundaries. These can be byte, word, or paragraph boundaries.

Byte Alignment: A segment can begin on any byte boundary.

Word Alignment: The beginning address of a segment must occur on an even address.

Paragraph Alignment: The beginning address of a segment must occur on a segment (16-byte) boundary.

5. Combine Type

A combine type is an attribute of a segment; it tells the Linker how to combine segments of a like name or it relays other information about the properties of a segment. Combine types are: stack, public, private, and common. The way MS-LINK arranges these combine types is discussed in the next section.

CHAPTER 2

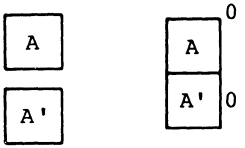
MS-LINK TECHNICAL INFORMATION

2.1 HOW MS-LINK COMBINES AND ARRANGES SEGMENTS

MS-LINK works with four combine types, which are declared in the source module for the assembler or compiler: private, public, stack, and common. The memory combine type available in Microsoft's Macro Assembler is processed the same as public combine type. MS-LINK does not automatically place memory combine type as the highest segments (as defined in the Intel standard).

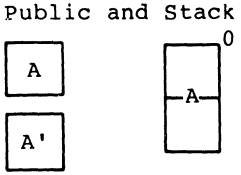
MS-LINK arranges these combine types as follows:

Private



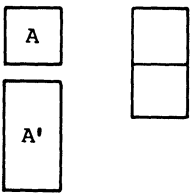
Private segments are loaded separately and remain separate. They may be physically (but not logically) contiguous even if the segments have the same name. Each private segment has its own canonical frame.

Public and Stack



Public and stack segments of the same name and class name are loaded contiguously. Offset is from the beginning of the first segment loaded through the last segment loaded. There is only one canonical frame for all public segments of the same name and class name. Stack and memory combine types are treated the same as public. However, the Stack Pointer is set to the last address of the first stack segment.

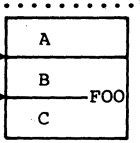
Common



Common segments of the same name and class name are loaded overlapping one another. There is only one canonical frame for all common segments of the same name. The length of the common area is the length of the longest segment.

Placing segments in a group in the assembler provides offset addressing of items from a single canonical frame for all segments in that group.

DS:DGROUP--->XXXX0H.....0 -- relative offset



Any number of other segments may intervene between segments of a group. Thus, the offset of FOO may be greater than the size of segments in the group combined, but no larger than 64K.

An operand of DGROUP:FOO in assembly language returns the offset of FOO from the beginning of the first segment (segment A here).

Segments are partitioned by declared class names. The Linker loads all the segments belonging to the first class name encountered, then loads all the segments of the next class name encountered, and so on until all classes have been loaded.

If your program contains:

```
A  SEGMENT 'FOO'
B  SEGMENT 'BAZ'
C  SEGMENT 'BAZ'
D  SEGMENT 'ZOO'
E  SEGMENT 'FOO'
```

They will be loaded as:

```
'FOO'
   A
   E
'BAZ'
   B
   C
'ZOO'
   D
```

If you are writing assembly language programs, you can control the order of classes in memory by writing a dummy module and listing it first after the MS-LINK Object Modules: prompt. The dummy module declares segments into classes in the order you want the classes loaded.

WARNING

Do not use this method with BASIC, COBOL, FORTRAN, or Pascal programs. Allow the compiler and the Linker to perform their tasks in the normal way.

Example:

```
A       SEGMENT 'CODE'
A       ENDS
B       SEGMENT 'CONST'
B       ENDS
C       SEGMENT 'DATA'
C       ENDS
D       SEGMENT STACK   'STACK'
D       ENDS
E       SEGMENT 'MEMORY'
E       ENDS
```

Make sure you declare all classes to be used in your program in this module. If you do not, you lose absolute control over the ordering of classes.

Also, if you want memory combine type to be loaded as the last segments of your program, you can use this method. Simply add MEMORY between SEGMENT and 'MEMORY' in the E segment line above. Note, however, that these segments are loaded last only because you imposed this control on them, not because of any inherent capability in the Linker or assembler operations.

