Application Note 3108
Modifying and Rebuilding Slush

www.maxim-ic.com

INTRODUCTION

A major concern of many embedded device programmers is the robustness of their application. Will the device always be able to start and perform its intended function? One of the easiest ways to gain this reliability on the TINI® microcontroller is to place the main application in flash memory. However, by placing code here, the default application, *slush*, gets overwritten. Most developers would like to keep the functionality of *slush* and gain robustness for their application too. This can be accomplished by making modifications to *slush,* adding the desired functionality. This application note describes the steps required to modify *slush*, including where to change the code. It also discusses how to rebuild the application and ways to reduce *slush's* size.

CHANGING THE SOURCE CODE

Before you can change the *slush* source code, you must extract it from the distribution package. All the source code for *slush* can be found in the `SlushSrc.jar` file located in the `src` directory of each TINI release. Once you have done this, you can begin making your changes. Two of the most common modifications are adding custom commands to the shell and combining the functionality of another program with the features of *slush*. The steps required for each task are outlined below.

ADDING A CUSTOM COMMAND

The first step is writing the code for the command. Every command in *slush* must implement the `com.dalsemi.slush.command.SlushCommand` interface. This interface has two methods:

```
String getUsageString();

void execute(SystemInputStream in, SystemPrintStream out, SystemPrintStream
err, String[] args, Hashtable env) throws Exception;
```

The `getUsageString` method should return the message you want users to see when using *slush's* `help` command. The real work is done in the `execute` method. This is where you will add the code to be executed when the command is run. The `in`, `out`, and `err` parameters are the console's standard input stream, output stream, and error stream, respectively. The `args` array will contain all the arguments the user entered when they executed the command. The `env` parameter will contain various system properties and environmental variables.

Once you have finished coding the command, you need to add it to *slush's* list of known commands. Find the `initializeShellCommands` method in the `Slush.java` file located in the `com.dalsemi.slush` package. Inside this method, create an instance of your command and pass it to the `addCommand` method of the `com.dalsemi.slush.CommandInterpreter` class. This method has the following prototype:

```
public static final boolean addCommand(String name, SlushCommand executer)
```

The first parameter is the string you want the user to type at the *slush* command line to run the command. The second parameter is the instance of the command itself.

ADDING NEW FUNCTIONALITY

If you want your code to run every time *slush* runs, and not just when a certain command is entered, you need to make changes to the `com.dalsemi.slush.Slush` class. The best place to make these changes is in the `main` method. In the default implementation of *slush*, there is only one line of source code, a call to the `Slush` constructor. In the constructor, the network servers are initialized, all the commands are added, and the startup file is executed. Add your code before or after the call to the `Slush` constructor, based on your program's needs.

BUILDING THE BINARY

Now that you have made your changes, you need to compile the new source into a file that can be loaded into TINI's flash memory. The first step is to compile your source into class files. Use whatever tool you prefer for this step. If you use *javac* on a windows system, the following two command lines may be used:

```
dir %SlushSource%\*.java /B /S /ON > files

javac -bootclasspath %TINIBin%\tiniclasses.jar;%TINIBin%\modules.jar –d
   %SlushBin% @files
```

where `%SlushSource%` is the directory holding all the *slush* source code and your modifications, `%TINIBin%` is the bin directory of the TINI SDK, and `%SlushBin%` is the directory where you want the resulting class files stored. Once you resolve any compiling errors, use the BuildDependency tool to convert the class files into TINI's flash file format. The following command is used to build *slush* for TINI releases (Note: this command should be entered on a single line):

```
java -classpath %TINIBin%\tini.jar BuildDependency -f %SlushBin% -d %TINIBin%\tini.db
   -o slush.tbin -l -p %TINIBin%\modules.jar -add FTPCLIENT;MAILTO -fake –noreflect-
   ref com.dalsemi.slush.util.FakeMainMaker -ref
   com.dalsemi.slush.command.SlushCommand -ref com.dalsemi.protocol.mailto.Handler
```

where `%TINIBin%` and `%SlushBin%` are the same as in the previous step. This command should be used when building *slush* for the TINIm390 module. If you need to build *slush* for the TINIm400, add a "`-t 470100`" switch to the command line above. Executing this command will result in a `slush.tbin` file that can be loaded into TINI's flash.

REDUCING THE SIZE

Many times after making changes to *slush*, you will discover the resulting `slush.tbin` file is too large to fit into one bank of the flash on the TINIm390. Unless you've added extra flash to your TINI board, you only have 65,280 bytes available, and you must reduce the size of your application. The easiest solution is to remove some of the *slush* commands you do not need. First, look through the list of commands in the `com.dalsemi.slush.command` package and decide which ones you would like to remove. Second, edit the `initializeShellCommands` method in the `Slush.java` file and remove the calls to `CommandInterpreter.addCommand` where the commands you selected are configured. Next, follow the steps in the previous section to rebuild *slush*, making sure to remove the class files of the commands you removed before running the last step. Continue removing commands until *slush* reaches the appropriate size.

CONCLUSION

By making additions and modifications to *slush*, you can merge the functionality of your project with the many features already available in *slush*. Since your application will now be loaded into TINI's flash memory instead of residing in RAM, you will also gain robustness for your application.

MAXIM INTEGRATED PRODUCTS/DALLAS SEMICONDUCTOR CONTACT INFORMATION

Company Addresses:

Maxim Integrated Products, Inc.
120 San Gabriel Drive
Sunnyvale, CA 94086
Tel: 408-737-7600
Fax: 408-737-7194

Dallas Semiconductor
4401 S. Beltwood Parkway
Dallas, TX 75244
Tel: 972-371-4448
Fax: 972-371-4799

Product Literature/Samples Requests:
800-998-8800
408-737-7600

Sales and Customer Service:

World Wide Website:
www.maxim-ic.com

Product Information:
www.maxim-ic.com/MaximProducts/products.htm

Ordering Information:
www.maxim-ic.com/BuyMaxim/Sales.htm

FTP Site:
ftp://ftp.dalsemi.com