Chameleon

series

Flash Programming Plugin for Chameleon Debugger

User Guide

SIGNUM SYSTEMS

SIGNUM SYSTEMS

11992 CHALLENGER COURT
MOORPARK, CA 93021, U.S.A
PHONE 805 • 523 • 9774
WWW.SIGNUM.COM

Purpose   *This document explains how to use the flash programming utility built into the Signum Systems Chameleon Debugger software.*

Installation

Chameleon Debugger is furnished with a flash programming plug-in. This plug-in is installed by default, but can be uninstalled or reinstalled at any time if necessary. For details, please refer to *Chameleon User Manual.*

Flash Programming Processes

The process of programming flash using the plug-in programmer consists of two phases:

- configuring (setting the parameters of) the plug-in and

- executing proper plug-in commands.

▶   To program the flash memory on your target system

Open the Flash Programmer plug-in dialog by selecting the Flash Programmer option from the Tools menu. The Flash Programming dialog box appears (Figure 1).

FIGURE 1   Running the Flash Programmer.

Configuring the Plug-In

Befre it can program flash memory, the programmer needs to be configured. The first and critical step in setting the necessary programmer parameters is selecting the appropriate flash device.

▶   To choose the flash device

Choose the flash device programmer and set its parameters in the Flash Device tab (Figure 2).
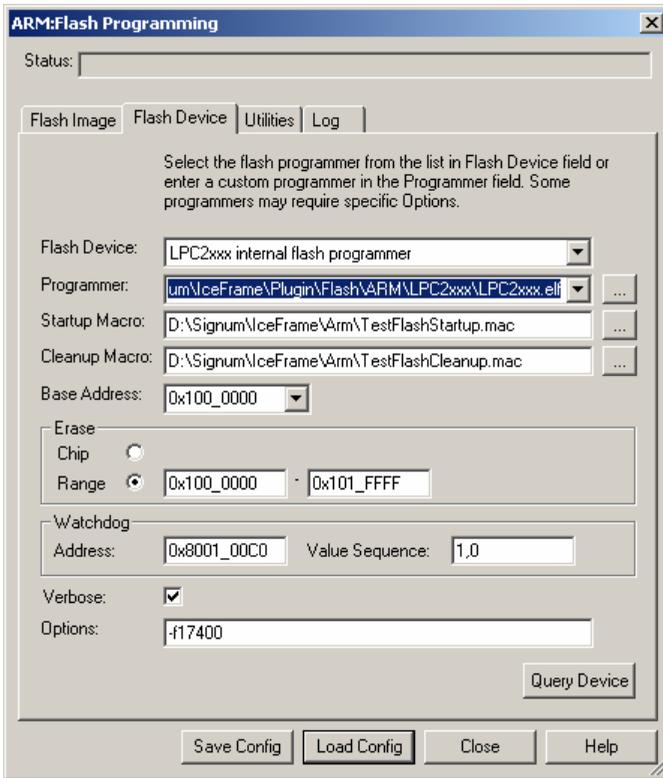
FIGURE 2 Setting up the flash device and its programmer.

- **Select the flash device** from the Flash Device drop-down list. A Programmer file associated with the current flash device appears in the Programmer drop-down list. Use the Custom option (at the bottom of the Flash Device list) to add new devices.

- **If you are using a Custom device, select the Programmer file** from the Programmer drop-down list or navigate to the file using the browse button on the right.

- **Select the Startup Macro** file. The startup macro prepares the processor for flash programming. Usually, this preparation consists in configuring the

flash and RAM memories where the programmer is to be loaded, disabling interrupts, as well as in enabling and configuring semi-hosting.

- **Select the Cleanup Macro** file. The cleanup macro can be used to reverse the changes made by the startup macro.

- **Set the Programmer parameters**. Refer to your Programmer's documentation for a description of the Programmer options.

PARAMETER	COMMENT
Base Address	The start memory address of the flash. Passes to the programmer the -b\<base_addr\> option.
Erase	Flash erase control. • **Chip** – erases the entire flash device. • **Range** – erases all sectors containing the specified range. Eg., to erase one sector at 0x100_0000, the range can be set to 0x100_0000-0x100_0000. Passes to the programmer the -e\<addr\>-\<addr\> option.
Watchdog	A sequence of operations of writing to the watchdog register that "kicks the dog." • **Address** – watchdog register address. • **Value Sequence** – coma-separated list of values to be written to the watchdog register to "kick the dog." Passes to the programmer the -w\<addr\>:\<value1\>,\<addr\>:\<value2\>, … option.
Verbose	Generates extended debugging information from the programmer for troubleshooting purposes. Passes to the programmer the –v option.