2.2 SEGMENT ADDRESSES

The 8086 must be able to address all segments in memory. Any 20-bit number can be addressed. The 8086 represents these numbers as two 16-bit numbers; for example, HEX F:12. The F represents a canonical frame address and the 12 is the offset. The canonical frame address is the largest frame address or segment address that can contain the segment. An offset is the segment's location, offset from the beginning of the canonical frame.

The Linker recognizes a segment by its canonical frame address and its offset within the frame.

To convert the segmented address F:12 to a 20-bit number, shift the frame address left 4 bits, and add the offset. Using the above example:

```
      F0
+     12
      ---

F:12 = 102   (20-bit address)
```

2.3 HOW MS-LINK ASSIGNS ADDRESSES

To assign addresses to segments, MS-LINK:

1. Orders each segment by segment and class name.

2. On the basis of the alignment and size of each segment (assuming they are contiguous), the Linker assigns a frame address and an offset to each segment. This information is used for resolving relocatable references. The addresses start at 0:0.

2.4 RELOCATION FIXUPS

MS-LINK performs relocation fixups (i.e., resolves) on four types of references in object modules:

Short

Near Self-Relative

Near Segment-Relative

Long

These references and the Linker's fixups are described in the next sections.

2.4.1 Short References

Short references are all self-relative. The implication is that the frame address of the target and source frames are the same. MS-LINK will generate the fixup error message

Fixup offset exceeds field width

under the following conditions:

1. The target and source frame addresses are different.

2. The target is more than 128 bytes before or after the source frame address.

The resulting value of the short reference must fit into one signed byte.

2.4.2 Near Self-Relative References

When near self-relative references are used, the frame address of the target and source frames are the same. MS-LINK will generate the fixup error message under the following conditions:

1. The target and source frame addresses are different.

2. The target is more than 32K before or after the source frame address.

The resulting value of the near self-relative reference must fit into one signed word (16 bits).

2.4.3 Near Segment-Relative References

Given the target's canonical frame, another frame is specified (via an ASSUME directive or the : operator in assembly language; or via a high-level language convention). The target must be addressable through the canonical frame specified. MS-LINK will generate the fixup error message under the following conditions:

1. The offset of the target within the specified frame is greater than 64K or less than zero.

2. The beginning of the canonical frame of the target is not addressable by the specified frame.

The resulting value of a near segment-relative reference must be an unsigned 16-bit quantity.

2.4.4 Long References

Long references have a target and another frame (specified by an ASSUME or by a high-level language). The target must be addressable through the canonical frame specified. MS-LINK will generate the fixup error message under the following conditions:

1. The offset of the target within the specified frame is greater than 64K or less than zero.

2. The beginning of the canonical frame of the target is not addressable by the specified frame.

The resulting value of a long reference must be a frame address and an offset.

2.5 SAMPLE MS-LINK SESSION

The following example illustrates the type of information that is displayed during an MS-LINK session.

In response to the MS-DOS prompt (>), the system responds with the following messages and prompts. Answers to the prompts are underlined. Note that pathnames are supported under MS-DOS 2.0. Therefore, your answers to MS-LINK prompts can be full pathnames instead of filenames.

```
Microsoft Object Linker V.2.00
(C) Copyright 1982 by Microsoft Inc.

Object Modules [.OBJ]: IO SYSINIT
Run File [IO.EXE]:
List File [NUL.MAP]: IO /MAP
Libraries [.LIB]: ;
```

Notes:

1. By specifying /MAP, you can get both a sorted alphabetic listing and a sorted address listing of public symbols.

2. By responding PRN to the List File: prompt, you can redirect your output to the printer.

3. By specifying the /LINE switch, MS-LINK gives you a listing of all line numbers for all modules. (Note that the /LINE switch can generate a large volume of output.)

4. By pressing <RETURN> in response to the Libraries: prompt, an automatic library search is performed.

Once MS-LINK locates all libraries, the linker map displays a list of segments in the order of their appearance within the load module. The list might look like this:

```
Start     Stop      Length      Name
00000H    009ECH    09EDH       CODE
009F0H    01166H    0777H       SYSINITSEG
```

The information in the Start and Stop columns shows the 20-bit hex address of each segment relative to location zero. Location zero is the beginning of the load module.

