zilog
Embedded in Life
An IXYS Company

Application Note

Modifying Linker Files to Provide Maintainability of Project Files

AN033401-0811

Abstract

This document provides an overview of the functionality of the Zilog Developer Studio II (ZDS II) development tool for the eZ80Acclaim! Linker/Locator, and how this functionality can be utilized to assemble all required files and objects into one project directory. It aims to provide a mechanism to effectively improve project maintainability through version control systems and to avoid issues associated with upgrading ZDS II and transferring a project to another development machine.

Note: Throughout this document, it is assumed that the user is knowledgeable about the different link configurations of the ZDS II development tool for eZ80Acclaim! products. This document is intended to lead the user toward separating the linker command file from the project file to maximize project maintainability when using a version control system.

Discussion

ZDS II offers a capability for developers to control how source code files will be linked together via the linker command file. This type of linkage is often useful when the application to be developed requires modifications to the standard start-up files included in the ZDS II installation and/or when the application requires special interrupt vector handling. In such cases, it is not a good idea to directly modify source files from the ZDS II installation directory. The following reasons explain why:

1. Such modifications will definitely affect other applications that will be developed on the same machine.
2. Compile/build issues may arise when upgrading to the latest ZDS II version.
3. Because not all necessary files are contained in a single directory, transferring project development files onto another machine may introduce compile/build issues, and proper source code version control may not be implemented accordingly.
4. Compile/build issues will be introduced to other projects not related to the modifications in the start-up files; therefore, the developer should make the development machine specific only to a certain project.

Overview of the Zilog IEEE 695 Linker/Locator

ZDS II for eZ80Acclaim! includes a Linker/Locator that is responsible for creating a single executable file from a set of object files. Object files may either be in the form of a module or a library. The Linker/Locator links together objects and resolves symbols – as well as external references to public symbols – across each object. It is also responsible for assigning physical memory addresses into which each segment will be loaded. Furthermore, the Linker/Locator provides the user the ability to control how source files will be linked together through the use of the linker command file.

Note: A separate linker command file may be incompatible with future releases of ZDS II for eZ80Acclaim!. In particular, a future release of the compiler may require corresponding changes in the library files supplied with the release. If upgrading to a new release of ZDS II for eZ80Acclaim!, it is best to temporarily restore the **Generate from Settings** option, generate a new linker command file (with a different name), then manually copy the files in the ZDS II distribution paths to the linker command file to be used for the build.

The output of the Linker/Locator is an executable file that can exist in IEEE-695 (`*.LOD`) format or in Intel Hex32 Records (`*.HEX`) format. The IEEE-695 format is a debugging format that can be used mostly when debugging a project within ZDS II; conversely, the Intel Hex32 Records format is a more generic executable file that presents an executable file in ASCII. The Intel Hex32 Records format is used primarily when the debugging phase is complete and the application is to be run without the ZDS II debugger.

Why Use a Custom Linker File?

The linker command file (or linker file) is automatically created by ZDS II depending on the context presented in the **Project Settings** dialog box. In most cases, it is certain that the **Project Settings** dialog box is adequate for offering users the flexibility and ease of configuring their project according to their requirements. These settings are saved and embedded into the ZDS II project file. After building the project, a copy of the linker file produced as a result of these project settings is also saved into the defined output directory.

Some developers opt to save the linker file separately from the project file; here are two general reasons why:

1. When using version control, it is easier to monitor changes in the link configuration than monitor change from the project file; the workspace and project files are always updated as soon as the project is opened. If these files are placed into version control, it is a common scenario that version control will always flag these files as changed or modified. As a result, the developer is left with the impression that these flags exist only because the project was opened. Any accidental or unwanted changes in the project settings may be overlooked when these files are committed into source control. Contrary to custom linker files, the command file will never change unless you modify it on purpose.
2. The separate linker file offers total control as to how your project will be linked together; some projects require only one or two of the three standard start-up files to be modified. In such a case, the developer wishes to ensure that his unmodified start-up files come straight from the ZDS II installation directory. However, one limitation of the ZDS II **Project Settings** dialog box is that it only allows the user to choose either to load the start-up modules from the ZDS II installation (Standard) or load the

start-up modules from the project itself (Included in Project). In such a case, defining the linker file is beneficial toward properly locating the start-up modules.

