ARM

series

SIGNUM SYSTEMS CORPORATION

Chameleon Debugger for ARM and XScale

Installation Instructions

SIGNUM SYSTEMS

DISCLAIMER

Signum Systems makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Also, Signum Systems reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of Signum Systems to notify any person or organization of such revision or changes.

WARRANTY

Signum Systems warrants to the original purchaser that this product is free of defects in material and workmanship and performs to applicable published Signum Systems specifications for a period of TWELVE MONTHS from the date of shipment. If defective, the product must be returned to Signum Systems, prepaid, within the warranty period, and it will be repaired or replaced (at our option) at no charge. Equipment or parts which have been subject to misuse, abuse, alteration, neglect, accident, unauthorized installation or repair are not covered by warranty. This warranty is in lieu of any other warranty expressed or implied. IN NO EVENT SHALL SIGNUM SYSTEMS BE LIABLE FOR CONSEQUENTIAL DAMAGES OF ANY KIND. It is up to the purchaser to determine the reliability and suitability of this product for his particular application.

SIGNUM SYSTEMS

IAR SYSTEMS

1211 FLYNN RD., UNIT #104
CAMARILLO, CA 93012, U.S.A.
PHONE 805 • 383 • 3682
WWW.SIGNUM.COM

Purpose *This document describes the Chameleon Debugger software installation process for use with the Signum Systems JTAGjet emulator for ARM processors. For examples of connecting the JTAGjet to selected ARM and XScale target boards, please refer to "ARM Board Setup: User Guide."*

Note:
You must install the USB driver before this step. For instructions on USB driver installation, please refer to the "USB 2.0 Driver for JTAGjet and ADM51: Installation Instructions" document.

Installation Procedure

1. Insert the *JTAGjet* CD into your CD drive. From the Master Setup window, select Chameleon Debugger, and double-click ARM and XScale, as shown in Figure 1. (This selection includes the TI OMAP processor family.)

FIGURE 1 Chameleon Master Setup window.

2. Follow the online instructions to complete the installation process.

3. Start Chameleon Debugger and in the System Configuration window, click the **Add Target** button (Figure 2).

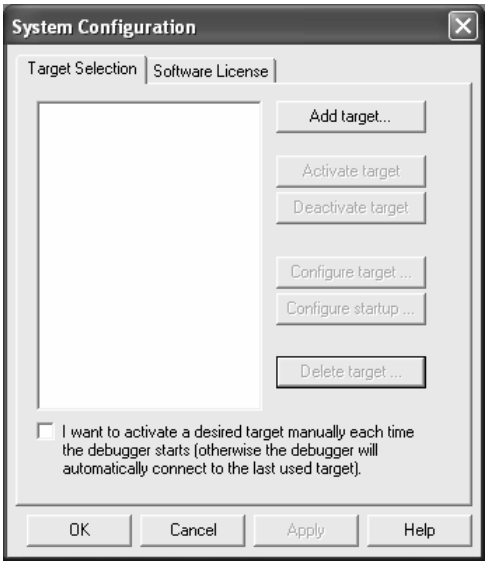

FIGURE 2 Creating a target in the debugger.

4. The Target Selection window appears. Click the **Enter a Key** button and copy the license key—it came with your Signum emulator, printed on a separate page titled **Product User License Certificate**—in the Enter Upgrade Key box. Click **OK**.

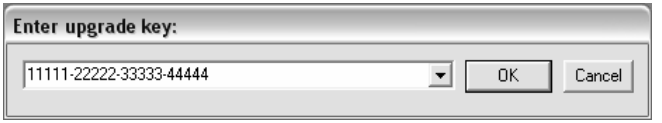

FIGURE 3 Entering the software activation/upgrade key. The actual key is included in the emulator package.

5. In the Target Selection dialog box, choose **ARM** as the CPU Family. In the Emulator/API section, highlight **JTAGjet (Signum Systems)**., as shown in Figure 4.