Because the /MAP switch was used, MS-LINK displays the public symbols by name and value. For example:

```
ADDRESS         PUBLICS BY NAME
009F:0012       BUFFERS
009F:0005       CURRENT_DOS_LOCATION
009F:0011       DEFAULT_DRIVE
009F:000B       DEVICE_LIST
009F:0013       FILES
009F:0009       FINAL_DOS_LOCATION
009F:000F       MEMORY_SIZE
009F:0000       SYSINIT

ADDRESS         PUBLICS BY VALUE
009F:0000       SYSINIT
009F:0005       CURRENT_DOS_LOCATION
009F:0009       FINAL_DOS_LOCATION
009F:000B       DEVICE_LIST
009F:000F       MEMORY_SIZE
009F:0011       DEFAULT_DRIVE
009F:0012       BUFFERS
009F:0013       FILES
```

The addresses of the public symbols are in the frame:offset format, showing the location relative to zero as the beginning of the load module. In some cases, an entry may look like this:

```
780:A2
```

This entry appears to be the address of a load module that is almost one megabyte in size. Actually, the area being referenced is relative to a segment base that is pointing to a segment below the relative zero beginning of the load module. This condition produces a pointer that has effectively gone negative.

When MS-LINK has completed processing, the following message is displayed:

```
Program entry point at 0009F:0000
```

2.6 ERROR MESSAGES

All messages, except for the warning messages, cause the MS-LINK session to end. After you locate and correct a problem, you must rerun MS-LINK.

Messages appear in the List file and are displayed on the screen. If you direct the List file to CON, the error messages will not be displayed on the screen.

MS-LINK error messages are described in Chapter 9 of the MS-DOS User's Guide.

INDEX

ADDENDUM to the Microsoft MS-DOS Macro Assembler Manual

MS-LINK

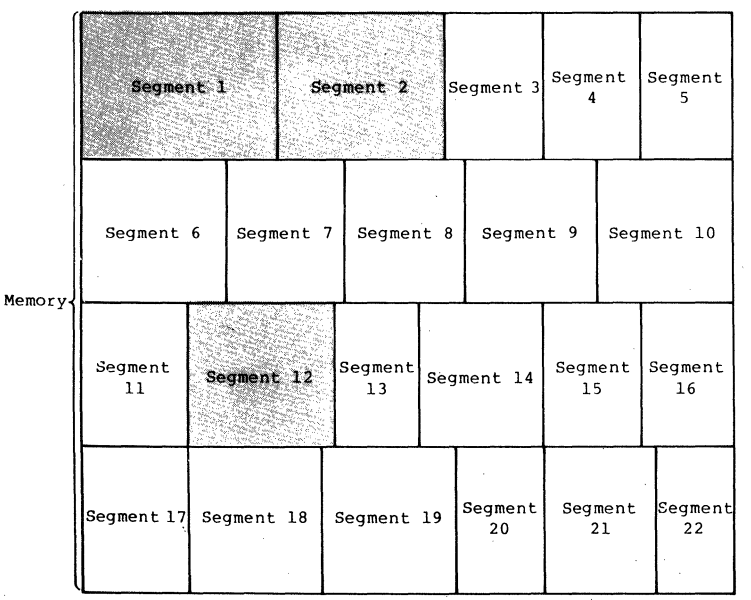
NOTE

References in the Macro Assembler Manual to the MS-DOS User's Guide refer to this addendum. You may want to place this addendum before the MS-LINK section in this manual.

1.0 DEFINITIONS

Some of the terms used in the MS-LINK section of this manual are explained below to help you understand how MS-LINK works. Generally, if you are linking object modules compiled from BASIC, Pascal, or a high-level language, you will not need to know these terms. If you are writing and compiling programs in assembly language, however, you will need to understand MS-LINK and the definitions described below.

In MS-DOS, memory can be divided into segments, classes, and groups. Figure 1 illustrates these concepts.



shaded area = a group (64K bytes addressable)

Figure 1. How Memory Is Divided

Example:

	Segment Name	Segment Class Name
Segment 1	PROG.1	CODE
Segment 2	PROG.2	CODE
Segment 12	PROG.3	DATA

Note that segments 1, 2, and 12 have different segment names but may or may not have the same segment class name. Segments 1, 2, and 12 form a group, with a group address of the lowest address of segment 1 (i.e., the lowest address in memory).