Default Linker Command File

This section describes the default contents of the linker command file, and aims to itemize the contents of the linker file for discussion purposes. For details about the different linker commands, please refer to the Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144).

Linker Output Format & Linker Warning Settings

This portion of the linker file describes the desired output file format of the linker. There are 2 file formats that the ZDS II Linker/Locator can produce: the IEEE-695 format and the Intel Hex32 Records format. ZDS II supports producing both files simultaneously. This portion of the linker file is also where settings for warning generation are declared.

The default output format and warning settings of the linker are declared in the linker command file, as shown in the following code segment:

```
-FORMAT=OMF695,INTEL32
-map -maxhexlen=64 -quiet -warnoverlap -xref -unresolved=fatal
-sort ADDRESS=up -warn -debug -NOigcase
```

Address Spaces and Linking Sequence

This portion of the linker command file is responsible for setting up linker parameters related to the memory map. It is also responsible for establishing the link sequence and setting up a dynamic range for contiguously-mapped address spaces.

The default address space configuration of the linker is declared in the linker command file as shown in the following code segment. The define statements create user-defined public symbols at link time that are used to resolve external references at assembly time.

```
RANGE ROM $000000 : $03FFFF
RANGE RAM $B7E000 : $B7FFFF
RANGE EXTIO $000000 : $00FFFF
RANGE INTIO $000000 : $0000FF

CHANGE STRSECT is ROM

ORDER .RESET,.IVECTS,.STARTUP,CODE,DATA
COPY DATA ROM

DEFINE __low_romdata = copy base of DATA
DEFINE __low_data = base of DATA
DEFINE __len_data = length of DATA
DEFINE __low_bss = base of BSS
DEFINE __len_bss = length of BSS
DEFINE __stack = highaddr of RAM + 1
DEFINE __heaptop = highaddr of RAM
DEFINE __heapbot = top of RAM + 1
DEFINE __low_romcode = copy base of CODE
DEFINE __low_code = base of CODE
DEFINE __len_code = length of CODE
DEFINE __copy_code_to_ram = 0
DEFINE __crtl = 1
DEFINE __CS0_LBR_INIT_PARAM = $10
DEFINE __CS0_UBR_INIT_PARAM = $1f
DEFINE __CS0_CTL_INIT_PARAM = $a8
DEFINE __CS0_BMC_INIT_PARAM = $02
DEFINE __CS1_LBR_INIT_PARAM = $c0
DEFINE __CS1_UBR_INIT_PARAM = $c7
DEFINE __CS1_CTL_INIT_PARAM = $28
DEFINE __CS1_BMC_INIT_PARAM = $02
DEFINE __CS2_LBR_INIT_PARAM = $80
DEFINE __CS2_UBR_INIT_PARAM = $bf
DEFINE __CS2_CTL_INIT_PARAM = $28
DEFINE __CS2_BMC_INIT_PARAM = $81
DEFINE __CS3_LBR_INIT_PARAM = $00
DEFINE __CS3_UBR_INIT_PARAM = $00
DEFINE __CS3_CTL_INIT_PARAM = $00
DEFINE __CS3_BMC_INIT_PARAM = $02
DEFINE __RAM_CTL_INIT_PARAM = $C0
DEFINE __RAM_ADDR_U_INIT_PARAM = $B7
DEFINE __FLASH_CTL_INIT_PARAM = $68
DEFINE __FLASH_ADDR_U_INIT_PARAM = $00

define _SYS_CLK_FREQ = 20000000
define _OSC_FREQ = 20000000
define _SYS_CLK_SRC = 0
define _OSC_FREQ_MULT = 1
define __PLL_CTL0_INIT_PARAM = $00
define _zsl_g_clock_xdefine = 50000000
```

Objects and Libraries List

This portion of the linker command file contains a list of object files required to create the executable file. These object files are created by the compiler and are saved in the directory defined in **Project Settings → General → Immediate Output Directory**.

The default objects and libraries list in a simple application is declared in the linker command file, as shown in the following code segment. Note that the default start-up files are taken from the ZDSII installation directory.

