Technical Article

How Code and Data are Placed in Memory Using ZDSII

TA000402-0404

Introduction

What are the commands and settings in ZDSII that determine where code is placed in memory? How does ZDSII handle the data initializations that are required in C? How does the Link Configuration feature affect code and data placement and how does this work with start-up code? This Technical Article answers these questions in the discussion that follows.

Note: For definitions and descriptions of the terms used in this Technical Article, please see the ZiLOG Developer Studio II–eZ80 User Manual (UM0144).

Discussion

The following ZDSII GUI controls are used to determine how and where code and data are placed and used on a target system. Each control is highlighted in *italic* typeface. All controls start from the **Project Settings** command in the **Project** menu. The **Project Settings** dialog box contains 5 tabs, as shown.

1. The **Target** tab in the **Project Settings** dialog—please see Figure 1.
   a. *Memory*

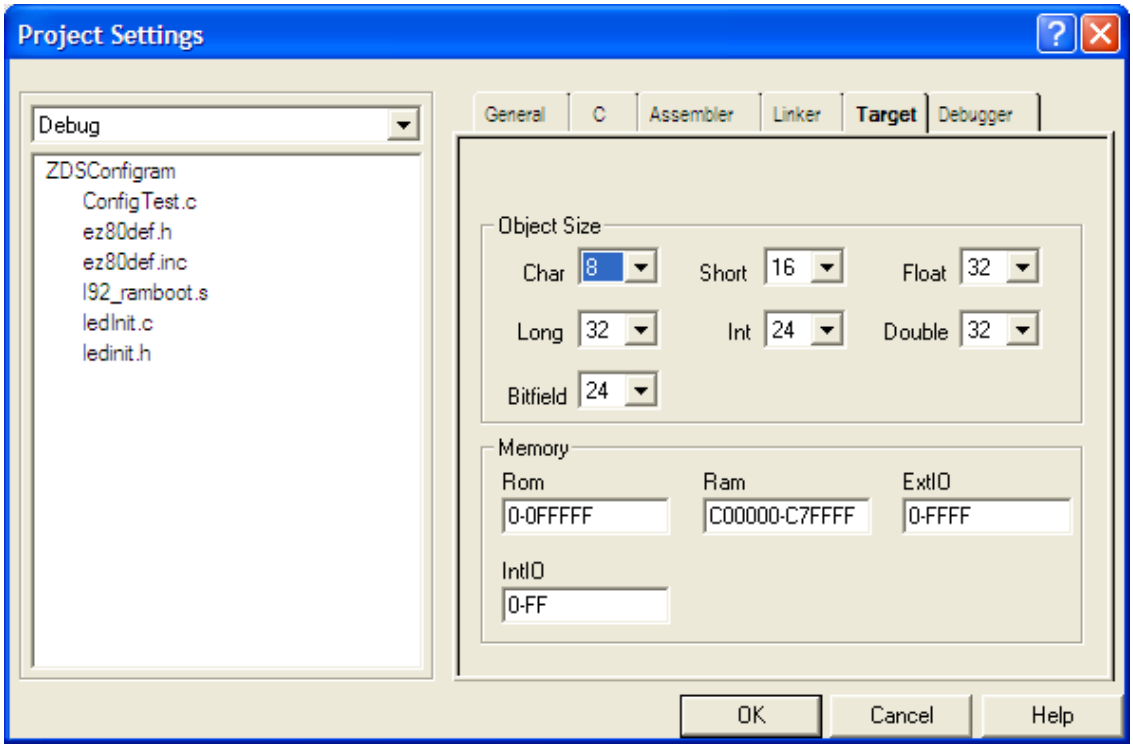

Figure 1. The Project Settings Dialog Box—Target Tab

2. The **Linker** tab in the **Project Settings** dialog.
   b. **General Category**—please see Figure 2.
      i. *Link Configuration*

