ColdFire® IAR C-SPY hardware debugger systems

User Guide

for Freescale's

ColdFire® Microcontroller Family

EDITION NOTICE

First edition: September 2007

Part number: CSCFB-1

This guide applies to version 1.x of ColdFire IAR Embedded Workbench®.

Internal reference: ISUD.

Contents

Introduction to C-SPY® hardware debugger systems

This guide introduces you to the ColdFire IAR C-SPY hardware debugger systems and to how they differ from the IAR C-SPY Simulator.

This guide assumes that you already have some working knowledge of the target system you are using, as well as some working knowledge of the IAR C-SPY Debugger. For a quick introduction, see the tutorials in the *IAR Embedded Workbench® IDE User Guide*.

Note that a few additional features may have been added to the software after this guide was written. The release notes available in the file cscfb.htm provide the latest information.

The IAR C-SPY hardware debugger systems

The C-SPY Debugger consists of both a generic part which provides a basic set of C-SPY features, and a driver. The C-SPY driver is the part that provides communication with and control of the target system. The driver also provides a user interface—special menus, and dialog boxes—to the functions provided by the target system.

For further details about the concepts that are related to the IAR C-SPY Debugger, see the *IAR Embedded Workbench® IDE User Guide*.

The IAR C-SPY Debugger for the ColdFire microcontroller supports evaluation boards that have a standardized 6-pin BDM (background debug mode) connector or a standardized 26-pin BDM connector. Background debug mode is used for system development, in-circuit testing, field testing, and programming.

The IAR C-SPY Debugger for the ColdFire microcontroller are available with drivers for the following hardware debug interfaces:

- J-Link for ColdFire® Processors (IAR Systems)
- J-Link ColdFire BDM 26 (Segger Microcontroller Systeme)
- USB Multilink (P&E Microcomputer Systems)
- Cyclone PRO and Cyclone MAX (P&E Microcomputer Systems).

In addition to the C-SPY driver, a hardware interface driver DLL is used for communicating with the BDM interface. This driver communicates with the BDM interface module over a parallell, serial, Ethernet, or USB connection.

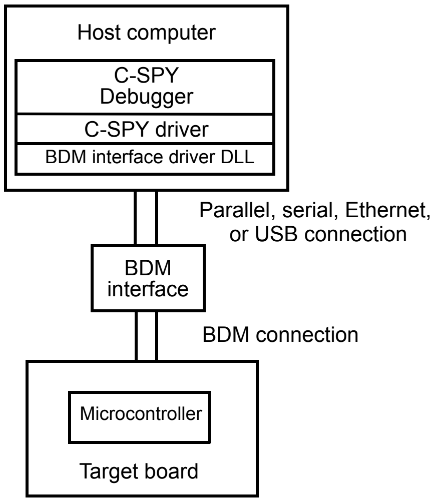


Figure 1: C-SPY BDM debugger communication overview

DIFFERENCES BETWEEN THE C-SPY DRIVERS

The following table summarizes the key differences between the C-SPY drivers:

Feature	**Simulator**	**J-Link**	**Multilink and Cyclone PRO/MAX**
Flash loading	--	x	x
Code breakpoint (OP-fetch)	x	x	x
Data breakpoints	x	--	--
Execution in real time	--	x	x
Simulated interrupts	x	--	--
Real interrupts	--	x	x
Cycle counter	x	--	--
Code coverage	x	--	--
Data coverage	x	--	--
Profiling	x	x *	x * †

Table 1: Differences between available C-SPY drivers

*** Cycle counter statistics are not available.**
† Profiling works provided that enough breakpoints are available.

Getting started

In this example, a demo project is set up for the IAR Systems MCF52223-SK board and for the Freescale M52223EVB board, where the LEDs are connected to a general purpose I/O port. When you run the demo program, the LEDs will blink.

You can find two ready-made workspaces, `JLINK_demo.eww` and `PEBDM_demo.eww`, together with source files in the `cf\src\examples` directory.

There are two projects, one each for the two evaluation boards. The first project contains the `LCDmain.c` file, which sends out data to `PORTAN`. The second project contains the `GPIOmain.c` file, which sends out data to `PORTTC`.

Note: The procedure in this example can be applied to other evaluation boards as well.