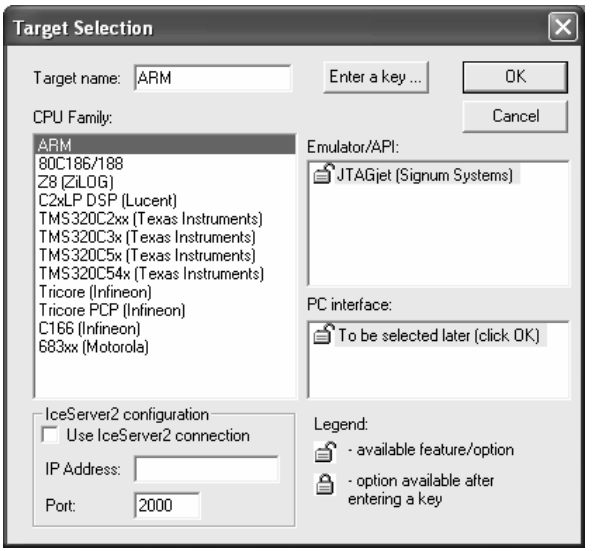

FIGURE 4 Selecting the emulator and its communication interface.

In the **Target name** text box, enter a name corresponding to your target board. The default name is "ARM" (see Figure 1), but your board's name or the project's title should provide a naming cue for differentiating the board from other ARM target boards, if any. Click OK. In the dialog box that appears, select the communication port used by the JTAGjet emulator.

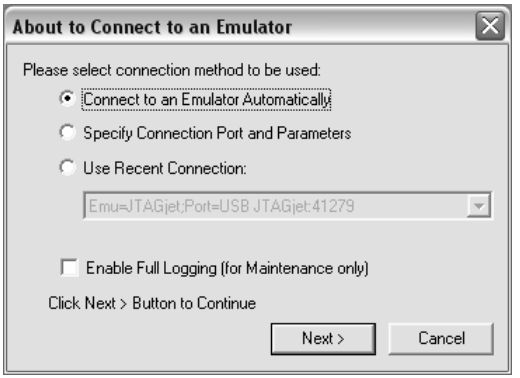

FIGURE 5 Choosing the emulator-PC communication method.

The hassle-free automatic method attempts to find the fastest connection available. If you need greater control over the connection, choose the "Specify Connection Port and Parameters" option.

6. The Startup Configuration Selection window displays a list of supported boards sorted by the manufacturer of the ARM device used on the board. (Figure 6).

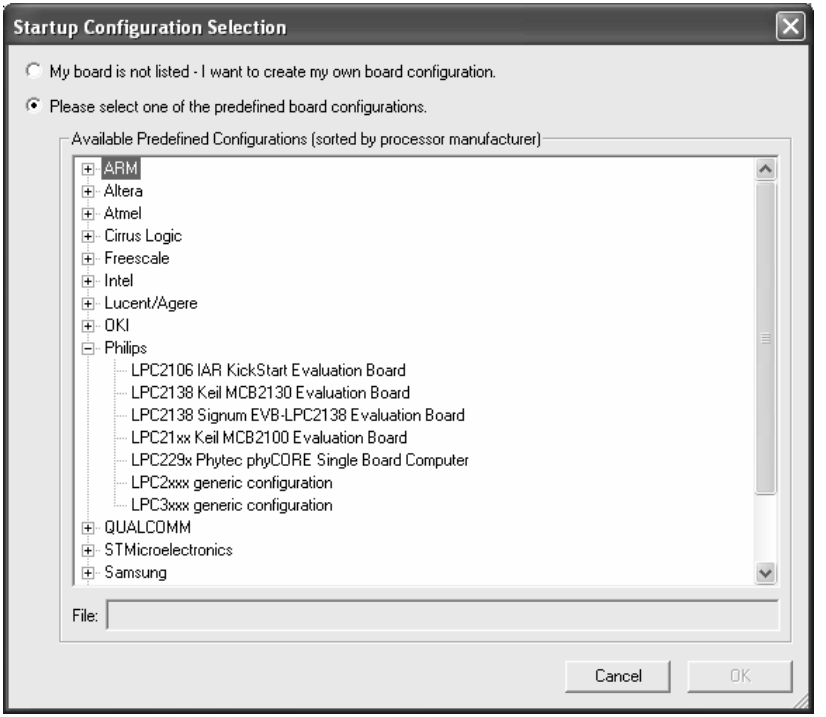

FIGURE 6 The Startup Configuration Selection window.

