Mini-Z Shell and Flash Loader

Reference Manual

RM006101-0311

zilog
Embedded in Life
An IXYS Company

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

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

Revision History

Each instance in this document's revision history reflects a change from its previous version. For more details, refer to the corresponding pages linked in the table below.

Date	Revision Level	Description	Page No
Mar 2011	01	Original issue.	All

Table of Contents

The Mini-Z Shell: An Overview

Zilog's Mini-Z modules are shipped with a preloaded application which allows interaction with the module via a console program such as HyperTerminal. The application allows you to power up and start using the module without any programming. You can turn each of the ports on and off, check the analog-to-digital (ADC) converters, activate the pulse width modulation (PWM) systems, and even check the logic values of each of the ports. Additionally, you can flash and execute an application using the console program. The application also provides a `debugoutput()` function which allows you to print the information produced by your application without having to set up or configure the UART.

The serial connection is configured for 57,600 baud, 8 bits, no parity, 1 stop bit and no flow control. After your module is plugged into a design board, you can connect the serial cable or USB cable (depending on the serial connection the design board provides) to the PC with the console program.

Note: In this manual, we discuss the output of the Z16MiniZ28 Module; however, the HyperTerminal output you see may be slightly different depending on the module you are using.

With the console connected, power up the module. You should see the following prompt on your monitor:

```
Z16MiniZ>
```

Using your keyboard, enter "?" and press the Enter key to list all of the available commands. Entering a command followed by a space, then a "?", returns a help text for that command. As a developer, you can add commands and functionality to the Shell by providing a handler for the command request. All commands are case-insensitive and the parameters

are separated by one or more spaces. See the Programming Considerations section on page 11 for more information.

Shell Commands and Descriptions

This section describes the nine Mini-Z Shell commands.

? Help

Description As a command, ? provides a list of all available built-in commands. As a parameter, it requests help information for the command.

Example 1

```
Z16MiniZ>?
```

Return

```
Commands Available:
  ?
  SetPort
  SetLed
  GetPort
  GetADC
  ExecApp
  FlashApp
  ConfPWM
  PWMDuty
```

Example 2

```
Z16MiniZ>getadc ?
```

In Example 2, the `getadc` help command retrieves the ADC (0–3) value. The reported voltage measurement ranges from 0–2 V.

Syntax: GETADC 0|1|2|3

SetPort Set a specific port to a value

Description The `SetPort` command sets the requested port to either ON (3.3 volts) or OFF (0 volts). The command contains two parameters; the first is the port number, referenced as P0 through P19. The second is the state to set the port. The specific pin is set to output, and is set to either High (ON) or Low (OFF).

Example

```
Z16MiniZ>SetPort P8 on
```

In the example above, Port P8 is placed into output mode and the pin is set to output 3.3 volts.

SetLed Turn on specific LEDs on the Mini-Z Module

The Mini-Z modules feature 3 LEDs, colored red, green and yellow. You can turn them on with the `SetLed` command. There are two parameters; the first governs the LED by color. Only the first letter is required; however, the Shell will accept the entire word. The second parameter governs the state to which the LED is set – either ON or OFF.

Example

```
Z16MiniZ>SetLed g on
```

In the example above, the green LED on the Z16MiniZ28 Module is turned ON.

GetPort — Get the port value

This command sets a specific port to input mode and reads the value. There is only one parameter, which is the port to read. The port is referenced as P0 through P19.

Example 1

```
Z16MiniZ>getport P18
```

Return

```
Port P18: off
```

The example above shows if the port value is high or low. Do not be confused between when you *set* the port versus when you *get* the port value. When you set the port, you are placing it into output mode; therefore, the connection is internal. When you get the port value, you are placing the port into input mode; therefore an external factor must be controlling the voltage level of the pin.

Example 2

If you execute the following command:

```
Setport P18 on
```

Port P18 will turn ON in output mode. Then, when you issue the following command:

```
Getport P18
```

The returned value will be:

```
Port P18: off
```

This returned value shows that Port 18 is off because it is in input mode and nothing is connected to it.

GetADC Read the associated ADC

The `GetADC` command sets the specific port to the alternative function for the ADC, then issues a conversion request and displays the voltage measured. This command requires the ADC number as the parameter (0–3). The voltage reading is between 0–2 volts.

Example

```
Z16MiniZ>getadc 1
```

Return

```
ADC 1 Reading: 0.311 volts
```

The reading above is 0.311 volts. Because some of the pins are typically floating, it is very possible to encounter stray noises that show as minor voltage levels on the ADC lines.

ConfPWM Configure the PWM

This command configures the PWM cycle length, which is the total duration of each cycle; the default length is 1000 cycles per second. The parameter is the PWM cycle length, specified in 1000 cycles/second (KHz). If there are no parameters, then the current PWM cycle length is displayed.

Example 1

```
Z16MiniZ>confPWM 20
```

Return

```
PWM is configured for 20KHz cycles. Set duty cycle to
start.
```

Example 2

```
Z16MiniZ>confPWM
```

Return

```
PWM is configured for 1KHz cycles. Set duty cycle to
start.
```

PWMDuty Configure the PWM Duty Cycle

The `PWMDuty` command sets the duty cycle for the PWM output. The duty cycle is defined as the duration of the PWM output, i.e., how long it remains ON. For example, if the duty cycle is set to 20%, the output signal will be ON for 20% of the PWM cycle, then OFF for 80% of the cycle; this process then repeats.