RUNNING THE DEMO PROGRAM

To run the demo program, follow this procedure.

1. To open the demo workspace:

   - Choose **Help>Startup Screen**
   - Click **Example workspaces** to open the **Open Example Workspace** dialog box
   - To open the workspace, choose either **JLINK_demo** or **PEBDM_demo** depending on your debug interface.

2. In the workspace window, select either the **lcd** or the **gpio** project, for J-Link or P&E BDM respectively. Choose **Project>Options**. In addition to the factory settings, make sure the following options are selected:

Category	Page	Option/Setting
General Options	Target	Device: **MCF52223** 1)
Debugger	Setup	Driver: **J-Link** or **PEmicro BDM**
PEmicro BDM	Communication	Communication type: P&E Multilink USB 2)

Table 2: Project options for C-SPY debugger example

1) A default linker command file (`xcl`) and device description file (`ddf`) will automatically be used depending on your choice of device.
2) If you use a different communication type, you may have to specify a port, accordingly.

For further details about the C-SPY options and how to configure C-SPY to interact with the target board, see *C-SPY options for debugging using hardware systems*, page 5.

Click OK to close the **Options** dialog box.

3 Click the **Make** button to compile and link the program, which should build without any diagnostic messages.

4 Start C-SPY by clicking the **Debug** button or by choosing **Project>Debug**. While debugging, useful information is displayed in the Debug Log message window. To open this window, choose **View>Messages>Debug Log**.

If C-SPY should fail to establish contact with the target hardware, see *Resolving problems*, page 12.

5 Set a breakpoint on line:

```
delay(100);
```

Note: The number of physical hardware breakpoints used when setting breakpoints is limited. To read more about this, see *Using breakpoints*, page 10.

6 Click the **Go** button a few times to see how the LEDs on the board change as different values are written to the port.

7 To see the LEDs blink continuously, remove the breakpoint and press **Go**.

8 Click the **Break** button to stop execution.

Debugging using hardware systems

This chapter describes the options, settings, and configurations needed for using the C-SPY hardware debugger systems. The application can be run in real-time when using these features, which provides a powerful tool for locating problems in the application or the hardware. More specifically, this chapter contains the following sections:

- C-SPY options for debugging using hardware systems
- Debugger menu
- Using breakpoints
- Resolving problems.

C-SPY options for debugging using hardware systems

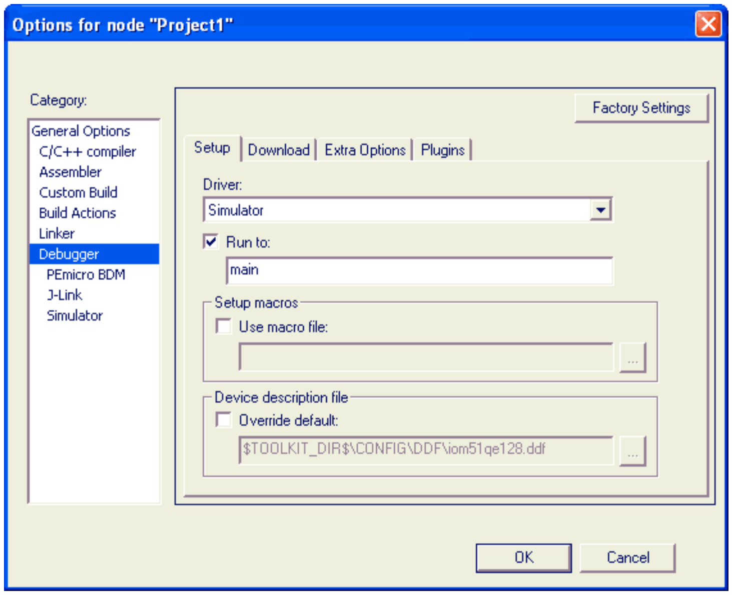
Before you start any C-SPY hardware debugger you must set some options for the debugger system—both generic options and options required for the specific system you are using. Follow this procedure:

1 To open the **Options** dialog box, choose **Project>Options**.

2 To set C-SPY generic options, select **Debugger** from the **Category** list.

3 On the **Setup** page, select the appropriate C-SPY driver from the **Driver** list. Choose between:

- **J-Link** for one of the interfaces from either IAR Systems or from Segger
- or **PEmicro BDM** for any of the interfaces from P&E.