Please select your board from the list and click OK. The debugger's main window appears, signaling the completion of the target selection process.

If the power is applied to the board, the board is automatically configured by a startup script (.mac macro file). The macro file can be executed using a macro button on the toolbar menu. When clicked, the button displays a complete list of sub-routines found in the startup script file, enabling you to execute them selectively.

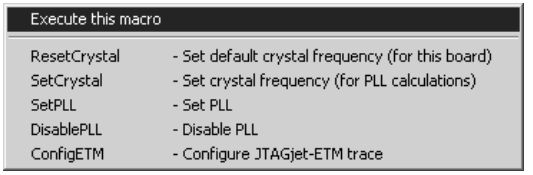

FIGURE 7

For information on other debugger features, please refer to the debugger's Help menu.

To add another target board to your system configuration, select System Configuration from the View menu. By repeating the process, you can populate the Target Selection list, which greatly facilitates switching from board to board or accessing them concurrently.

Skip to step 12, if your installation has been successful. Otherwise, proceed to the next point (7).

7. If your board is not listed in the Startup Configuration Selection window (Figure 6), or if it requires custom settings, select:

◉ My board is not listed – I want to create my own board configuration

Click OK. A CPU/ARM core selection list appears. It is recommended that you try to locate your CPU's name rather then the core name, as the former usually contains more information about the device. If the CPU name is not found in the list, it is very important to choose the correct ARM7, ARM9 or XScale core of your device or ASIC. Please refer to the device data sheet to verify your selection.

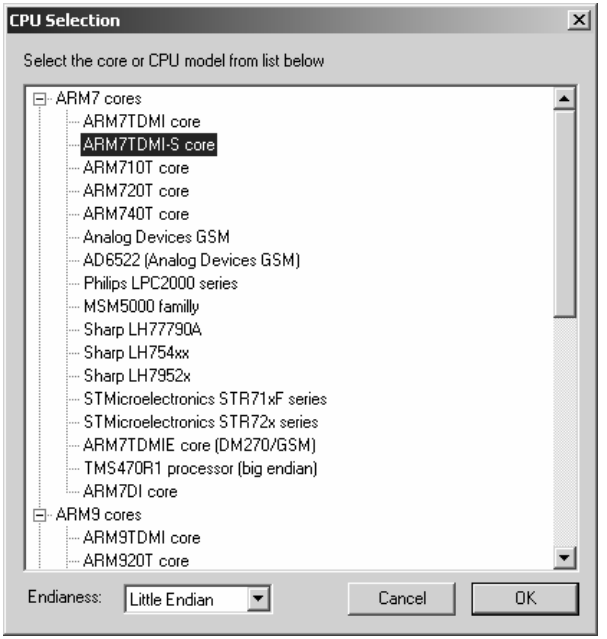

FIGURE 8 The CPU selection list.

8. When the proper core is selected, select the Endianess of the board. Most ARM devices and boards are configured for the Little Endian mode. Refer to the board manual if in doubt. Click OK

9. The next target configuration step requires the JTAG chain geometry, or the number and sizes of the devices on the JTAG chain.

