

## **Section 2**

# **ZiLOG Z8 Linker User's Guide**

L  
i

## 13.1 Introduction

The ZiLOG Z8 Linker (**Z8LINK**) is a linker/locator which creates a single executable file from a set of object modules and object libraries. It is a linker because it links together object modules and resolves external references to public symbols. It is a locator because it allows you to locate where code and data will be stored in the target processor at runtime.

There are six major types of objects which are manipulated during the linking process:

- modules,
- libraries,
- address spaces,
- segments,
- groups, and
- copy segments.

Modules are created by assembling a file with the assembler or compiling a file with the compiler. Object libraries are collections of object modules created by the librarian. Segments, address spaces, groups, and copy segments are defined below.

### 13.1.1 Segments

Segments are named logical partitions of data or code that form a continuous block of memory. Segments with the same name residing in different modules are concatenated together at link time. Segments are assigned to an address space and may be relocatable or absolute. Relocatable segments may be randomly allocated by the linker while absolute segments are assigned a physical address within its address space.

### 13.1.2 Address spaces

Address spaces correspond to either a physical or logical block of memory on the target processor. For example, a Harvard architecture that divides memory into program and data stores each with its own set of addresses. In this case RAM and ROM may represent these address spaces. Logical address spaces are often used to partition a large contiguous block of memory in order to separate data and code on a RAM based system.

### 13.1.3 Groups

Groups are collections of logical address spaces. They are typically used for convenience of locating a set of address spaces.

### 13.1.4 Copy Segments

Copy segments are segments of initialized data that are copied into another address space by the linker. Though the code still addresses the original segment, a startup routine typically moves data from the copy segment into the original segment to initialize data prior to invoking a program.

## 13.2 Runtime Interface

The linker is invoked with the following command line:

**Z8LINK [options] <command list>**

or

**Z8LINK @<control file>**

The above command line can be entered with any mixture of lower or upper case characters. The options may be specified in any order but should precede the file(s) to be linked/located.

Modules are linked in the order you specify. Modules to be linked are **.OBJ** files created by the assembler, or **.LIB** files created by the librarian.

**<control file>** is the name of a file which contains legal commands and/or options. Commands within a control file may be broken up across several lines by using the backslash (\) line continuation character as the last character on the line. A command line can not exceed 3096 bytes. Lines in a control file that begin with a semicolon (;) are treated as comment lines.

## 13.3 Linker Options

The following section describes the options allowed by the linker. All options must be preceded by a - or /. Options may be abbreviated but the abbreviation must be unique; otherwise an error results.

### 13.3.1 COLOR

The **COLOR** option causes the linker to calculate memory overlays for relocatable segments through graph coloring. The default setting is for **COLOR** to be on. **NOCOLOR** will suppress graph coloring.

Format:

**-COLOR**

### 13.3.2 HELP

The **HELP** option displays the linker help screen.

Format:

**-HELP**

### 13.3.3 FORMAT

The **FORMAT** option sets the format of the hexfile output of the linker to SREC (Motorola S-records), IHEX (Intel Hex records), or OMF695 (IEEEG95 format).

Format:

**-FORMAT=<type>**

Example:

**-FORMAT = SREC**

### 13.3.4 NOCOLOR

The **NOCOLOR** option suppresses graph coloring for memory overlays. The default setting is **COLOR**.

Format:

**-NOCOLOR**

### 13.3.5 NOHEX

The **NOHEX** option prevents the executable file from being generated. The default is to generate an executable file.

Format:

**-NOHEX**

### 13.3.6 NOMAP

The **NOMAP** option causes the link map file to be suppressed. The default is to generate a link map file.

Format:

**-NOMAP**

### 13.3.7 NOWARN

The **NOWARN** option suppresses warning messages. The default is to generate a link map.

Format:

**-NOWARN**

### 13.3.8 WARN

The **WARN** option specifies that warning messages are to be generated. An optional warning file may be specified to redirect messages. The default setting is to generate warning messages on the screen and in the map file.

Format:

**-WARN**

**-WARN = <warn filename>**

Example:

**-WARN=warnfile.txt**

## 13.4 Linker Commands

The following describes the commands supported by the linker. These commands may be placed on the command line, or placed in a link control file.

### 13.4.1 <hexfile>=<module list>

This command defines the output file, object modules, and libraries involved in the linking process. The default extension is **.HEX** as specified by the **FORMAT** option.

<modulelist> is a list of object modules or libraries to be linked together to create the output file.

Example:

```
Z8LINK output=module1,library1
```

### 13.4.2 CHANGE

The **CHANGE** command is used to rename a **group**, address **space**, or **segment**. The **CHANGE** command may also be used to move an address **space** to another **group** or to move a **segment** to another address **space**.

Format:

```
CHANGE <name> = <newname>
```

<name> may be a **group**, address **space**, or **segment**.

<newname> is the new name to be used in renaming a **group**, address **space**, or **segment**; or the name of the **group** where an address **space** is to be moved; or the name of the address **space** where a **segment** is to be moved.