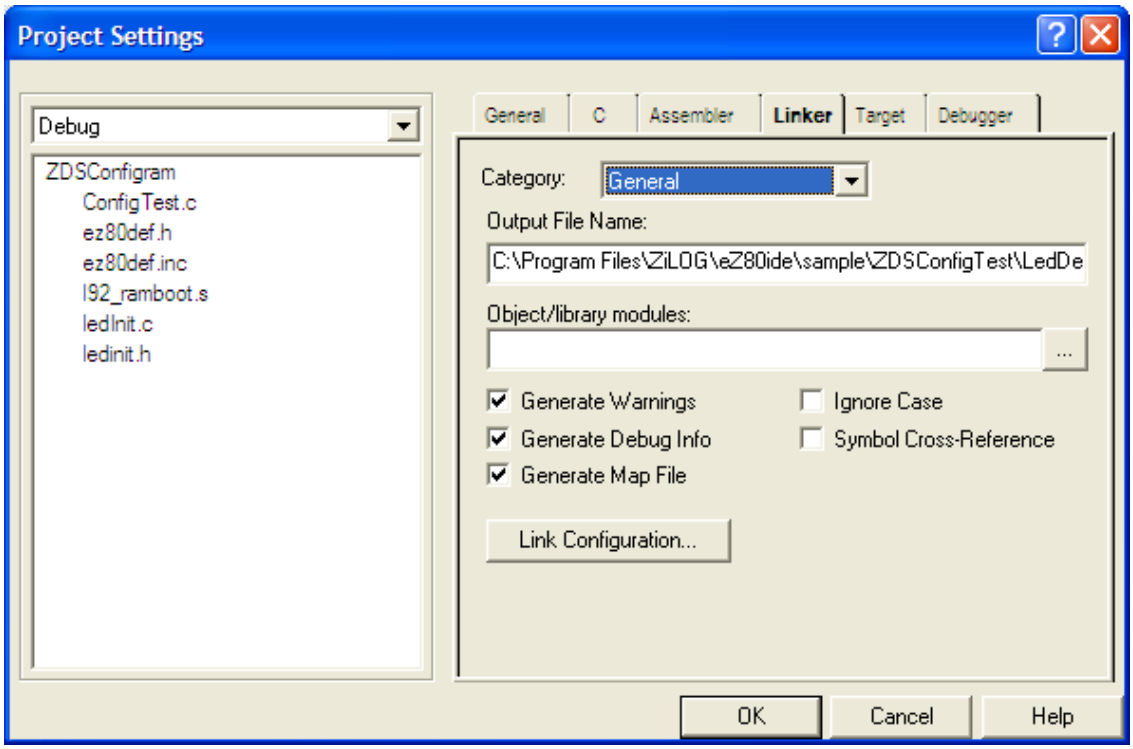

Figure 2. The Project Settings Dialog Box—Linker Tab, General

c. **Input Category**—please see Figure 3.
   i. *Startup Module*
   ii. *Link Control File*