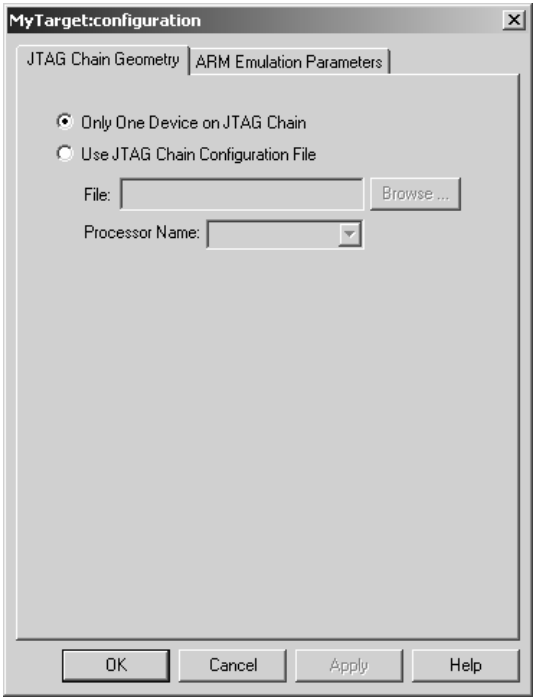

FIGURE 9 Configuring the JTAG chain.

Define the devices in the JTAG daisy chain using one of the two available options: Only One Device on JTAG Chain or Use JTAG Chain Configuration File.

◉ Only One Device on JTAG Chain

Select this option if there is only one device in the chain and skip the rest of this section to continue with setup.

◉ Use JTAG Chain Configuration File

If there are multiple devices in the chain, it is necessary to create an ASCII JTAG configuration file that specifies the symbolic names, types, and the lengths of the JTAG registers. While the file name is arbitrary, the extension must be .cfg.

Except for comments and empty lines, each line in the .cfg file refers to a separate device on the chain. Below is the file syntax:

```
NAME1          TYPE1
NAME2          TYPE2
[+]NAME3       TYPE3
     .
     .
     .
```

where

`NAME` is a unique name identifying the device, e.g., "CPU_A", including the required double quotes.

`TYPE` is the type of the device, such as ARM7TDMI, ARM940T, or PXA27x. To bypass a device, specify their type as the length of the instruction register in a two-digit format appended to the word BYPASS. For example, BYPASS2e denotes a bypassed device with a 46-bit (2e hex) instruction register. If TYPE contains non-alphanumeric characters, or begins with a digit, enclose it in double quotes, as in `"ARM926EJ-S"`.

`+` signifies the sharing of a single JTAG device on the chain between the processor (here NAME3) and the processor that precedes it on the list (NAME2). Use the + designator to list the cores interfaced with the JTAG chain through a single CoreSight Debug Access Port (DAP). For details, see See CoreSight Configuration further in the text.

Any text between the semicolon character and the end of the line is treated as a comment and ignored by the debugger. The order in which the JTAG devices are specified in the configuration file matters. Namely, the first line corresponds to the device closest to the TDI, the second line corresponds to the next device in the chain, and so on. The last line refers to the device on the TDO end of the chain.

A sample JTAG configuration file is shown below.

```
;*********************************************
;* JTAG configuration file
;*********************************************

; Emulator TDI

"ARM1"    ARM7TDMI
"ARM2"    OMAP
"DSP"     BYPASS2e        ; IR=46 dec
"ARM3"    "ARM926EJ-S"

; End of file
```

CoreSight Configuration

Multiple Cortex processors can be connected to the JTAG chain through a single CoreSight Debug Access Port (DAP). The names of the devices on a single DAP are prefixed with a plus sign (+). The general format for specifying a CoreSight configuration is like this.

```
NAME1 TYPE1[:port[@address,options]]
+NAME2 TYPE2[:port[@address,options]]
    .
    .
    .
```

For instance,

```
"M4" "Cortex-M4"
+"M0" "Cortex-M0"
```

The order of the processors on a single CoreSight DAP is arbitrary and has no significance.