Each segment has a segment name and a class name. MS-LINK loads all segments into memory by class name, from the first segment encountered to the last. All segments assigned to the same class are loaded into memory contiguously.

During processing, MS-LINK references segments by their addresses in memory (where they are located). MS-LINK does this by finding groups of segments.

A group is a collection of segments that fit within a 64K byte area of memory. The segments do not need to be contiguous to form a group (see Figure 1). The address of any group is the lowest address of the segments in that group. At link time, MS-LINK analyzes the groups, then references the segments by the address in memory of that group. A program may consist of one or more groups.

If you are writing in assembly language, you may assign the group and class names in your program. In high-level languages (BASIC, COBOL, FORTRAN, Pascal), the naming is done automatically by the compiler.

2.0 FILES THAT MS-LINK USES

MS-LINK performs the following functions:

Works with one or more input files

Produces two output files

May create a temporary disk file

May be directed to search up to eight library files

For each type of file, you can give a three-part file specification. The format of MS-LINK file specifications is the same as that of a disk file:

[d:]<filename>[<.ext>]

where: d: is the drive designation. Permissible drive designations for MS-LINK are A: through O:. The colon is always required as part of the drive designation.

filename is any legal filename of one to eight characters.

.ext is a one- to three-character extension to the filename. The period is always required as part of the extension.

2.1 Input File Extensions

If no filename extensions are given in the input (object) file specifications, MS-LINK will recognize the following extensions by default:

.OBJ	Object
.LIB	Library

2.2 Output File Extensions

MS-LINK appends the following default extensions to the output (run and list) files:

.EXE	Run (may not be overridden)
.MAP	List (may be overridden)

2.3 VM.TMP (Temporary) File

MS-LINK uses available memory for the link session. If the files to be linked create an output file that exceeds available memory, MS-LINK will create a temporary file, name it VM.TMP, and put it on the disk in the default drive. If MS-LINK creates VM.TMP, it will display the message:

```
VM.TMP has been created.
Do not change diskette in drive, <d:>
```

Once this message has been displayed, you must not remove the disk from the default drive until the link session ends. If the disk is removed, the operation of MS-LINK will be unpredictable, and MS-LINK might display the error message:

```
Unexpected end of file on VM.TMP
```

The contents of VM.TMP are written to the file named following the Run File: prompt. VM.TMP is a working file only and is deleted at the end of the linking session.

WARNING

Do not use VM.TMP as a filename for any file. If you have a file named VM.TMP on the default drive and MS-LINK needs to create a VM.TMP file, MS-LINK will delete the VM.TMP already on disk and create a new VM.TMP. Thus, the contents of the previous VM.TMP file will be lost.

3.0 HOW TO START MS-LINK

MS-LINK requires two types of input: a command to start MS-LINK and responses to command prompts. In addition, seven switches control MS-LINK features. Usually, you will type all the commands to MS-LINK on the terminal keyboard. As an option, answers to the command prompts and any switches may be contained in a response file. Command characters can be used to assist you while giving commands to MS-LINK.

MS-LINK can be started in any of three ways. The first method is to type the commands in response to individual prompts. In the second method, you type all commands and switches on the line used to start MS-LINK. To start MS-LINK by the third method, you must create a response file that contains all the necessary commands, and then tell MS-LINK where that file is when you start MS-LINK.

Summary of Methods to Start MS-LINK

Method 1	LINK
Method 2	LINK <filenames>[/switches]
Method 3	LINK @<filespec>

3.1 Method 1: Prompts

To start MS-LINK with Method 1, type:

```
LINK
```

MS-LINK will be loaded into memory. MS-LINK will then display four text prompts that appear one at a time. You answer the prompts to command MS-LINK to perform specific tasks.

At the end of each line, you may type one or more switches, preceded by the switch character, a forward slash (/).

The command prompts are summarized below.

PROMPT	RESPONSES
Object Modules [.OBJ]:	List .OBJ files to be linked. They must be separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. There is no default; a response is required.
Run File [.EXE]:	Give filename for executable object code. The default is first-object-filename.EXE. (You cannot change the output extension.)
List File [NUL.MAP]:	Give filename for listing. The default is NUL.MAP.
Libraries [.LIB]:	List filenames to be searched, separated by blank spaces or plus signs (+). If a plus sign is the last character typed, the prompt will reappear. The default is to search for default libraries in the object modules. (Extensions will be changed to .LIB.)