Driver: Simulator

[✓] Run to: main

[ ] Use macro file:

[ ] Override default: $TOOLKIT_DIR$\CONFIG\DDF\iom51qe128.ddf

Figure 2: Project options dialog box

For information about the settings **Setup macros**, **Run to**, and **Device descriptions**, as well as for information about the pages **Extra Options** and **Plugins**, see the *IAR Embedded Workbench® IDE User Guide*.

Note that a default linker command file and device description file are automatically selected depending on your selection of device on the **General Options>Target** page. Use the **Override device description file** check box and the browse button if you want to override the default device description file.

4 To set options for the download, click the **Download** tab. For details about these options, see *PEmicro BDM Setup options*, page 8.

5 To set the options specific to the P&E interfaces, select **PEmicro BDM** from the **Category** list. Note that these options are only available when you have selected **PEmicro BDM** from the **Driver** list on the **Debugger>Setup** page.

Note: There are no options specific to J-Link required.

6 To set options for the P&E interface, click the **Setup** tab. For details about these options, see *PEmicro BDM Setup options*, page 8.

7 To set the options for communication, click the **Communication** tab. For details about these options, see *Communication options*, page 9.

8 When you have set all the required options, click **OK** in the **Options** dialog box.

DOWNLOAD OPTIONS

The **Download** page contains the options for configuring the download.

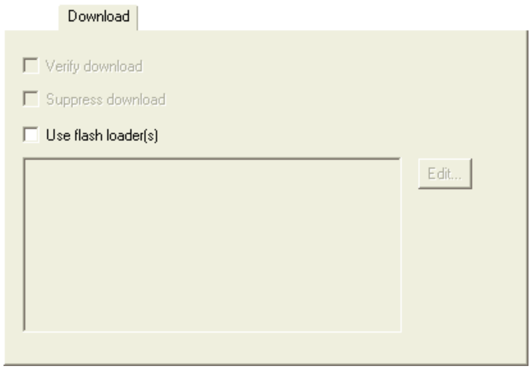


Figure 3: Download page

Verify download

Use this option to check every byte after loading to verify the download and that the memory of the target hardware is writable and mapped in a consistent way. A warning message will be generated if there are any errors during download.

Suppress download

Use this option to disable download. This option is useful if you already have your application in memory and you want to:

- minimize the number of times you write to flash
- disable the time-consuming download.

The implicit `RESET` performed at C-SPY startup is not disabled.

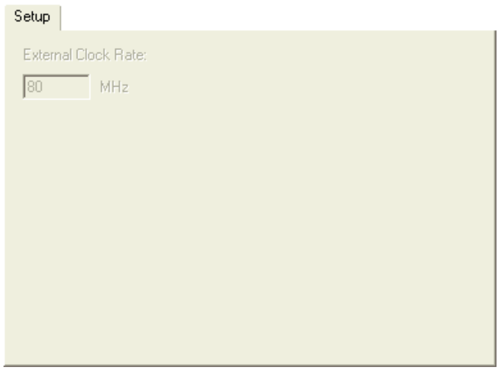
Use flash loader(s)

Use the **Use flash loader(s)** option to use one or several flash loaders for downloading your application to flash memory. If a flash loader is available for the selected device, it will be used as default. Press the **Edit** button to open the **Flash Loader Overview** dialog box.

To read more about flash loaders, see *Using flash loaders*, page 15.

PEMICRO BDM SETUP OPTIONS

The **Setup** page contains the options for debugging using a P&E interface.



External Clock Rate: 80 MHz

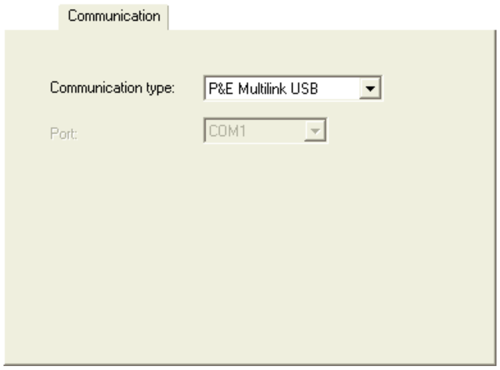
Figure 4: PEmicro BDM page

External Clock Rate

Use this option to set a value of the external clock rate.