The CoreSight interface allows the debugger to automatically recognize the devices connected to the DAP by relying on the information stored in the CoreSight ROM table. However, if the assignment of the processor to a DAP port cannot be unambiguously determined, the port number must be specified explicitly, as in the following example.

```
"1R4" "Cortex-R4:1"
+"1M3" "Cortex-M3:3"
```

Similarly, if the CoreSight ROM table address is unknown, the processor address of the debug port must be specified explicitly:

```
"A8" "Cortex-A8:1@0xE0008000"
```

If no CoreSight ROM table exists, any attempt to access it may lock access to the CoreSight ports. To avoid the problem, use the NOROM option which prevents the debugger from trying to read a nonexistent ROM table:

```
"A9_0"  "Cortex-A9:1@0xC0000000,norom"
+"A9_1" "Cortex-A9:1@0xC0002000,norom"
```

When a correct JTAG chain configuration file is created, use the **Browse** button to direct the debugger to the file, and select the proper device in the **Processor Name** filed.

Filename — Specifies the JTAG configuration file. The file must reside in the directory where Chameleon Debugger was installed—typically, C:\Signum\Chameleon.

Processor name — Specifies the name of the emulated JTAG device. Choose the device that needs to be debugged.

10. Click on the **ARM Emulation Parameters** tab. This tab allows you to control semihosting (virtual I/O), the JTAG clock speed, the JTAG pin configuration, and the reset timing. Note the difference between the tab for ARM targets and the tab for XScale targets (Figure 10).

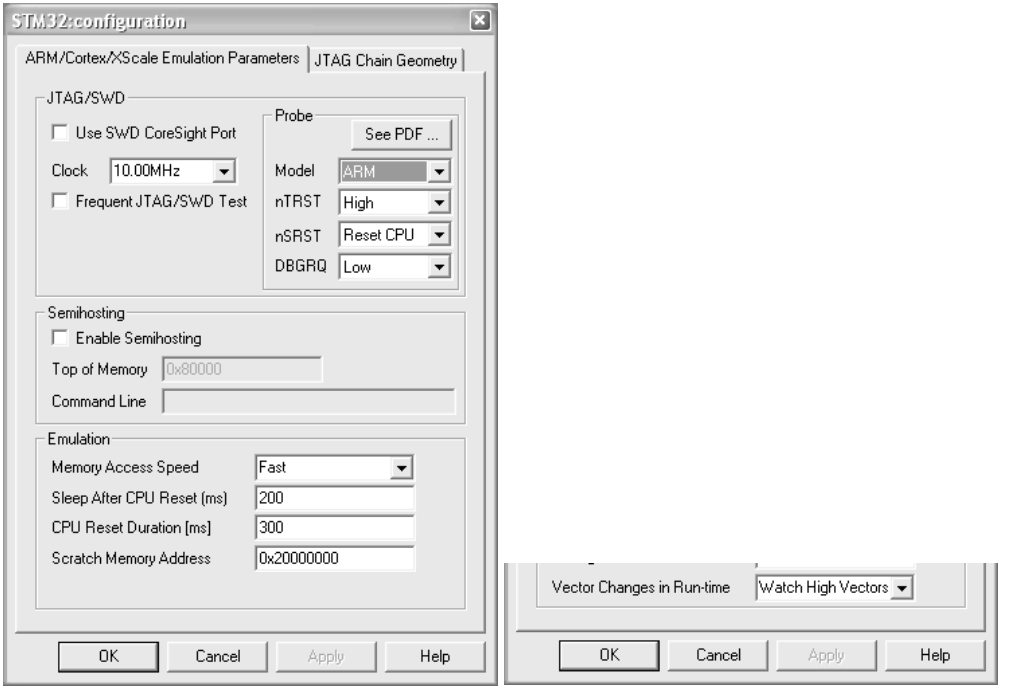

FIGURE 10 The ARM/Cortex/XScale Emulation Parameters tab for ARM targets (left) and XScale targets (right).

The controls on the Arm/Cortex/XScale Emulation Parameters tab are divided into several groups: JTAG/SWD, Probe, Semihosting, and Emulation.

JTAG/SWD

The JTAG/SWD section enables you to select the JTAG clock frequency. If your core has the –S suffix in its name, it internally synchronizes the JTAG TCK clock with the core clock and returns it to the emulator. If your core supports such an adaptive clocking mechanism (RTCK), select Adaptive from the Clock drop-down menu.