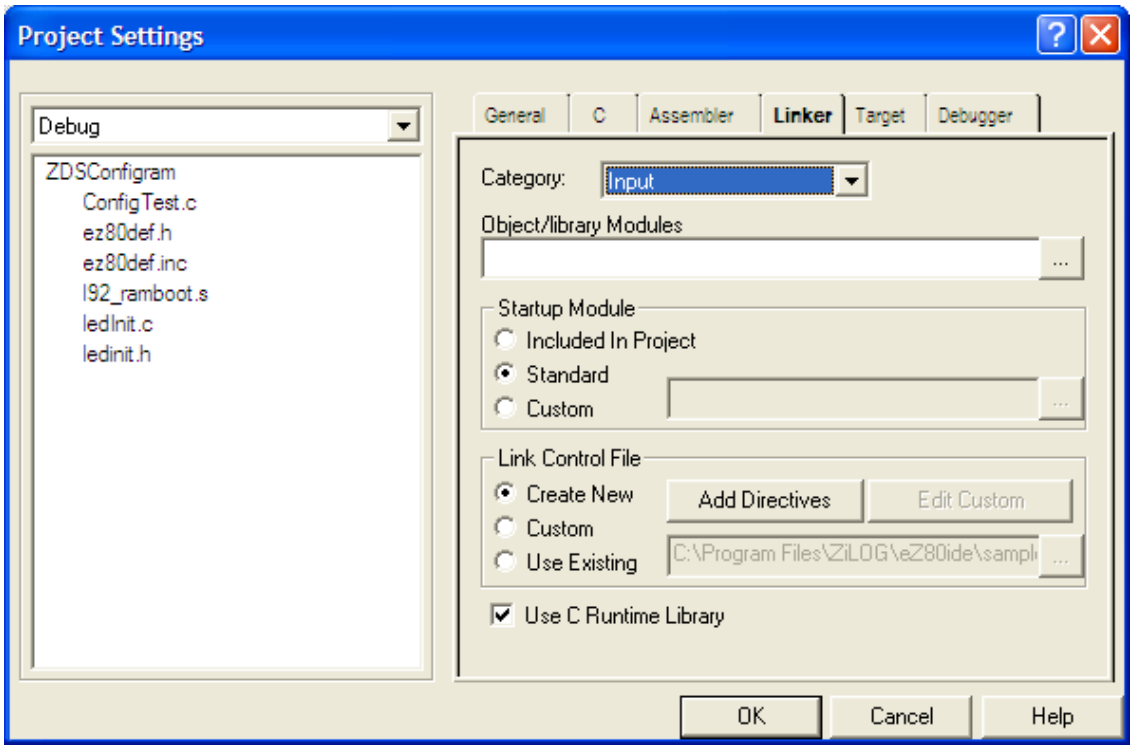

Figure 3. The Project Settings Dialog Box—Linker Tab, Input

d. **Output Category**—please see Figure 4.
   i. *Executable Format*

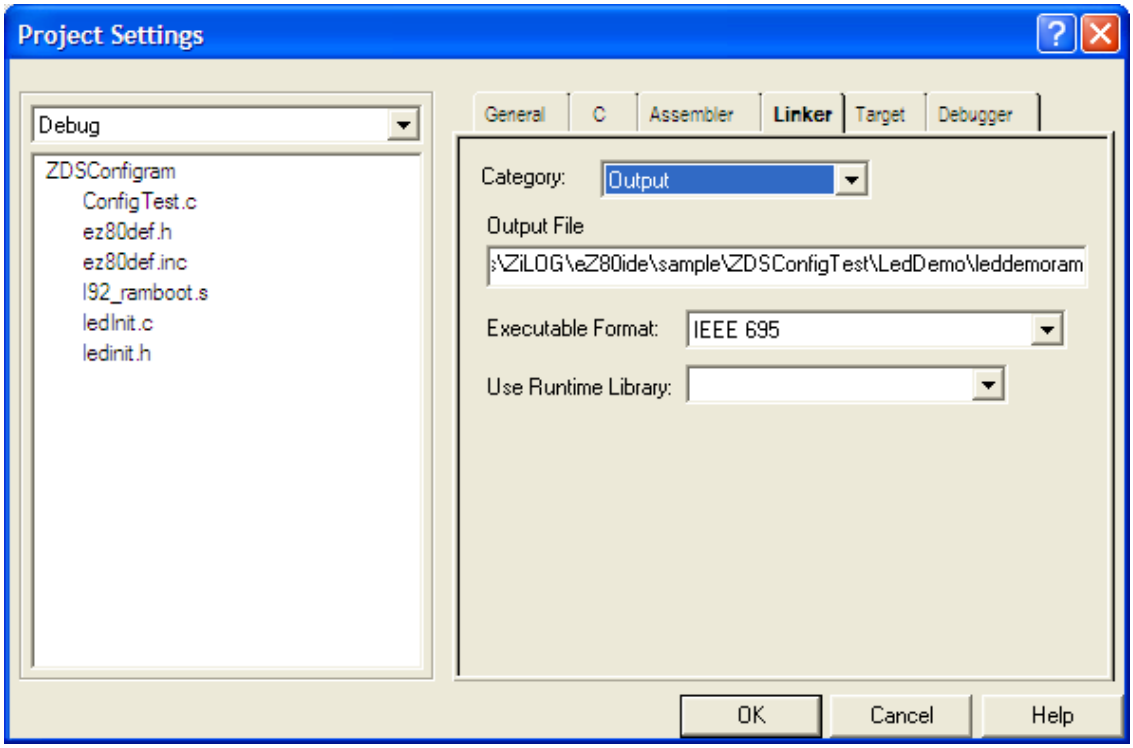

Figure 4. The Project Settings Dialog Box—Linker Tab, Output

3. The Debugger tab in the Project Settings dialog.
   a. *Initializations*—please see Figure 5.

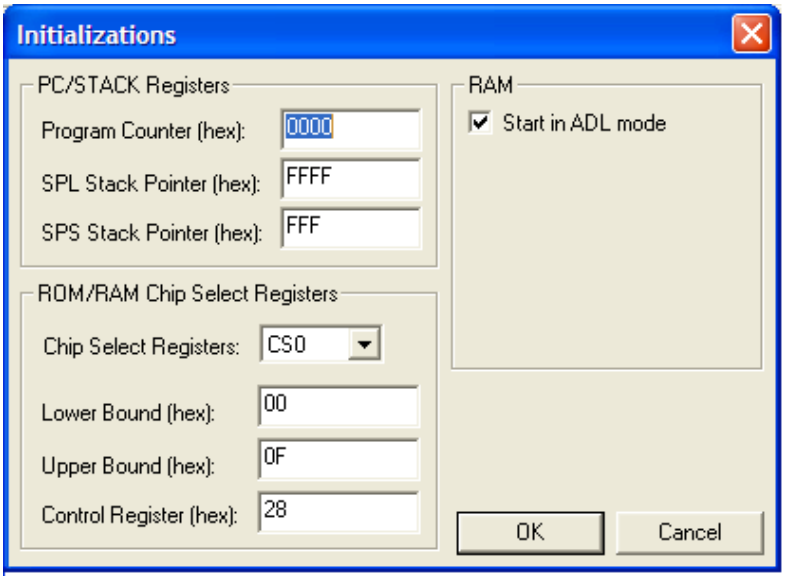

Figure 5. The Initializations Dialog Box

The following files are affected by the settings in the seven controls above:

Map File. This file is output from a build when the **Generate Map File** box is checked in the **General Category** of the **Linker** tab of the **Project Settings** dialog box. The file contains the same filename as the project name, with the addition of a *.map* extension. A text editor can be used to display this type of file.

Link Control File. This file is created when the **Project Settings** dialog box is closed and **Create New** is selected. The file contains the same filename as the project name, with the addition of a *.lnk* extension. (On some operating systems, the *.lnk* extension is not shown.) If **Custom** is selected, *_cst* is appended to the filename. A text editor can be used to display this type of file.

The following section describes how each of the above seven controls determines where code and data are placed and their effects as shown in the Map and Link Control files.

Note: Please note that the terms ROM and RAM found in the GUI displays, the Map files, the Link Control files, and the ZDSII documentation are used in different contexts. The adjective modifiers *Memory Label* or *labeled* and *Linker Designation* or *designated* are therefore included to keep the contexts separate.

Memory Control

The settings in the Memory Control group inform the linker of the physical address locations of the memory spaces labeled ROM and RAM as well as the I/O spaces labeled ExtIO and IntIO. Normally, memory

labeled ROM is specified with physical ROM memory locations and memory labeled RAM is specified with RAM memory locations, but not necessarily. The memory space labeled ROM is normally the memory space in which code is executed and initialized variables are stored. This memory can physically be RAM or ROM memory. The memory space labeled RAM is normally the memory space where noninitialized data (buffer space) is located and initialized data is copied to. This memory must physically be RAM memory.

Note: The specification of the physical address range for noncontiguous blocks is determined by separating the ranges with commas.

Link Configuration Control

The Link Configuration Control feature of ZDSII aids in the generation of the correct linker directives in the Link Control file for the selected configuration. There are four configurations provided by ZDSII. By default, the linker places the CODE and DATA COPY segments in memory designated as ROM and the BSS and DATA segments in memory designated as RAM. Each is based on the assumption that a startup module will copy the data segments—and in some cases the code segment—from the memory designated ROM to the memory designated RAM. A standard Startup Module is provided with ZDSII (see the Startup Module Control section on page 8). It is designed to work with all of the link configurations, although a user startup module can be used instead.