COMMUNICATION OPTIONS

The **Communication** page contains all the communication options.



Communication type: P&E Multilink USB

Port: COM1

Figure 5: Communication page

The following settings are available:

Setting	Description
Communication type	Selects one of the supported communication types: • P&E Multilink USB • Cyclone MAX Serial • Cyclone MAX USB • Cyclone MAX Ethernet.
Port	Selects one of the supported ports. Choose between COM1–4.

Table 3: Serial Communication options

Debugger menu

When running any of the C-SPY hardware debugger systems, a dedicated menu appears.

Figure 6: Debugger menu

The following command is available on the menu:

Breakpoint usage	Opens the Breakpoint Usage window, see *Breakpoint Usage dialog box*, page 11.

Using breakpoints

This section provides information about breakpoints specific to the debugger system you are using. The following is described:

- *Hardware and software breakpoints*, page 10
- *Breakpoint Usage dialog box*, page 11.

For information about the different methods for setting breakpoints, the facilities for monitoring breakpoints, and the different breakpoint consumers, see the *IAR Embedded Workbench® IDE User Guide*.

HARDWARE AND SOFTWARE BREAKPOINTS

The physical breakpoint used on the target can be one of two types—*hardware* and *software*.

This table summarizes the characteristics of hardware and software breakpoints:

Type	**Number**	**Execution overhead**
Hardware	Normally four	No
Software (statement located in RAM memory)	Unlimited	Yes
Software (statement located in flash memory) *	Unlimited	Yes †

Table 4: Hardware and software breakpoints

*** Software breakpoints in flash memory is only possible when using the J-Link interface.**
† Slightly more overhead than for software breakpoints in RAM memory.

For the J-Link interfaces, C-SPY will use a hardware breakpoint if available; if not available, a software breakpoint is used.

For the P&E interfaces, C-SPY will initially always try to use a software breakpoint. If this is not possible, because your application is not located in read/write memory, C-SPY will try to use a hardware breakpoint. If all hardware breakpoints are occupied, C-SPY will issue a message in the Debug Log window. You can get information about the used breakpoints in the **Breakpoints Usage** dialog box.

Hardware breakpoints

Hardware breakpoints are available in the microcontroller and can be set in any type of memory (RAM, ROM, or flash). The number of breakpoints is normally four.

Software breakpoints

For the J-Link interfaces, software breakpoints can be used if the statement is located in either RAM or flash memory. Software breakpoints imply some execution overhead, which will be slightly more for flash memory than for RAM memory.

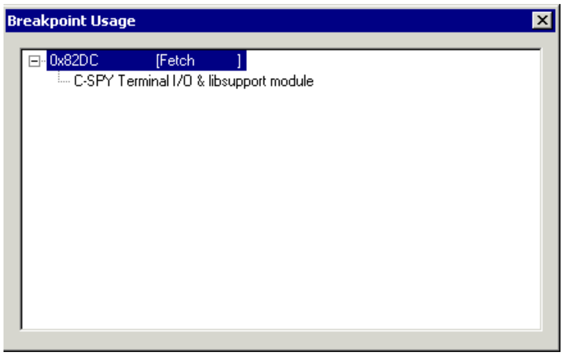
For P&E interfaces, software breakpoints can only be used when the statement you want to set a breakpoint on is located in read/write memory. This means that you have to temporarily link your application, or parts of it, so that it is located in read/write memory.

Software breakpoints are implemented in such a way that they temporarily substitute the actual instruction with the `HALT` instruction. Before resuming execution, the original instruction will be restored. This will generate execution time overhead when running an application.

Note: It is not possible to set software breakpoints on consecutive 1-byte instructions.

BREAKPOINT USAGE DIALOG BOX

The **Breakpoint Usage** dialog box—available from the driver-specific menu—lists all active breakpoints.



0x82DC [Fetch]
C-SPY Terminal I/O & libsupport module

Figure 7: Breakpoint Usage dialog box

In addition to listing all breakpoints that you have defined, this dialog box also lists the internal breakpoints that the debugger is using.