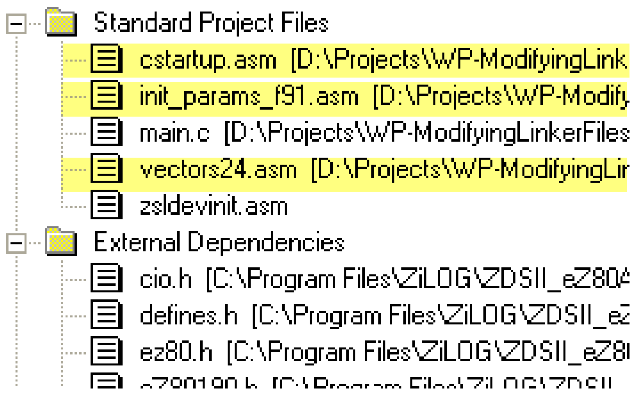
```
"D:\Projects\WP-
ModifyingLinkerFiles\Source\Debug\ProjectMaintainability"= \
C:\PROGRA~1\ZiLOG\ZDSII_~1.1\lib\zilog\vectors24.obj, \
C:\PROGRA~1\ZiLOG\ZDSII_~1.1\lib\zilog\init_params_f91.obj, \
C:\PROGRA~1\ZiLOG\ZDSII_~1.1\lib\zilog \cstartup.obj, \
.\main.obj
```

Note: When editing this portion of the linker command file, consider the available configurations and choose which is appropriate for your project. For more details, please refer to the Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144).

Configuring ZDS II to use a Custom Start-Up Module and Custom Linker File

This section explains the steps required to successfully compile and build a project that uses modified C start-up modules.

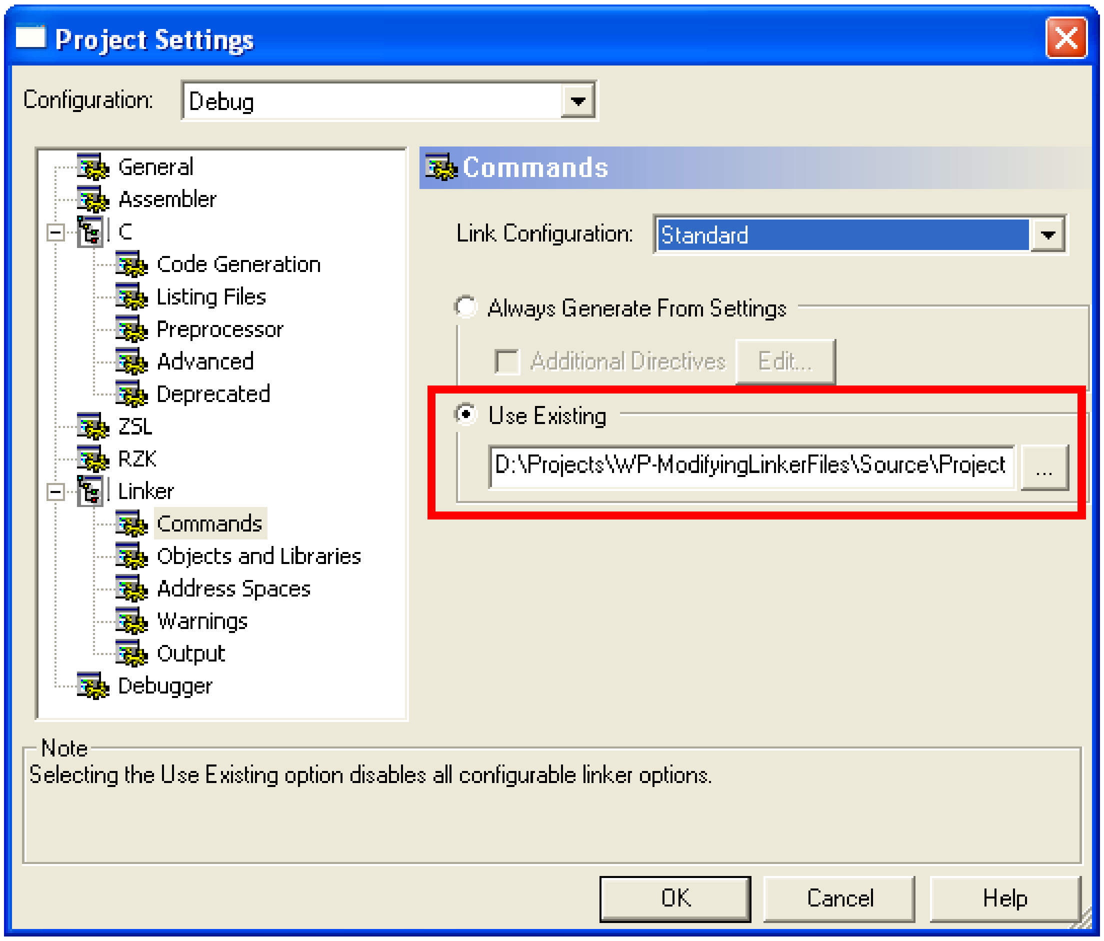
1. Check to determine if all of the project settings are configured according to the requirements of the project to ensure that you only make minor modifications to the custom linker file later.
2. From the **Project** menu, select **Export Makefile**. A **Save As...** dialog box will appear. Enter a filename for your linker file, and click **Save**. Two files will be saved: a make file and a linker command file; only the linker command file will be required.
3. The following ZDS II start-up files are required; locate them in Windows Explorer:
   - src\boot\common\vectors24.asm (for eZ80F91)
   - src\boot\common\vectors16.asm (for eZ80190, eZ80L92, eZ80F92, eZ80F93)
   - src\<device>\init_params_<device>.asm
   - src\boot\common\cstartup.asm
4. Copy the start-up files that require modification (highlighted in Figure 1) to the target project directory, and add these files to the project.



```
Standard Project Files
    cstartup.asm  [D:\Projects\WP-ModifyingLink
    init_params_f91.asm  [D:\Projects\WP-Modify
    main.c  [D:\Projects\WP-ModifyingLinkerFiles
    vectors24.asm  [D:\Projects\WP-ModifyingLir
    zsldevinit.asm
External Dependencies
    cio.h  [C:\Program Files\ZiLOG\ZDSII_eZ80A
    defines.h  [C:\Program Files\ZiLOG\ZDSII_eZ
    ez80.h  [C:\Program Files\ZiLOG\ZDSII_eZ8
```

Figure 1. The Standard Project Start-Up Files

5. Navigate via **Project Settings** → **Linker** → **Commands**, and select the **Use Existing** radio button, as indicated in Figure 2. Click the "..." button to the right of the **Use Existing** entry field to browse to the location of the linker command file that you created in Step 2.



Configuration: Debug

Commands

Link Configuration: Standard

Always Generate From Settings

Use Existing

D:\Projects\WP-ModifyingLinkerFiles\Source\Project

Note
Selecting the Use Existing option disables all configurable linker options.

Figure 2. The Linker Command Interface

Note: When the option in Step 5 is selected, linker settings can no longer be edited from within the **Project Settings** dialog box. Future modifications should be performed directly on the linker command file.

6. Click **OK**. When asked to rebuild the project, choose **No**.
7. Open the linker command file created in Step 2 in a text editor. In the **Objects and Libraries** portion of the file (located at the bottom part of the file), change the directory path of the start-up files that you imported into the project.

8. Save the modified linker command file and rebuild your project.

How to Establish a Short Filename and Path

Observe the following brief procedure to ensure short filenaming in the DOS 8.3 format.

1. From the Windows **Start** menu, enter **Run** to open the **Run** dialog, then enter **cmd.exe** in the **Run** dialog's **Open:** field to launch a DOS prompt.
2. At the prompt, change your current drive to the drive in which ZDS II is installed. If ZDS II is installed in the default directory path, this path will exist in drive C:\.
3. Change your current directory to where ZDS II is installed, as shown in the following example:

   ```
   cd "Program Files\Zilog\ZDSII_eZ80Acclaim!_5.1.1"
   ```

4. Enter `command.com` to change the current command shell to the older DOS 8.3 shell, which displays its results in the short filename format.

Note: The `dir \x` command shows both the long and short file names for the current directory.

Caution: For files that are not copied into the target project directory, make sure to retain the *short file names* format generated by ZDS II, because the space required for a long file name format will result in problems in the linker. When upgrading ZDS II, make sure to update these target paths to reference the latest version.

References

The following documents describe the functional specifications and toolsets for the eZ80Acclaim! MCU. Each is available for download from the Zilog website.

eZ80F91 MCU Product Specification (PS0192)

eZ80 CPU User Manual (UM0077)

Zilog Developer Studio II – eZ80Acclaim! User Manual (UM0144)

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.