All RAM

The All RAM configuration assumes that the memory labeled ROM is specified with physical RAM memory locations and that the linker will consecutively locate the memory designated as RAM and ROM to this memory labeled ROM. It also assumes that the Startup Module will create a BSS section and move the DATA_COPY section to the DATA section. As in all configurations, the program code accesses data in the DATA section, which is always in memory designated as RAM. Please see Figure 6.

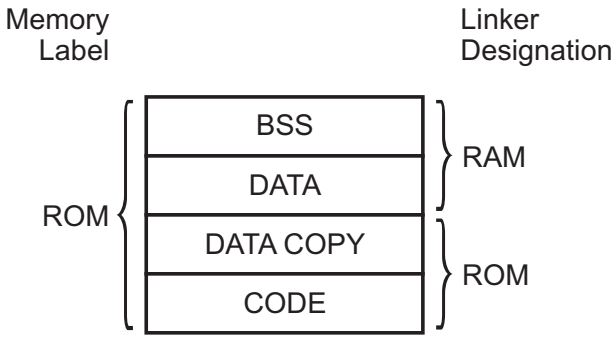

Figure 6. All RAM Configuration

Standard

The Standard configuration assumes that the memory labeled ROM is specified with physical ROM memory locations and that the memory labeled RAM is specified with physical RAM memory locations. In this

case, the linker locates the memory designated ROM to the memory labeled ROM. Likewise, the linker locates the memory designated RAM to the memory labeled RAM. This configuration also assumes that the Startup Module will create the BSS section in the memory space labeled RAM and transfer the data from DATA_COPY in the memory space labeled ROM to the DATA section in the memory space also labeled RAM. Please see Figure 7.

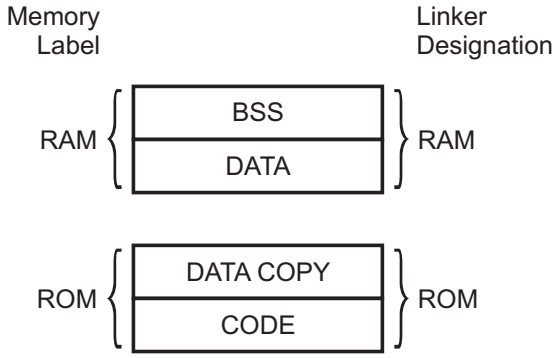

Figure 7. Standard Configuration

Copy to RAM

The Copy to RAM configuration uses the same assumptions as the Standard configuration with the additional assumption that the Startup Module will also copy code from the memory designated ROM to the memory designated RAM. The code copied is from the CODE_COPY section starting at `__low_romcode` to the memory designated RAM starting at `__low_code`. The length of this code is defined in the Link Control file by `len_code`. In this configuration, `__copy_code_to_ram` is also defined in the Link Control file for the Startup Module to determine if this transfer should take place. Please see Figure 8.

Note: In ZDSII versions 4.4.0 or newer, the Standard start-up file must be selected.

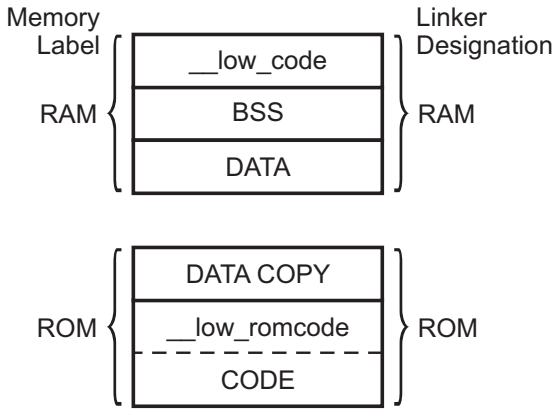

Figure 8. Copy To RAM Configuration

Custom

The Custom configuration assumes that the user will supply the Link Configuration File and the Startup Module. This configuration, however, creates a Link Control file that is the same as the one generated for the Standard configuration. This configuration offers the user something to start with and to modify as required. This Link Control file includes definitions for a startup module if the Standard Startup File is selected in the Input Category within the Linker Tab of the Projects Settings dialog box. Please see Figure 9.