PARAMETER	COMMENT
Options	Text field for entering special purpose programmer options not found on the Flash Device tab. As of this writing, the following such options are supported: • **-f<CCLK-kHZ>** – system clock frequency for the Philips LPC2xxx programmer. • **-p<mode>** – protection pin value (0 or 1) for the Mitsubishi flash devices (used by CFI programmers). An example of the Options field contents might look like this: -f17400 –p0.
Query Device	Button for querying the type of the memory device at the base address. • If a Common Flash Interface (CFI) compliant flash device is found, the size of the flash, the number and sizes of the sectors, and the Programmer-recommended name are displayed. (The flash log files described in the *Flash Programming Log* section provide more information about the device). The Memory Device Information dialog box appears, allowing you to set selected parameters to the values returned by the query (Figure 3). • If a non-CFI compliant device or RAM is found, a message informing you that ROM or RAM, respectively, has been found appears.

Memory Device Information

Two 16-bit Intel/Sharp Extended Command Set flash devices in parallel at address 0x00000000. Combined device size 0x01000000 (sectors: 64 x 0x20000 x 2). Manufacturer: Intel (0x89), device ID: 0x17

Recommended programmer: CFI_2x16_<load-address>.elf

Detailed information is displayed in the Log tab.

Do you want to update the Programmer with these parameters?

Yes   No

FIGURE 3 Flash query using the Query Device button. The results can be used to configure the Programmer automatically.

Flash Erasing, Programming and Verification

▶   To program the flash, verify the programming, or erase the flash

1. **Set up the flash programming parameters** in the Flash Image tab. Choose the image file type in the File Type drop-down list box. Then enter the flash image file name in the Image Name box, or browse for the file. You may need to enter the loading address offset in the Address Offset text box. This offset specifies the loading address for binary files or the offset added to the loading addresses in HEX, SREC and linker output files.
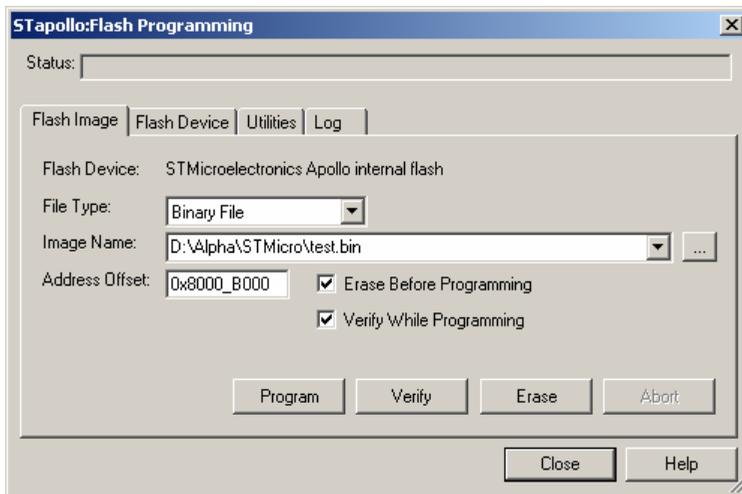
FIGURE 4 Selecting the flash image parameters and programming operations.

2. Program, Verify, Erase or Abort. This step depends on the intended operation. Some of these operations may require selecting additional options before you press the appropriate operation button.

PROGRAM BUTTON

Normally, programming the flash device requires earlier erasing the flash memory. Also, it is advisable that you verify the contents of the programmed flash. These two operations are controlled by the following options:

Erase Before Programming	Instructs the Programmer to erase the entire flash memory prior to uploading data to the device when the Program button is pressed. When uploading multiple files to the flash, deselect Erase Before Programming, and use the Erase button to clear the entire flash only before programming the first file.

Verify While Programming	Instructs the Programmer to perform its own programming verification. (This verification is not to be confused with the verification triggered by the Verify button, which is performed by the flash programming plug-in. See the *VERIFY BUTTON* section further in the text.)