The `PWMDuty` command requires two parameters; the first causes the PWM to start. This PWM must already have been configured by the `ConfPWM` command without issuing any other commands on the port. The second parameter is the duty cycle duration from 0–100; 0 represents *always off*, and 100 represents *always on*. Any reconfiguration of the port will disable the PWM configuration and duty cycles.

Example

```
Z16MiniZ>PWMDuty 1 75
```

In the example above, PWM 1 is switched on for a duration of 75% of the cycle.

FlashApp Flashes an application from a hex file

The `FlashApp` command prepares and *bootloads* an application onto the Mini-Z Module. The command requires no parameters.

When you execute this command, it erases the contents of the Flash memory space in which your code will be located. When this erasure is complete, you will be prompted to load a hex file or press the `Esc` key to exit. To load Flash memory (assuming you are using HyperTerminal), select **Transfer** from the HyperTerminal window's menu, then **Send Text File…** to display a **File** dialog box. In the **Files of Type** drop-down menu, change the **File Type** selection to **All Files (*.*)**, then locate and select the hex file for the application you wish to flash.

The program then sends the data to the Module. As the Module processes each line, it displays a period (`.`). When this process completes, you will be informed as to whether there were any errors or if the flash was successful.

Example

```
Z16MiniZ>flashapp
```

Return

```
Mini-Z Bootloader
Erasing Flash, One moment please
.....
Flash erased
Please load HEX file to Flash, press 'Esc' key to
cancel
.....
Flash successfully updated. Use ExecApp command to
start execution.
```

ExecApp Execute the application

This command executes the application that has been flashed. It does not require any parameters. If there is no valid application, an error is returned.

Example

```
Z16MiniZ>execapp
```

Return

```
Application in control
```

If the application returns, the following message is displayed, followed by the `Z16MiniZ>` prompt:

```
App has exited
```

Programming Considerations

As a developer, you have full control over the MCU. If you utilize the USB Smart Cable and mini-debug adapter (see your Mini-Z Module product specification for more information) with the appropriate ZDS tools set, you can use all of the MCU's memory upon removing the Shell and Flash loader program. If you wish to use the Shell program, you can build your application using the ZDS tools, then flash the hex file. You can add functionality for the Shell by building the application with your own additional features, flashing the application, then running the application to register the entry point. When the application exits, the Shell will have the additional functionality.

To control the MCU on the Mini-Z Module without saving the Shell, use the ZDS tools suite as configured for the appropriate MCU. Example, if you were using a Z16MiniZ28 Module, you would use the ZDSII tool for ZNEO, and select the Z16F2810AG CPU with the default project settings. You would then code your application as you would for any MCU.

To keep the Shell but write your own application, you only need to include the Mini-Z-specific `startups.asm` file in your project (located in the `inc` directory) and include a library file in your project settings. You have the choice of letting the main function return, which will bring the Shell active again. Everything else operates normally, except that you have a built-in command:

```
void debugoutput(char *str)
```

This command allows you to output debug messages to the console while running your app without having to set up or configure the UARTs. The `SimpleSample` project is an application that reports a brief message and returns control back to the Shell.

The `AddMsg` project sample shows a skeleton in which you can add your own functionality to the Shell. Essentially, you would create your ZDS

project as defined by the Mini-Z Shell project instructions; then you would add the `ProcessCommand()` function to register your callback function address just before exiting the `main()` function. When `main()` exits, the Shell will resume control. When the Shell receives a command from the console, the Shell will parse the command and call the callback function that was registered.

The callback function prototype is:

```
BOOL callback(void);
```

The prototype to register the function is:

```
int ProcessCommand(PC_CMD *fun);
```

Your callback function does not get any parameters. The Shell exposes the `Tokens` variable, which is an array of character pointers that points to each of the parts of the command, as such: `Tokens[0]` points to the command, `Tokens[1]` is the first parameter, etc.

There is a maximum of three tokens. Your callback function should return a Boolean value to indicate if you have completed all of the processing for the command. If your callback function returns TRUE, then the shell will not perform any further processing of the command. If the callback function returns FALSE, then the shell will process the command normally.

Download the Mini-Z Shell Project

The following files are contained in the Mini-ZShellLibrary.zip file, which is available for download from the *Zilog Concept Products* section of the IXYS Colorado store at http://ixyscolorado.com/store.

- Lib
- Inc
- Demo
  - Simplesample

– AddMsg

Mini-Z Shell Project Instructions

When working with the Z16MiniZ28 Module, you can copy either the `AddMsg` or `SimpleSample` project and build on each for quick starting.

To start a new project:

1. Using ZDSII for ZNEO version 4.12, select **File → New Project**.
2. Key in your project name, then select **CPU Z16F2810AG** in the **CPU** drop-down menu.
3. Select **Continue**.
4. Uncheck the **Standard C Startup Module**. You may either leave or uncheck the other two boxes – **C Runtime Library** or **Floating Point Library**.
5. Select **Finish**.
6. From the ZDSII menu, select **Project → Settings**.
7. In the **Linker** tab, select **Objects and Libraries**.
8. In the **Additional Object/Library Modules**, add the `MiniZLib.lib` file, with the correct path to the file.
9. Select **OK**.
10. Make sure that you include the `startups.asm` file in your Standard Project Files.

When you build your application, the ZDS tool will output a hex file to a folder of the project type (typically the Debug folder). This hex file is the file to flash the Module with.

Customer Support

To learn more about this product, find additional documentation, get your technical questions answered or report issues, please contact esales@zilog.com.