3.2 Method 2: Command Line

To start MS-LINK using Method 2, type all commands on one line. The entries following LINK are responses to the command prompts. The entry fields for the different prompts must be separated by commas. Use the following syntax:

```
LINK <object-list>,<runfile>,<listfile>,<lib-list>[/switch]
```

where: object-list is a list of object modules, separated by plus signs.

runfile is the name of the file that receives the executable output.

listfile is the name of the file that receives the listing.

lib-list is a list of library modules to be searched.

/switch refers to optional switches, which may be placed following any of the response entries (just before any of the commas or after the <lib-list>, as shown).

To select the default for a field, simply type a second comma with no spaces between the two commas.

Example:

```
LINK
FUN+TEXT+TABLE+CARE/P/M,,FUNLIST,COBLIB.LIB
```

This command causes MS-LINK to be loaded; then the object modules FUN.OBJ, TEXT.OBJ, TABLE.OBJ, and CARE.OBJ are loaded. MS-LINK then pauses (as a result of using the /P switch). MS-LINK links the object modules when you press any key, and produces a global symbol map (the /M switch). MS-LINK then defaults to the FUN.EXE run file; creates a list file named FUNLIST.MAP; and searches the library file COBLIB.LIB.

3.3 Method 3: Response File

To start MS-LINK with Method 3, type:

```
LINK @<filespec>
```

where: filespec is the name of a response file. A response file contains answers to the MS-LINK prompts (shown in Method 1) and may also contain any of the switches. When naming a response file, use of the filename extension is optional. Method 3 permits the command that starts MS-LINK to be entered from the keyboard or within a batch file, without requiring you to make any further responses.

To use this option, you must create a response file containing several lines of text, each of which is the response to an MS-LINK prompt. The responses must be in the same order as the MS-LINK prompts discussed in Method 1. If desired, a long response to the Object Modules: or Libraries: prompt may be typed on several lines by using a plus sign (+) to continue the same response onto the next line.

Switches and command characters can be used in the response file the same way as they are used for responses typed on the terminal keyboard.

When the MS-LINK session begins, each prompt will be displayed in order with the responses from the response file. If the response file does not contain answers for all the prompts (in the form of filenames, the semicolon command character, or carriage returns), MS-LINK will display the prompt which does not have a response, then wait for you to type a legal response. When a legal response has been typed, MS-LINK continues the link session.

Example:

```
FUN TEXT TABLE CARE
/PAUSE/MAP
FUNLIST
COBLIB.LIB
```

This response file tells MS-LINK to load the four object modules named FUN, TEXT, TABLE, and CARE. MS-LINK pauses to permit you to swap disks before producing a public symbol map (see discussion under /PAUSE in the "Switches" section before using this feature). When you press any key, the output files will be named FUN.EXE and FUNLIST.MAP. MS-LINK will then search the library file COBLIB.LIB, and will use default settings for the switches.

4.0 COMMAND CHARACTERS

MS-LINK recognizes three command characters.

Plus sign — Use the plus sign (+) to separate entries and to extend the current line in response to the Object Modules: and Libraries: prompts. (A blank space may be used to separate object modules.) To type a large number of responses (each may be very long), type a plus sign/<RETURN> at the end of the line to extend it. If the plus sign/<RETURN> is the last entry following these two prompts, MS-LINK will prompt you for more module names. When the Object Modules: or Libraries: prompt appears again, continue to type responses. When all the modules to be linked and libraries to be searched have been listed, be sure the response line ends with a module name and a <RETURN> and not a plus sign/<RETURN>.

Example:

```
Object Modules [.OBJ]: FUN TEXT TABLE
CARE+<RETURN>
Object Modules [.OBJ]:
FOO+FLIPFLOP+JUNQUE+<RETURN>
Object Modules [.OBJ]: CORSAIR<RETURN>
```

Semicolon To select default responses to the remaining prompts, use a single semicolon (;) followed immediately by a carriage return at any time after the first prompt (Run File:). This feature saves time and overrides the need to press a series of <RETURN> keys.