Finally, press the Program button. Flash programming commences and the Status text box starts displaying progress report messages.
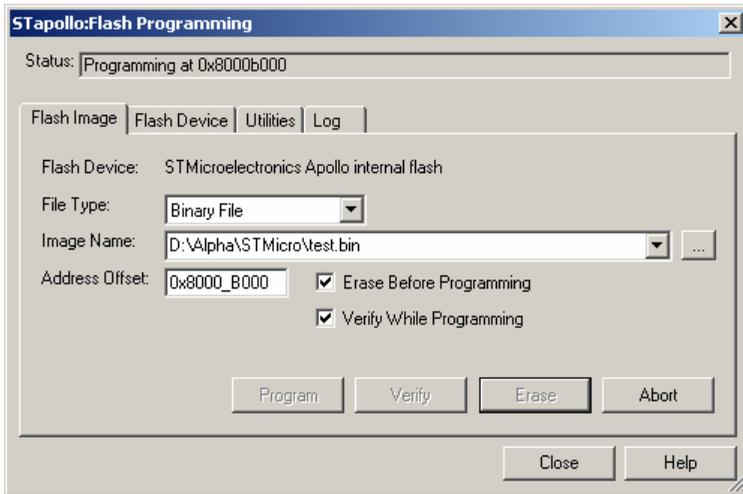
FIGURE 5 Programming operation under way. The status is displayed at the top of the dialog box.

VERIFY BUTTON

The Verify button is used to compare the data in a flash image file with the data read from the flash memory. (Please make sure that the Image Name and Address Offset are set appropriately before you press the Verify button.)

Since this verification is performed by the plug-in, and not by the Programmer, the Flash Device does not need to be specified. Verification performed using the Verify button is possible only if the entire flash memory to be verified can be read by the plug-in.

ERASE BUTTON

This button clears the entire flash memory. Make sure that the currently selected flash device is the same as the one used for programming. No other parameter is required.

ABORT BUTTON

To terminate the programming, verification or erasing operation, press the Abort button.

Note: *Some flash devices do not allow the erase process to be interrupted.*

Utility Functions

The flash programming plug-in provides two functions frequently used in connection with flash programming: Save and Blank Check. To execute these functions, select the Utilities tab.
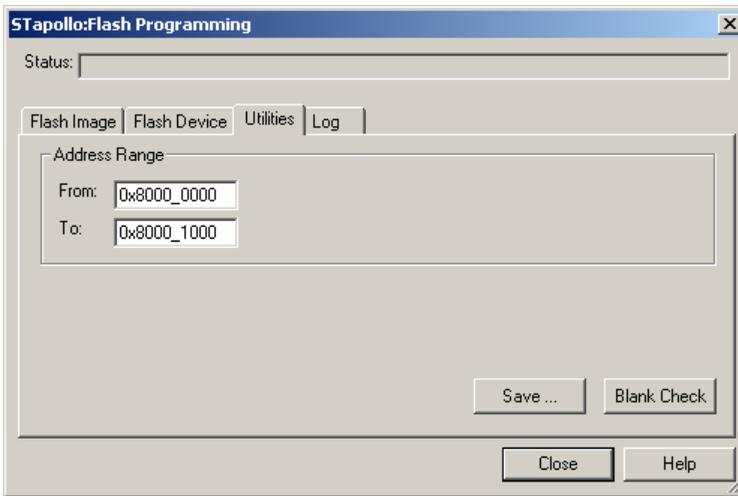
FIGURE 6 The handy Save and Blank Check functions are accessible via the Utilities tab.

SAVE

The Save function saves the flash memory within a given address range in a binary data file. It is equivalent to the debugger's SAVE BIN command. Recall that debugger commands can be executed in the Chameleon Command window. Since the Save function is not limited to flash memory only, you can use it to store memory blocks of any type. The Save function does not depend on any of the parameters set in the Flash Image or Flash Device tabs.

▶   To save a memory range to a file

1. Enter the memory range start and end addresses in the From and To text boxes, respectively.

2. Press the Save button. A file open dialog box appears.

3. Enter the file name, or browse for the binary file, in which the memory range data is to be saved.

BLANK CHECK

The Blank Check function is used to check if all the bits in the specified memory range are set to 1s. As flash memory is erased by setting its bits to 1s, the function verifies that the flash has been erased. The Blank Check function does not depend on any of the parameters set in tabs the Flash Image or Flash Device tabs.