Example:

```
CHANGE romseg=RAM
```

### 13.4.3 COPY

The **COPY** command is used to make a copy of a **segment** into a specified address **space**. This is most often used to make a copy of initialized RAM in ROM so that it may be initialized at runtime.

Format:

**COPY <segment> <name>[at<expression>]**

<segment> may only be a **segment**.

<name> may only be an address **space**.

Example:

**COPY ram\_data ROM**  
**COPY ram\_data ROM at \$100000**

#### 13.4.4 DEFINE

The **DEFINE** command creates a user defined public symbol at link time.

Format:

**DEFINE <symbol name> = <expression>**

<symbol name> is the name assigned to the public symbol.

<expression> is the value assigned to the public symbol.

Example:

**DEFINE copy\_size = copy top of data\_seg - copy base of data\_seg**

#### 13.4.5 GROUP

The **GROUP** command provides a method of collecting multiple address **spaces** into a single manageable entity.

Format:

**GROUP <groupname> = <name>[,<name>]**

<groupname> may only be a **group**.

<name> may only be an address **space**.

Example:

**GROUP MEMORY = ROM, RAM**



### 13.4.6 LOCATE

The **LOCATE** command specifies the address where a **group**, address **space**, or **segment** is to be located. If multiple locates are specified for the same **space**, the last specification takes precedence. A warning is flagged on a **Locate** of an absolute **segment**.

The **LOCATE** of a **segment** overrides the **LOCATE** of an address **space**. A **LOCATE** will not override an absolute **segment**.

Format:

**LOCATE <name> AT <expression>**

**<name>** is either a **group**, address **space**, or **segment**.

**<expression>** is the address to begin loading.

Example:

**LOCATE ROM AT \$10000**

### 13.4.7 MAP

The **MAP** command causes the linker to output a link map file. The link map file contains the location of address **spaces**, **segments** and **symbols**. The default is to create a link map file. **NOMAP** will suppress the generation of a link map file.

Format:

**MAP**  
**MAP = <mapfile>**

**mapfile** has the same name as the executable file with the **.MAP** extension unless an optional **<mapfile>** is specified.

Example:

**MAP = myfile.map**

### 13.4.8 MAXLENGTH

The **MAXLENGTH** command causes a warning message to be issued if a **group**, address **space**, or **segment** is longer than the specified **size**.

The **RANGE** command will set address boundaries. The **MAXLENGTH** command allows further control of these boundaries.

Format:

**MAXLENGTH <name> <expression>**

<name> may be a **group**, address **space**, or **segment**.

<expression> size of the **MAXLENGTH** allowed.

Example:

**MAXLENGTH ROM \$FFFF**

### 13.4.9 ORDER

The **ORDER** command establishes a linking sequence and sets up a dynamic **RANGE** for contiguously mapped address **spaces**. The base of the **RANGE** of each consecutive address **space** is set to the top of its predecessor.

Format:

**ORDER <name>[,<name>]@**

<name> may be a **group**, address **space**, or **segment**. However, a **RANGE** is only established for an address **space**.

Example:

**ORDER ROM,RAM**

### 13.4.10 RANGE

The **RANGE** command is used to set the lower and upper bounds of a **group**, address **space**, or **segment**. If an address falls outside of the **RANGE** specified, a message is issued to that effect.

**NOTE:** White space must be used to separate the colon from the range command operands.

Format:

**RANGE** <name><expression> : <expression>  
(,<expression> : <expression> )@

<name> may be a **group**, address **space**, or **segment**.

The first <expression> marks the lower boundary for a specified address **RANGE**.

The second <expression> marks the upper boundary for a specified address **RANGE**.

Example:

**RANGE ROM \$1000 : \$1FFF,\$4000 : \$4FFF**

**NOTE:** If a **RANGE** is specified for a **segment** this range should be within any **RANGE** specified by that segment's address **space**.

### 13.4.11 SEARCHPATH

The **SEARCHPATH** command establishes an additional search path to be specified in locating files.

The search order is:

1. current directory
2. environment path
3. search path

Format:

**SEARCHPATH** = "<path>"

Example:

**SEARCHPATH** = "C:\Z8\rtl"

**NOTE:** The environment path is set by the DOS command:

**SET Z8=<path>**

which is usually placed in the **autoexec.bat** file.

### 13.4.12 SEQUENCE

The **SEQUENCE** command forces the linker to allocate a **group**, address **space**, or **segment** in the order specified.

Format:

**SEQUENCE <name>(,<name> )@**

<name> is either a **group**, address **space**, or **segment**.

Example:

**SEQUENCE code,data,bss**

**WARNING:** If the sequenced segments do not all receive space allocation in the first pass through the available address ranges then the sequence of segments is not maintained.

### 13.4.13 START ADDRESS

The **START ADDRESS** command is used to specify or override the start address of the executable file.

Format:

**START ADDRESS <expression>**

<expression> is the value of the start address.

Example:

**START ADDRESS c\_startup**

---

# Using the Linker with Libraries

---

## Chapter 14

When using the linker with libraries, you must tell the linker which libraries to search in order to resolve references to undefined symbols. The linker follows these steps when using libraries.