NOTE

Once the semicolon has been entered (by pressing the <RETURN> key), you can no longer respond to any of the prompts for that link session. Therefore, do not use the semicolon to skip some prompts. To skip prompts, use the <RETURN> key.

Example:

```
Object Modules [.OBJ]: FUN TEXT TABLE
CARE<RETURN>
Run Module [FUN.EXE]: ;<RETURN>
```

No other prompts will appear, and MS-LINK will use the default values (including FUN.MAP for the list file).

<CONTROL-C> Use the <CONTROL-C> key to abort the link session at any time. If you type an erroneous response, such as the wrong filename or an incorrectly spelled filename, you must press <CONTROL-C> to exit MS-LINK, then you must restart MS-LINK. If the error has been typed but you have not pressed the <RETURN> key, you may delete the erroneous characters with the backspace key, but for that line only.

5.0 MS-LINK SWITCHES

The seven MS-LINK switches control various MS-LINK functions. Switches must be typed at the end of a prompt response, regardless of which method is used to start MS-LINK. Switches may be grouped at the end of any response, or may be scattered at the end of several. If more than one switch is typed at the end of a response, each switch must be preceded by a forward slash (/).

All switches may be abbreviated. The only restriction is that an abbreviation must be sequential from the first letter through the last typed; no gaps or transpositions are allowed. For example:

Legal	Illegal
/D	/DSL
/DS	/DAL
/DSA	/DLC
/DSALLOCA	/DSALLOCT

/DSALLOCATE

Using the /DSALLOCATE switch tells MS-LINK to load all data at the high end of the Data Segment. Otherwise, MS-LINK loads all data at the low end of the Data Segment. At runtime, the DS pointer is set to the lowest possible address to allow the entire DS segment to be used. Use of the /DSALLOCATE switch in combination with the default load low (that is, the /HIGH switch is not used) permits the user application to dynamically allocate any available memory below the area specifically allocated within DGroup, yet to remain addressable by the same DS pointer. This dynamic allocation is needed for Pascal and FORTRAN programs.

NOTE

Your application program may dynamically allocate up to 64K bytes (or the actual amount of memory available) less the amount allocated within DGroup.

/HIGH

Use of the /HIGH switch causes MS-LINK to place the run file as high as possible in memory. Otherwise, MS-LINK places the run file as low as possible.

IMPORTANT

Do not use the /HIGH switch with Pascal or FORTRAN programs.

/LINENUMBERS

The /LINENUMBERS switch tells MS-LINK to include in the list file the line numbers and addresses of the source statements in the input modules. Otherwise, line numbers are not included in the list file.

NOTE

Some compilers produce object modules that do not contain line number information. In these cases, of course, MS-LINK cannot include line numbers.

/MAP

/MAP directs MS-LINK to list all public (global) symbols defined in the input modules. If /MAP is not given, MS-LINK will list only errors (including undefined globals).

The symbols are listed alphabetically at the end of the list file. For each symbol, MS-LINK lists its value and its segment:offset location in the run file.

/PAUSE

The /PAUSE switch causes MS-LINK to pause in the link session when the switch is encountered. Normally, MS-LINK performs the linking session from beginning to end without stopping. This switch allows the user to swap disks before MS-LINK outputs the run (.EXE) file.

When MS-LINK encounters the /PAUSE switch, it displays the message:

```
About to generate .EXE file
Change disks <hit any key>
```

MS-LINK resumes processing when you press any key.

CAUTION

Do not remove the disk which will receive the list file, or the disk used for the VM.TMP file, if one has been created.

/STACK:<number>

number represents any positive numeric value (in hexadecimal radix) up to 65536 bytes. If a value from 1 to 511 is typed, MS-LINK will use 512. If the /STACK switch is not used for a link session, MS-LINK will calculate the necessary stack size automatically.

All compilers and assemblers should provide information in the object modules that allow the linker to compute the required stack size.

At least one object (input) module must contain a stack allocation statement. If not, MS-LINK will display the following error message:

```
WARNING: NO STACK STATEMENT
```

/NO

/NO is short for NODEFAULTLIBRARYSEARCH. This switch tells MS-LINK to not search the default (product) libraries in the object modules. For example, if you are linking object modules in Pascal, specifying the /NO switch tells MS-LINK to not automatically search the library named PASCAL.LIB to resolve external references.