▶ To verify that a memory range has been erased

1. Select the desired memory address range in the Address Range group.

2. Press the Blank Check button.

Flash Programming Log

The flash programming plug-in creates a log of flash programming activity. You can control the amount and type of information stored, save, copy and clear the log using the controls found in the Log tab.
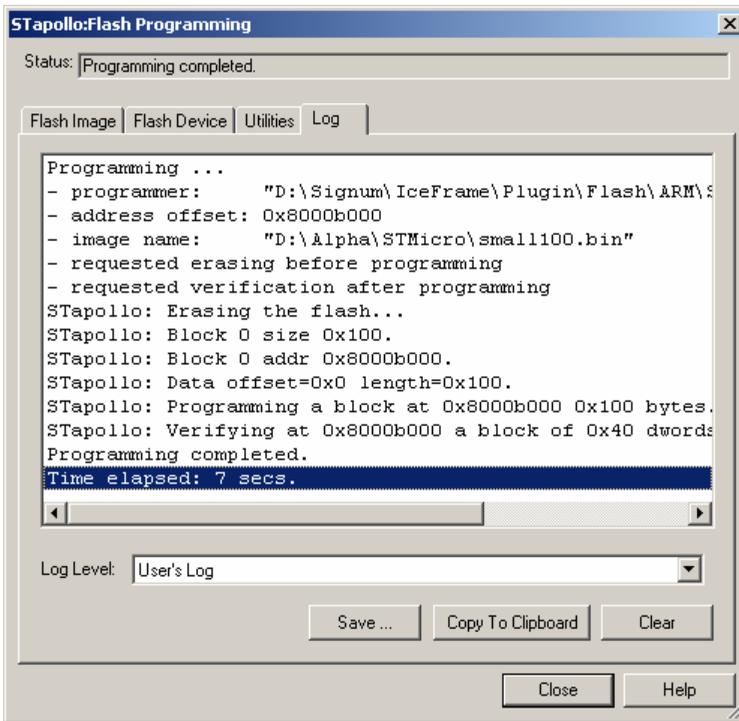
FIGURE 7 The flash programming log window.

LOG LEVEL

The log content can be filtered in several ways. Use the Log Level drop-down list to select the filtering criteria that fit your needs best. The available levels are listed below below in increasing order of comprehensiveness.

Errors Only   The log records only error messages.

Normal   In addition to Errors Only level messages, the log records commands (Erase, Program, Save, Bank Check, etc.) along with their parameters.

User's Log	In addition to Normal level messages, the log records messages from the Programmer execution on the target board. The Programmer's messages are prefixed by the target name. For example, a message from the STapollo target may look like this: "STapollo: Erasing the flash . . ."
Calls Except Data Access	In addition to User's Log level messages, the log records requests from the Programmer code to the flash programming plug-in, with the exception of those relating to flash image data.
All Calls	In addition to Calls Except Data Access level messages, the log records those requests from the Programmer to the plug-in that relate to the flash image.
All Calls and Data	In addition to All Calls level messages, the log records the flash image data passed from the plug-in to the Programmer.

The Normal log level is designed to help you keep track of the performed operation, allowing you to verify the data used to program the flash, destination addresses, and the like.

The User's Log and higher levels are designed to troubleshoot the programming process or debug the Programmer code. These levels degrade programming performance considerably, and therefore are not recommended to be used routinely.

SAVE

▶ To save the current log

1. Click the Save button. The Save As dialog appears.

2. Enter the name of the log file, or browse for the existing file.

3. Press OK to save the log to the file.

COPY TO CLIPBOARD

The Copy To Clipboard button copies the entire current log to the MS Windows clipboard.

CLEAR

The Clear button erases the current log from the flash programming log window.

Macro Flash Programming

Setting Plug-In Parameters for Macro Flash Programming

Perhaps the easiest way to create a macro that configures the plug-in is to set the parameters via the plug-in GUI interface, as discribed earlier in the *Configuring the Plug-In* section, and then to save the configuration commands to a file by pressing the Save Config button (Figure 2).