For each breakpoint in the list, the address and access type are shown. Each breakpoint in the list can also be expanded to show its originator.

Resolving problems

Debugging using a hardware debugger system requires interaction between many systems, independent from each other. For this reason, it can be a complex task to set up this debug system. If something goes wrong, it may at first be difficult to locate the cause of the problem.

This section includes suggestions for resolving the most common problems that can occur when debugging.

For problems concerning the operation of the evaluation board, refer to the documentation supplied with it, or contact your hardware distributor.

WRITE FAILURE DURING LOAD

There are several possible reasons for write failure during load. The two most common are that your application has been incorrectly linked or that the wrong device description file is used.

Check that you are using correct files.

If you are using the IAR Embedded Workbench IDE, both of these files are automatically selected based on your choice of device:

- Choose **Project>Options**
- Select the **General Options** category
- Click the **Target** tab
- Choose the appropriate device from the **Device** drop-down menu.

To override the default DDF file, see *C-SPY options for debugging using hardware systems*, page 5. To override the default linker command file:

- Choose **Project>Options**
- Select the **Linker** category
- Click the **Config** tab
- Choose the appropriate linker command file in the **Linker command file** field.

NO CONTACT WITH THE TARGET HARDWARE

There are several possible reasons for C-SPY to fail to establish contact with the target hardware.

- Check the communication devices on your host computer
- Verify that the cable is properly plugged in and not damaged or of the wrong type
- Verify that the target chip is properly mounted on the evaluation board
- Make sure that the evaluation board is supplied with sufficient power
- Check that the correct options for communication have been specified in the IAR Embedded Workbench; see *Communication options*, page 9

- Examine the linker command file to make sure that the application has not been linked to the wrong address.

Using flash loaders

This chapter describes the flash loader, what it is and how to use it.

The flash loader

A flash loader is an agent that is downloaded to the target. It fetches your application from the C-SPY debugger and programs it into flash memory. The flash loader uses the file I/O mechanism to read the application program from the host. You can select one or several flash loaders, where each flash loader loads a selected part of your application. This means that you can use different flash loaders for loading different parts of your application.

The flash loader API, documentation, and implementation templates are available to make it possible for you to implement your own flash loader.

SETTING UP THE FLASH LOADER(S)

To use a flash loader for downloading your application:

1 Choose **Project>Options**.

2 Choose the **Debugger** category and click the **Download** tab.

3 Select the **Use Flash loader(s)** option, and click the **Edit** button.

4 The **Flash Loader Overview** dialog box lists all currently available flash loaders; see *Flash Loader Overview dialog box*, page 17. You can either select a flash loader or open the **Flash Loader Configuration** dialog box.

In the **Flash Loader Configuration** dialog box, you can configure the download. For reference information about the different flash loader options, see *Flash Loader Configuration dialog box*, page 18.

Setting up the target system using a C-SPY macro file

You can use a C-SPY macro to set up the target system before loading the flash loader to RAM. One example when this is useful is for targets where the RAM is not functional after reset; the macro is used for setting up the necessary registers for proper RAM operation.

The following criteria must be met for a macro function to be executed before downloading the flash loader:

- The macro file must be located in the same directory as the flash loader

- The macro file must have the filename extension `mac`
- The name of the macro file must be the same as the flash loader
- The setup macro function `execUserFlashInit` must be defined in the macro file. This macro function is called from the debugger before the flash loader is loaded into RAM. Note that debugging while when the flash loader is running as an application, the setup macro `execUserPreload` must be used instead of `execUserFlashInit`.

THE FLASH LOADING MECHANISM

When the **Use flash loader(s)** option is selected and one or several flash loaders have been configured, the following steps will be performed when the debug session starts:

1. C-SPY downloads the flash loader into target RAM.
2. C-SPY starts execution of the flash loader.
3. The flash loader opens the file holding the application code.
4. The flash loader reads the application code and programs it into flash memory.
5. The flash loader terminates.
6. C-SPY switches context to the user application.

The steps 1 to 5 are performed for each selected flash loader.