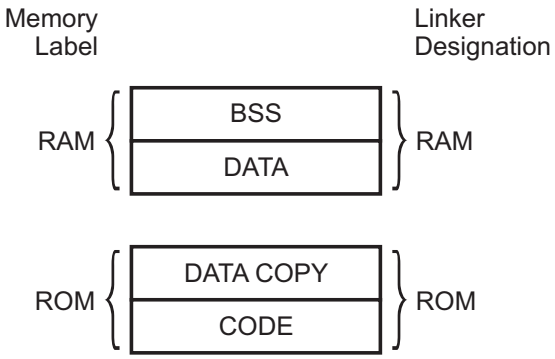

Figure 9. Custom Configuration

Startup Module Control

The Startup Module Control feature of ZDSII tells the linker if the Standard startup module should be included or that the user has provided one in the project. If the Standard Startup Module is selected, the linker includes the Standard Startup Module, *startup*, in the build. The Startup Module is taken from the ZDSII `rtl` subdirectory. The source code for this module is contained in the file `startup.asm` found in the `source` subdirectory to the `rtl` directory. Because only the object module is used in the project, the source code is not found in the project file list. The start-up object `startup.obj` does, however, show up in the Link Control file.

If a startup module is user-provided, then the user must include the startup module in the project list. This startup module must ensure that the code and data are moved to the correct locations as required, depending on the selected configuration.

Because the startup module, be it standard or user, is the first code executed after reset, it should be the first module in the sequence of objects placed in the CODE segment (as shown in the Link Control file). It contains the code that initializes the registers in the eZ80® device, initializes the BSS section in RAM, and copies the DATA_COPY segment in ROM to the DATA segment in RAM. The Standard Startup Module also checks if `__copy_code_to_ram` is defined as 0 or 1 in the Link Control file. If it is defined as 1, then the Standard Startup Module also copies the CODE segment to RAM using the `__low_romcode`, `__low_code`, and `__len_code` defines in the Link Control file.

Link Control File Control

The Link Control File Control is provided by ZDSII to select the Link Control File used in the build. There are three selections: Create New, Custom, and Use Existing. The Create New selection is normally used in the beginning phase of a project where settings are being changed. These changes are then captured in a new Link Control file to be used in subsequent builds. With Create New selected, the Add Directives button is activated to allow the user to add additional Linker commands.

A custom Link Control File Control based on the current link control file (Create New or Use Existing) is created with *_cst* added to the filename. This addition provides an easy way switch between two link control files according to user requirements. When Custom is selected, the Edit Custom button is activated to allow the user to edit the custom link control file.

If it is determined that a link control file from another source should be used (e.g, from another project), the Use Existing selection is used. The Browse button is activated to navigate to the existing link control file. This control can also be used to select link control files that have been renamed for safekeeping or for saving other configurations. This feature can be used to select the link control file for the build.

Link Control File

The Link Control file is created from the settings in ZDSII. It contains all the linker commands for the build. These linker commands include the linker options, the specification of the output format, the mapping of linker designated ROM and RAM memory to physical ROM and RAM locations; and the defines that can be used in the code execution. The linker commands are created based on the configuration selected and the project settings in ZDSII. It also contains comments indicating the parameters used for the compiler and assembler.

The linker groups compiled/assembled objects into the standard segments CODE, DATA, TEXT, STRSECT, BSS, and DATA_COPY and to the user-defined segments. Data, based on the type of data, is placed into one of the data segments DATA, TEXT, STRSECT, or BSS. Executable code is placed in the CODE segment.

These segments are mapped to either memory designated as ROM or RAM. The designated ROM and RAM memories are mapped to the physical memory locations labeled ROM and RAM.

The data segments are usually mapped to memory designated as RAM and the CODE segment is usually mapped to memory designated as ROM.

The mapping from designated memory to the memories labeled ROM or RAM is made by default or by linker commands. By default. ROM-designated memory is mapped to memory labeled as ROM, and RAM-designated memory is mapped to memory labeled as RAM. The linker CHANGE command changes the mapping.

The important linker commands in the Link Configuration File for each configuration are listed below.