Plug-in Configuration Macrofile Example

```
plugin paramset FlashImage.EraseBeforeProgramming=1
plugin paramset FlashImage.VerifyWhileProgramming=1
plugin paramset FlashImage.AddressOffset=0
plugin paramset FlashImage.FileType=0
plugin paramset
FlashImage.ImageName="D:\Alpha\AT91EB40A\Flash\demo.elf"
plugin paramset FlashDevice.FlashDevice="LPC2xxx internal
flash programmer"
plugin paramset
FlashDevice.Programmer="D:\Chameleon\Plugin\Flash\ARM\LPC2xx
x\LPC2xxx.elf"
plugin paramset FlashDevice.StartupMacro="D:\ Chameleon
\Arm\TestFlashStartup.mac"
plugin paramset FlashDevice.CleanupMacro="D:\ Chameleon
\Arm\TestFlashCleanup.mac"
plugin paramset FlashDevice.Options="-f17400"
plugin paramset FlashDevice.BaseAddress="0x100_0000"
plugin paramset FlashDevice.Erase.From="0x100 0000"
plugin paramset FlashDevice.Erase.To="0x11F_FFFF"
```

```
    plugin paramset FlashDevice.Watchdog.Address="0x8001 00C0"
    plugin paramset FlashDevice.Watchdog.ValueSequence="1,0"
    plugin paramset FlashDevice.Verbose=1
    plugin paramset FlashDevice.Erase.Chip=0
    plugin paramset FlashDevice.Erase.Range=1
    plugin paramset Utilities.AddressRange.From=0x0100 0000
    plugin paramset Utilities.AddressRange.To=0x011F FFFF
    plugin paramset
    Utilities.Save.FileName="D:\Alpha\EB40A\EB40A.bin"
    plugin paramset Log.LogLevel=1
```

For added flexibility, absolute file pathnames may be converted into relative paths with the use of the $ character. The plug-in treats the $ as representing the path to the debugger installation directory. Thus, for example, it is possible to use in a macro file statements like this:

```
    plugin paramset
    FlashDevice.StartupMacro="$\Arm\TestFlashStartup.mac"
    plugin paramset
    FlashDevice.CleanupMacro="$\Arm\TestFlashCleanup.mac"
```

Plug-In Commands for Macro Flash Programming

In a macro file, all plug-in programmer commands—PROGRAM, ERASE and VERIFY—must be preceded by the keyword "flash," for example:

```
    flash program
    flash erase
```

Plug-in programmer commands are case-insensitive.

Putting it Together

Commands that configure the programmer and commands operating on flash memory can be combined together to fully automate the process of flash programming. An example of a complete macro file that first erases and then programs flash is shown below.

```
    ;   flash.mac – program my demo program into the flash
```

```
plugin paramset FlashDevice.FlashDevice="LPC2xxx internal
flash programmer"
plugin paramset
FlashDevice.Programmer="D:\Chameleon\Plugin\Flash\ARM\LPC2xx
x\LPC2xxx.elf"
plugin paramset FlashDevice.StartupMacro=" "
plugin paramset FlashDevice.CleanupMacro=" "
plugin paramset FlashDevice.BaseAddress="0x100 0000"
plugin paramset FlashDevice.Erase.Chip=1

flash erase

plugin paramset FlashImage.EraseBeforeProgramming=0
plugin paramset FlashImage.VerifyWhileProgramming=0
plugin paramset FlashImage.FileType=0 ; 0-linker file, 1-
HEX, 2-binary, 3-SREC
plugin paramset
FlashImage.ImageName="D:\Alpha\AT91EB40A\Flash\demo.elf"
plugin paramset FlashImage.AddressOffset=0

flash program
flash verify

; End of flash.mac
```

Appendix

Sharp KEV75401 Evaluation Board

This target board is equipped with the LH28F320BFE flash device.

▶ To enable the board for flash programming, set the jumpers as follows:

JUMPER	SETTING	FUNCTION
JP19	Opened	Flash boot block not protected
JP20	Opened	Bus width 16 bit
JP27	Closed	Flash write protection disabled

▶ To select a flash memory address, set the jumpers as follows:

JUMPER	SETTING	FUNCTION
JP21	2 – 3	Flash is selected by CS0 and
JP22	1 – 2	located at address 0x40000000.
JP21	1 – 2	Flash is selected by CS1 and
JP22	2 – 3	located at address 0x44000000.

- When the flash is selected by CS0, select the "Sharp KEV75401 with the LH28F320BFE flash with CS0" programmer in the Flash Device tab. (See the *Configuring the Plug-In* section.)

- Otherwise, select "Sharp KEV75401 with the LH28F320BFE flash with CS1."

▶   To refine flash programming, set the following Options in the Flash Device tab (Figure 2):

OPTION SYNTAX	FUNCTION
-b\<flash-base-address\>	Sets the flash base address
-e\<start_addr\>[-\<end_addr\>]	Erases the sectors in the specified address range
-v	Turns the verbose mode on for troubleshooting purposes

For example,

```
-b0x4000_0000 -e0x4000_0000-0x4000_FFFF -v
```

defines the flash base address as 0x4000_0000, instructs the programmer to erase the flash memory between address 0x4000_0000 through 0x4000_FFFF, and enables the verbose mode of the programmer.

❑