1. First, the linker makes a pass through all specified object modules and builds a table containing all public and external entries found in the modules.
2. Next the linker resolves as many external entries as possible using public entries from the object modules.
3. Finally, if any unresolved external entries remain, the linker attempts to resolve them by making multiple passes through the libraries.

The linker searches the libraries in the order of presentation (left to right). The linker continually searches the specified libraries until either all externals are resolved or a pass through the libraries resolves no more references.

Example:

```
Z8LINK out.hex = myfile ,mylib1,mylib2
```

In the above example, **out.hex** will contain the S-Records created from linking **myfile** with the modules contained in **mylib1** and **mylib2**.

L  
i  
n  
k  
e  
r

## 15.1 Syntax

An **expression** is any occurrence of an <expression> within any linker command. The syntax of linker expressions and their operators is as follows:

<expression>  
~<expression>  
(<expression>)  
<expression>+<expression>  
<expression>-<expression>  
<expression>\*<expression>  
<expression>/<expression>  
<expression>&<expression>  
<expression>|<expression>  
<expression>^<expression>  
<expression>>><expression>  
<expression><<<expression>  
base of <name>  
copy base of <name>  
copy top of <name>  
freemem of <name>  
highaddr of <name>  
length of <name>  
lowaddr of <name>  
top of <name>

## 15.2 Expressions

The following describe the operators and their operands which form expressions allowed by the linker.

### 15.2.1 ADD(+)

The **ADD** operator is used to perform addition of two expressions.

Format:

**<expression>+<expression>**

### 15.2.2 AND (&)

The **AND** operator is used to perform a bitwise **AND** of two expressions.

Format:

**<expression> & <expression>**

### 15.2.3 BASE

The **BASE** operator provides the lowest used address of a **group**, address **space**, or **segment** (excluding any segment copies when **<name>** is a **segment**). The value of **BASE** is treated as an expression value.

Format:

**BASE OF <name>**

**<name>** may be a **group**, address **space**, or **segment**.

### 15.2.4 COPY BASE

The **COPY BASE** operator provides the lowest used address of a copy **segment**, **group**, or address **space**. The value of **COPY BASE** is treated as an expression value.

Format:

**COPY BASE OF <name>**



<name> may be either a **group**, address **space**, or **segment**.

### 15.2.5 COPY TOP

The **COPY TOP** operator provides the highest used address of a copy **segment**, **group**, or address **space**. The value of **COPY TOP** is treated as an expression value.

Format:

**COPY TOP OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.6 DIVIDE (/)

The **DIVIDE** operator is used to perform division.

Format:

**<expression>/<expression>**

### 15.2.7 FREEMEM

The **FREEMEM** operator provides the lowest address of unallocated memory of a **group**, address **space**, or **segment**. The value of **FREEMEM** is treated as an expression value.

Format:

**FREEMEM OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.8 HIGHADDR

The **HIGHADDR** operator provides the highest possible address of a **group**, address **space**, or **segment**. The value of **HIGHADDR** is treated as an expression value.

Format:

**HIGHADDR OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.9 LENGTH

The **LENGTH** operator provides the length of a **group**, address **space**, or **segment**. The value of **LENGTH** is treated as an expression value.

Format:

**LENGTH OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.10 LOWADDR

The **LOWADDR** operator provides the lowest possible address of a **group**, address **space**, or **segment**. The value of **LOWADDR** is treated as an expression value.

Format:

**LOWADDR OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.11 MULTIPLY (\*)

The **MULTIPLY** operator is used to multiply two expressions.

Format:

**<expression>\*<expression>**

### 15.2.12 NUMERIC VALUES

Numeric constant values may be used as part of an expression. **Digit** is a numeric digit 0..9. **Hexdigit** is a digit 0..9 or A..F.

For hexadecimal numbers the format is:

**\$ <hexdigit>+**

For decimal numbers the format is:

**<digit>+**

### 15.2.13 OR (|)

The **OR** operator is used to perform a bitwise inclusive **OR** of two expressions.

Format:

**<expression>| <expression>**

### 15.2.14 SHL (<<)

The **SHL** operator is used to perform a left shift. The first expression is the value to be shifted. The second expression is the number of bits to the left the value is to be shifted.

Format:

**<expression><<<expression>**

### 15.2.15 SHR (>>)

The **SHR** operator is used to perform a right shift. The first expression is the value to be shifted. The second expression is the number of bits to the right the value is to be shifted.

Format:

**<expression>>><expression>**

### 15.2.16 SUBTRACT (-)

The **SUBTRACT** operator is used to subtract two expressions.

Format:

**<expression>-<expression>**

### 15.2.17 TOP

The **TOP** operator provides the highest allocated address of a **group**, address **space**, or **segment** (excluding any segment copies when **<name>** is a **segment**). The value of **TOP** is treated as an expression value.

Format:

**TOP OF <name>**

<name> may be a **group**, address **space**, or **segment**.

### 15.2.18 XOR (^)

The **XOR** operator is used to perform a bitwise exclusive **OR** on two expressions.

Format:

**<expression>^<expression>**

### 15.2.19 NOT (~)

The **NOT** operator is used to perform a one's complement of an expression.

Format:

**~<expression>**