For cores without the RTCK, select a fixed JTAG clock frequency (typically, 10MHz). JTAG clock up to 30MHz can be used with processors with higher CPU clock rates to improve download times.

Check the Use SWD CoreSight Port check box when using the Serial Wire Debug (SWD) interface. The SWD interface is available with Cortex-M processors. The interface uses only 2 pins, leaving the remaining JTAG pins on the processor for the application's needs.

Probe

The Probe section allows you to select the JTAG probe and configure the JTAG lines.

Set the type of the JTAG probe by selecting the appropriate Model option.

MODEL	SIGNUM PROBE/CONNECTOR
ARM	ARM-20, ARM-14, ARM-20-LV
ARM-SWD	ARM-SWD Required to connect through the CoreSight SWD interface.
TI	TMS320 and TMS320-LV
cTI	CTI-20 Unlike the TMS320 probes, it has a system reset line.
ETM	38pin Mictor ETM connector

The rest of the settings in the Probe group are set to ARM default values and should not be modified.

Semihosting

The Enable Semihosting option turns on the debugger's support for semihosting (Virtual I/O). If selected, it makes the debugger catch all the SWI calls made by the semihosting libraries (printf, scanf, etc.) for redirection to the proper OS calls.

If **Enable Semihosting** is not checked, while your application calls a semihosting library function, the application will start executing code from the SWI vector location and most likely the program will crash due to an attempt to execute some non-existing code. To avoid the problem, enter the **Top Of**

Memory parameter to reflect the size of your target board's RAM memory. This parameter allows the ARM run-time library to determine where the heap and stack should be placed in memory. For example, set top_of_memory (a hex value) to

80000 for the evaluation board ARM Evaluator 7T

– or –

9000000 for the evaluation board Cogent CDK238.

If you do not use the C/C++ runtime library with semihosting support, disable semihosting. This will make one additional hardware breakpoint on ARM7 cores available.

Emulation

The **Emulation** section allows you to adjust the memory and reset the timing. It is recommended to leave the Memory Access Speed set to FAST. Initially, the Reset parameters are set to default values, but they can be adjusted as needed. Also the JTAG configuration settings can be modified at any time during debugging with the use of the **View/Target Configuration** menu.

Debug Handler Address (XScale only)

On XScale processors, it is necessary to specify a 2KB memory region dedicated exclusively to the debug handler. This setting becomes the handler code's base address. It must be reserved strictly for this purpose, be 2KB in size, and be aligned to a 2KB address. As a result, the address's 11 least significant bits must be zero.

11. The next step in custom target configuration is to select a Startup Macro. At this point a list of macro files will appear. These macros are for all target boards we know of. Obviously, if you were creating your own target configuration you will not find the macro on this list. However, if you have prepared your own macro ahead of time or it was given to you by someone else, please browse to the proper directory and select it here.