Standard

The Link Configuration file for all configurations contain the RANGE linker command, as shown below.

```
RANGE ROM $0 :       $0FFFFF
```

```
RANGE RAM $C00000 :  $C7FFFF
```

The RANGE commands specify the address space for the Linker-designated ROM and RAM address spaces. It is taken from the physical address space specified in the ZDSII Project Settings dialog box. In the Standard configuration, because ROM is mapped to ROM and RAM is mapped to RAM, the ROM and RAM physical address space specified in the Project Settings is mapped respectively to the ROM and RAM linker-designated memory address space.

Another important linker command is COPY, as shown below.

```
COPY DATA ROM
```

The COPY command is included in this Linker Configuration File to facilitate the storage of data segments in ROM so that they can be copied to the RAM linker-designated memory at execution time. The linker also ensures that the code contains addresses that access the DATA section in RAM instead of the DATA_COPY section in ROM. The start-up code uses the linker symbols to copy the DATA segment from ROM to RAM.

All RAM

In the All RAM configuration, the RANGE command is used differently than in the Standard configuration, and it also contains the GROUP Linker command.

```
RANGE MEMORY $0 : $C7FFFF
GROUP MEMORY = ROM, RAM
```

The GROUP linker command is used to combine the Linker-designated ROM and RAM memories into one. The RANGE linker command is provided to specify the range of the memory group.

With ROM and RAM designated as MEMORY (memory group), the linker designated ROM is mapped starting with the first (lowest) address space of MEMORY, and the linker designated RAM is mapped to the first address space in MEMORY following the ROM mapping (see the Map File section on page 11 for the All RAM configuration). When this configuration is selected, it is important that the specification of the address space for both physical ROM and RAM memory in the Memory Control is actual RAM memory. This action can be performed with the proper selection of Chip Selects.

Just as in the Standard configuration, a COPY command is included to make a copy of the DATA segment and to locate it in ROM.

```
COPY DATA ROM
```

This inclusion would not be necessary if a separate start-up file that did not copy the DATA from ROM to RAM is used instead of the standard startup file.

Copy to RAM

The RANGE linker commands in the Copy to RAM configuration are the same as in the Standard configuration, which means that the Linker-designated ROM is ROM physical memory and the Linker-designated RAM is RAM physical memory.

```
RANGE ROM $0 : $0FFFFF
RANGE RAM $C00000 : $C7FFFF
```

In the Copy to RAM configuration, not only are DATA segments accessed from physical RAM, but also the CODE segments. The Linker command that makes this change is the CHANGE command, which is shown below.

```
CHANGE CODE is RAM
```

As in the Standard configuration, a copy of the DATA segments must be made available for copy from ROM to RAM. However, in the Copy to RAM configuration, a copy of the CODE segments must also be made available for copy from ROM to RAM. Therefore, the COPY command is used for both the DATA and CODE segments, as follows:

```
COPY DATA ROM
COPY CODE ROM
```

The Copy to RAM configuration also includes a *define* that is set to 1.

```
define __copy_code_to_ram = 1
```

This define tag is used by the standard start-up code to determine if it must copy the CODE segment from ROM to RAM. The copy is performed when `__copy_code_to_ram` is set to 1. Only in this configuration is this *define* set to 1.

Custom

It is assumed that the user will provide the Link Control file for the Custom configuration. However, a default Link Control file is provided as a starting point that the user can modify. The important linker commands found in this default control file follow the Standard configuration.

Map File

A map file is generated in a build if the Generate Map File checkbox is checked in the General Category inset within the Linker Tab of the Project Settings dialog box. It contains everything in the Link Control file and describes where code and data are located. ROM and RAM in the map file are the same as the Linker-designated ROM and RAM described in the Link Configuration Control section on page 6.

Note: The addresses generated in the map file are a result of the Link Configuration and the addresses specified in the Memory inset within the Target tab of the Project Settings dialog box.

Important results from the map files for each configuration are shown as follows. Please refer to the entries used in the Memory settings for the ROM and RAM physical memory addresses.