BUILD CONSIDERATIONS

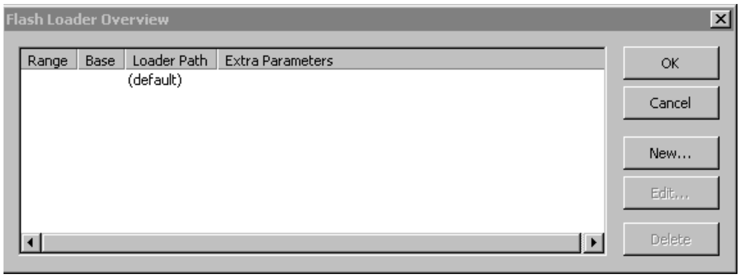
When you build an application that will be downloaded to flash, special consideration is needed. Two output files must be generated. The first is the usual UBROF file (`d68`) that provides the debugger with debug and symbol information. The second file is a simple-code file (filename extension `sim`) that will be opened and read by the flash loader when it downloads the application to flash memory.

The simple-code file must have the same path and name as the UBROF file except for the filename extension.

To create the extra output file, choose **Project>Options** and select the **Linker** category. Select the **Allow C-SPY-specific extra output file** option. On the **Extra Output** page, select the **Generate extra output file** option. Choose the **simple-code** output format and the format variant **None**. Do not override the default output file. For reference information about these options, see *Output*, page 405.

FLASH LOADER OVERVIEW DIALOG BOX

The **Flash Loader Overview** dialog box—available from the **Debugger>Download** page—lists all defined flash loaders. If you have selected a device on the **General Options>Target** page for which there is a flash loader, this flash loader is by default listed in the **Flash Loader Overview** dialog box.



Range	Base	Loader Path	Extra Parameters
		(default)	

Figure 8: Flash Loader Overview dialog box

The following function buttons are available:

Button	Description
OK	The selected flash loader(s) will be used for downloading your application to memory.
Cancel	Standard cancel.
New	Opens the **Flash Loader Configuration** dialog box where you can specify what flash loader to use; see *Flash Loader Configuration dialog box*, page 18.
Edit	Opens the **Flash Loader Configuration** dialog box where you can modify the settings for the selected flash loader; see *Flash Loader Configuration dialog box*, page 18.
Delete	Deletes the selected flash loader configuration.

Table 5: Function buttons in the Flash Loader Overview dialog box

FLASH LOADER CONFIGURATION DIALOG BOX

In the **Flash Loader Configuration** dialog box—available from the **Flash Loader Overview** dialog box—you can configure the download.

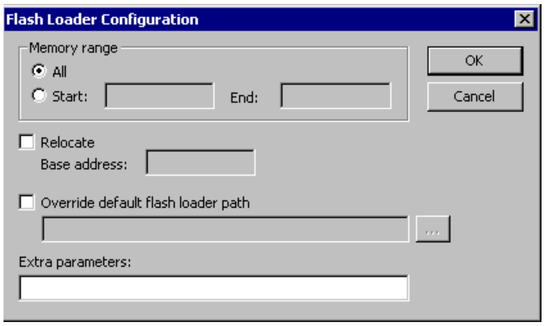


Figure 9: Flash Loader Configuration dialog box

Memory range

Use the **Memory range** options to specify the part of your application to be downloaded to flash memory. Choose between:

All	The whole application is downloaded using this flash loader.
Start/End	The part of the application available in the memory range will be downloaded. Use the **Start** and **End** text fields to specify the memory range.

Relocate

Use the **Relocate** option to override the default flash base address. The default base address used for writing the first byte—the lowest address—to flash is specified in the linker command file used for your application. However, it can sometimes be necessary to override the flash base address and start at a different location in the address space. This can, for example, be necessary for devices that remap the location of the flash memory.

Use the **Base address** text box to specify a numeric value for the new base address. You can use the following numeric formats:

123456	Decimal numbers
0x123456	Hexadecimal numbers
0123456	Octal numbers

Override default flash loader path

A default flash loader is selected based on your choice of device on the **General Options>Target** page. To override the default flash loader, select **Override default flash loader path** and specify the path to the flash loader you want to use. A browse button is available for your convenience.

Extra parameters

A flash loader can define its own set of specific options. Use this text box to specify options to control the flash loader.