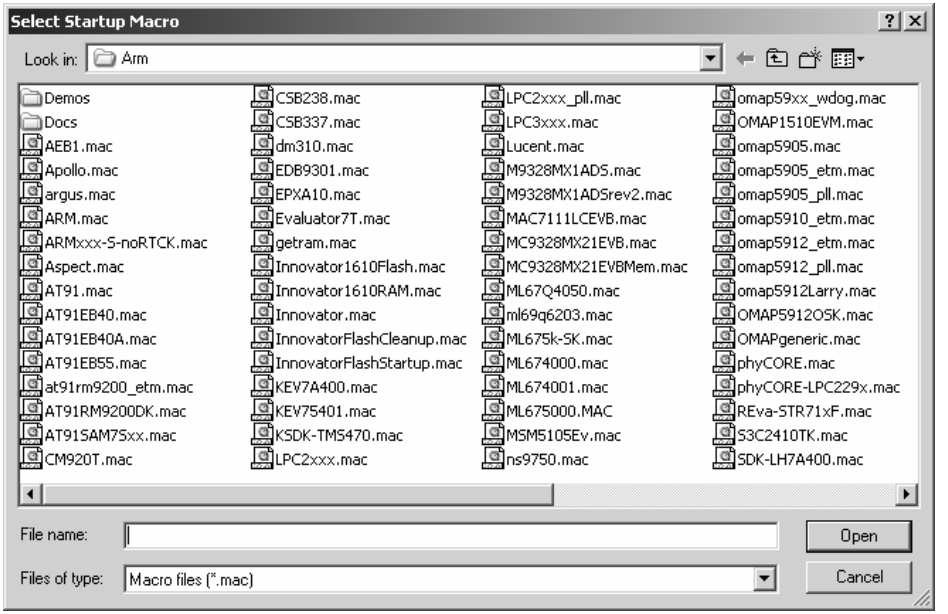

FIGURE 11 Selecting the startup macro file.

This concludes the custom target setup process.

12. After the debugger establishes connection with the CPU, a typical initial screen will look like this.

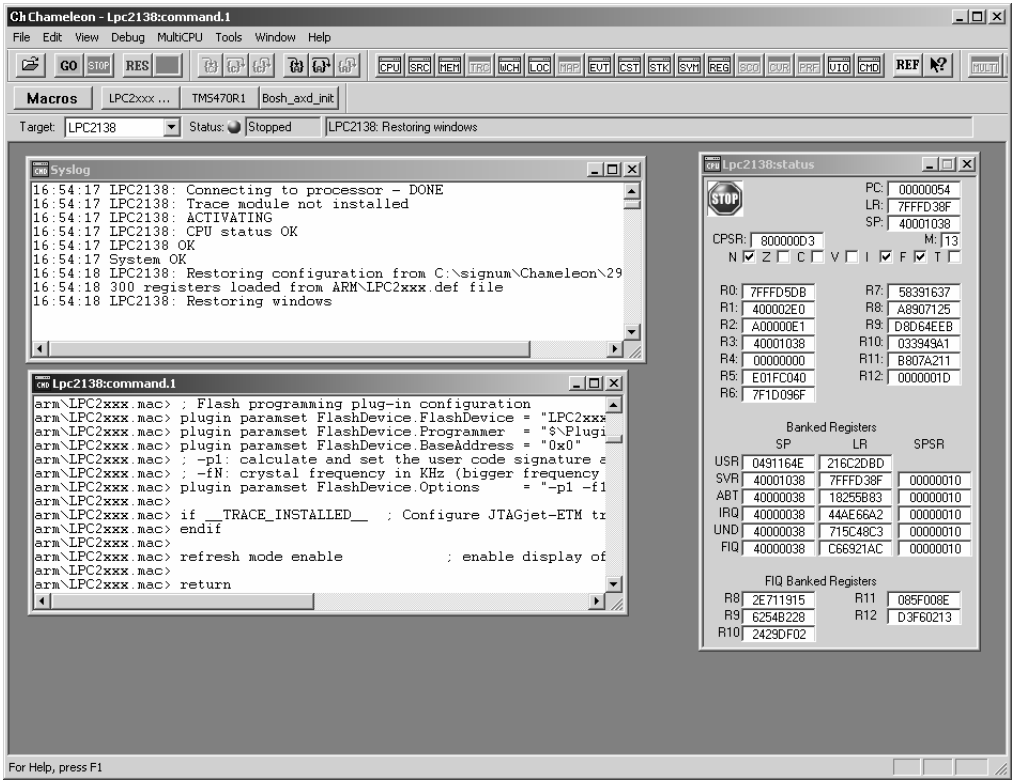

FIGURE 12 Chameleon's opening screen.

It is recommended that you familiarize yourself with the documents provided in the Chameleon Help menu before continuing with your project.

If you encounter difficulties when installing or starting the debugger, please provide the Signum Technical Support team with the .log and .ini files found in the Chameleon installation folder. For contact and additional information, please visit www.signum.com/support.htm.

❑