6.0 ERROR MESSAGES

All errors cause the link session to abort. After the cause has been found and corrected, MS-LINK must be rerun. The following error messages are displayed by MS-LINK:

ATTEMPT TO ACCESS DATA OUTSIDE OF SEGMENT BOUNDS, POSSIBLY BAD OBJECT MODULE
: There is probably a bad object file.

BAD NUMERIC PARAMETER
: Numeric value is not in digits.

CANNOT OPEN TEMPORARY FILE
: MS-LINK is unable to create the file VM.TMP because the disk directory is full. Insert a new disk. Do not remove the disk that will receive the List.MAP file.

ERROR: DUP RECORD TOO COMPLEX
: The DUP record in the assembly language module is too complex. Simplify the DUP record in your assembly language program.

ERROR: FIXUP OFFSET EXCEEDS FIELD WIDTH
: An assembly language instruction refers to an address with a short instruction instead of a long instruction. Edit your assembly language source and reassemble.

INPUT FILE READ ERROR
: There is probably a bad object file.

INVALID OBJECT MODULE
: An object module(s) is incorrectly formed or incomplete (as when assembly is stopped in the middle).

SYMBOL DEFINED MORE THAN ONCE
: MS-LINK found two or more modules that define a single symbol name.

PROGRAM SIZE OR NUMBER OF SEGMENTS EXCEEDS CAPACITY OF LINKER
: The total size may not exceed 384K bytes, and the number of segments may not exceed 255.

REQUESTED STACK SIZE EXCEEDS 64K
Specify a size greater than or equal to 64K bytes with the /STACK switch.

SEGMENT SIZE EXCEEDS 64K
64K bytes is the addressing system limit.

SYMBOL TABLE CAPACITY EXCEEDED
Very many and/or very long names were typed, exceeding the limit of approximately 25K bytes.

TOO MANY EXTERNAL SYMBOLS IN ONE MODULE
The limit is 256 external symbols per module.

TOO MANY GROUPS
The limit is 10 groups.

TOO MANY LIBRARIES SPECIFIED
The limit is 8 libraries.

TOO MANY PUBLIC SYMBOLS
The limit is 1024 public symbols.

TOO MANY SEGMENTS OR CLASSES
The limit is 256 (segments and classes taken together).

UNRESOLVED EXTERNALS: <list>
The external symbols listed have no defining module among the modules or library files specified.

VM READ ERROR
This is a disk error; it is not caused by MS-LINK.

WARNING: NO STACK SEGMENT
None of the object modules specified contains a statement allocating stack space, although the /STACK switch was specified.

WARNING: SEGMENT OF ABSOLUTE OR UNKNOWN TYPE
There is a bad object module, or an attempt has been made to link modules that MS-LINK cannot handle (e.g., an absolute object module).

WRITE ERROR IN TMP FILE
No more disk space remains to expand the VM.TMP file.

WRITE ERROR ON RUN FILE
Usually, there is not enough disk space for the run file.

MICROSOFT™
10700 Northup Way, Bellevue, WA 98004

Software Problem Report

Name _____

Street _____

City _____ State _____ Zip _____

Phone _____ Date _____

Instructions

Use this form to report software bugs, documentation errors, or suggested enhancements. Mail the form to Microsoft.

Category

_____ Software Problem

_____ Software Enhancement

_____ Documentation Problem (Document #_____)

_____ Other

Software Description

Microsoft Product _____

Rev. _____ Registration # _____

Operating System _____

Rev. _____ Supplier _____

Other Software Used _____

Rev. _____ Supplier _____

Hardware Description

Manufacturer _____ CPU _____ Memory _____ KB

Disk Size _____" Density: Sides:

Single _____ Single _____

Double _____ Double _____

Peripherals _____

Problem Description

Describe the problem. (Also describe how to reproduce it, and your diagnosis and suggested correction.) Attach a listing if available.

Microsoft Use Only

Tech Support _____ Date Received _____

Routing Code _____ Date Resolved _____

Report Number _____

Action Taken: