White Paper

Z8 Encore!® Using Zilog Standard Library (ZSL)

WP001002-0208

Abstract

This White Paper explains how to use the new Zilog Standard Library (ZSL) software component in Zilog Developer Studio II (ZDS II)—Z8 Encore!® projects. Different models are described for using, or not using ZSL.

Introduction

The Zilog Developer Studio II (ZDS II)—Z8 Encore! v4.9.0 distribution includes a new software component called the Zilog Standard Library (ZSL). ZSL is a set of library files that provides an interface between the application and on-chip peripherals of the Z8 Encore! microcontrollers. The ZSL included in this ZDS II distribution supports GPIO and UART on-chip peripherals.

To integrate the new ZSL software component, changes are done to the C-language runtime library (RTL) `sio.c` and `sio.h` files. These files originally held UART-related implementations that now reside in the ZSL. To make migration to ZDS II v4.9.0 easier, the `sio.c` and `sio.h` files still exist in the RTL as a wrapper for corresponding ZSL functions. That is, functions called from the `sio.c` file now call ZSL functions in turn to access the UART.

Supported Use Models

ZSL is created with the following use models:

- Migrating an existing application to ZDS II v4.9.0, adopting ZSL. This model has two different scenarios:
  - Using the RTL I/O functions.
  - Using ZSL APIs directly.
- Using ZSL in a completely new project.
- Running ZSL code on the simulator.
- Choosing not to adopt ZSL.
- Using ZSL with custom startup or link control files.

These models are described in the following sections.

Migrating an Existing Application to ZDS II v4.9.0

When you use ZDS II v4.9.0 to open a project created for an earlier version, the project is converted to a new format with the file extension `.zdsproj`. As part of this conversion, ZDS II also performs the following functions:

- Copies a new file, `zsldevinit.asm`, into the project. This is a ZSL-specific device initialization file. It contains the routine `_open_periphdevice()`, which initializes device-related parameters needed for ZSL to work properly. The standard startup module calls this initialization routine before calling the `main()` function.

- If the project uses any RTL or ZSL I/O calls, ZDS II automatically links the ZSL libraries relevant to the memory model (small or large) and stack frames (static or dynamic) used in the project.
- If the application does not use any RTL or ZSL I/O calls, ZSL still adds the following overhead to the application:
  - A call to `_open_periphdevice()` from the startup routine. In this case the `_open_periphdevice()` is an empty function that simply returns.
  - Five bytes of global variables (four bytes `of g_clock0/1` and one byte of `g_simulate`), declared in the `zsldevinit.asm` file.

When the project conversion completes, the project can be rebuilt and downloaded as usual.

Using RTL I/O Functions

As mentioned in the Introduction, the RTL I/O calls defined in `sio.c` file act as a wrapper for underlying ZSL APIs in ZDS II v4.9.0. This wrapper function has the following limitations:

- Only one UART can be used at any given time. You cannot dynamically switch between UARTs with the RTL `init_uart()` or `select_port()` functions, because they can only act on the default UART. These functions return an error if they do not reference the default UART.

  The default UART is set in ZSL's `uartcontrol.h` file by the following macro definition:

  ```
  #define DEFAULT_UARTUARTx /* UARTx is the default device */
  ```

  where *x* is either 0 or 1 for UART0 or UART1, respectively.

  After the default UART is changed, the ZSL and RTL libraries must be rebuilt before the UART is used in the application. For more information on building the ZSL libraries, refer to *Zilog Standard Library API Reference Manual (RM0038)*.
- Calling ZSL I/O functions via the RTL results in a slightly larger application size. This is worth considering when the application is written for a small memory model.

Using ZSL APIs Directly

If higher-level RTL I/O functions are not required, then the `sio.h` include file can be omitted and the ZSL API calls can be used directly.

Using ZSL in a New Project

ZSL provides a number of APIs to use and configure GPIO and UART peripheral devices. ZDSII v4.9.0 has a new GUI that allows you to specify some of the device-dependent parameters.

A new tab named **ZSL** has been added to the **Project** → **Settings** dialog box. The new tab allows you to select or deselect GPIO ports and UARTs to be used in the project. Based on the selection, ZDSII defines various C and assembly macros to initialize each selected device.

For more information on the ZDSII GUI, refer to *Zilog Developer Studio II—Z8 Encore!® User Manual (UM0130)*. For detailed steps to create a new project with ZSL and list of APIs, refer to *Zilog Standard Library API Reference Manual (RM0038)* available for download at www.zilog.com.

Running ZSL Code in the Simulator

When ZSL functions are executed in the ZDSII simulator, characters printed via the UART are routed back for display in the **Debug** tab of the **Output** window. No special action is required to use this feature, but the following considerations apply:

- Rebuild the project whenever the debugger driver is changed, (that is, Simulator, USB, or Serial Driver).
- The Simulator driver in ZDSII v4.9.0 cannot simulate read-related I/O calls such as `scanf()`, `getch()` and `getchar()`, so these do not work when this driver is selected. This restriction does not apply when running ZSL code on a development board or target hardware.

Choosing Not to Adopt ZSL

If you want to bypass ZSL and use your own peripheral driver for UART or GPIO access, the best way is to simply include your driver source files inside the project. This ensures that the application is always linked with the source file's implementation instead of ZSL's.

If your peripheral driver is in the form of a library, ensure that it comes before ZSL in the linking order. This is not an issue if ZDSII generates the link control file; in this case you can either add the library directly to the project like any source file, or include the library name in the linker settings.

Using ZSL with Custom Startup or Link Control Files

If you do not use the startup module provided with ZDSII, your application should explicitly call the function `_open_periphdevice()` from `zsldevinit.asm` before calling any ZSL functions.

Follow the steps below if you are using your own link control file:

- Define a linker directive `_zsl_g_clock_xdefine` and equate it with the value of the target clock frequency. For example, if the target clock frequency is 18432000, the directive should be:

  ```
  define _zsl_g_clock_xdefine = 18432000
  ```

  This definition is used inside the `zsldevinit.asm` file to configure the UART baud rate generators.

- Include ZSL libraries relevant to the desired target memory model (small or large) and stack frame model (static or dynamic). For a list of available libraries and their description, refer to *Zilog Standard Library API Reference Manual (RM0038)* included with the ZDS II tool package available for download at www.zilog.com.

Warning: DO NOT USE IN LIFE SUPPORT