Standard

The map file for the Standard configuration shows that the Linker-designated memories ROM and RAM are mapped to the ROM and RAM physical memory spaces, respectively, as shown in the following map file:

```
Space            Base        Top         Size
-------------    --------    --------    --------
RAM              D:C00000    D:C00085    86h
ROM              C:000000    C:000985    986h
```

Looking at one of the modules from the map file shows how each of the segments are mapped into the Linker-designated memory address space. BSS and DATA are mapped into the RAM physical memory space and CODE is mapped into the ROM physical memory space, as shown in the following map file:

```
Name             Base        Top         Size
-------------    --------    --------    --------
Segment: BSS     D:C00016    D:C00079    64h
Segment: CODE    C:0001B5    C:000204    50h
Segment: DATA    D:C00000    D:C00015    16h
```

All RAM

The map file for the All RAM configuration shows that the Linker-designated memory ROM is first mapped to the group Memory, then followed by the mapping of the Linker designated RAM. In this case, the memory specified for ROM in the ZDSII memory control should be physical RAM memory space, as shown in the following map file:

```
Group: Memory    Base        Top         Size
-------------    --------    --------    --------
Space: ROM       000000      000985      986h
Space: RAM       000986      000A0B      86h
```

The mapping of the segments can be visualized by one of the modules in the map file, as shown in the following map file:

```
Name             Base        Top         Size
-------------    --------    --------    --------
Segment: BSS     D:00099C    D:0009FF    64h
Segment: CODE    C:0001B5    C:000204    50h
Segment: DATA    D:000986    D:00099B    16h
```

Note: In the above case, all segments are mapped to low memory.

Copy to RAM

The map file for the Copy to RAM configuration shows the same mapping of the Linker-designated ROM and RAM to physical memory as in the Standard configuration. See the map file below.

```
Space            Base        Top         Size
-------------    --------    --------    --------
RAM              D:C00000    D:C00844    845h
ROM              C:000000    C:000985    986h
```

However, the above map file also shows that a copy of the CODE and DATA segments are made available in Linker-designated ROM, as shown in the following map file.

```
ROM              Type                Base        Top         Size
-------------    ----------------    --------    --------    --------
.STARTUP         normal data         C:000000    C:00010F    110h
CODE             * segment copy *    C:0001C7    C:000985    7BFh
DATA             * segment copy *    C:0001B1    C:0001C6    16h
```

Finally, for a given module, both the DATA and the CODE segments accessed during execution are from physical RAM, as shown in the following map file.

```
Name             Base        Top         Size
-------------    --------    --------    --------
Segment: BSS     D:C00016    D:C00079    64h
Segment: CODE    D:C0008A    D:C000D9    50h
Segment: DATA    C:C00000    D:C00015    16h
```

Custom

Because the Link Control file for the Custom configuration is user-provided, the map file is generated based on the user's Linker commands. The default Link Control file created by this configuration is the same as the Link Control file for the Standard configuration. The user can then modify the Link Control file starting with this configuration. Therefore, the map file for the Custom configuration is also the same as the Standard map file, until modifications to the Link Control file are made by the user.

Executable Format Control

The Executable Format Control allows the user to specify the format of the linker output. Two main types of output files can be generated: load files that possess the `.lod` extension, or hex files that possess the `.hex` extension. The choice of output type is shown in the Link Control file by the filename extension for the load file. The load files are used by the ZDSII debugger to download the code via the ZPAKII Debug Interface Tool. The hex files can be used by other applications outside of ZDSII, such as the Flash Loader utility that programs the Flash memory via the serial port. Because the hex file contains an address in each data record in memory, the Flash Loader uses this information to determine where to place the code and data. With the newer versions of ZDSII, hex files can also be downloaded using ZPAKII.

Initialization Parameters Control

The Initialization Parameters Control is used only by the debugger and has no impact on the build. When the target is powered on, all of the eZ80® registers are set to their default values. This Control provides the debugger with the information necessary to initialize important registers, such as the Chip Select and PC Counter registers, to values according to the target configuration and the placement of code. If the load file contains a startup module, the settings in the Initialization Parameters Control should agree with the settings defined by the startup module. This relationship is also true when the ZDS II Integrated Flash Loader is used. This Flash Loader utility can be found in version 4.4.2 or later of ZDS II for eZ80® devices. When an external Flash Loader is used, the hex file must contain a startup module to initialize the registers. As a result, the settings in the Initialization Parameters Control are not used.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.zilog.com