

IAR C-SPY hardware debugger systems

User Guide

for Freescale's
S08 Microcontroller Family

COPYRIGHT NOTICE

Copyright © 2008 IAR Systems AB.

No part of this document may be reproduced without the prior written consent of IAR Systems AB. The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such a license.

DISCLAIMER

The information in this document is subject to change without notice and does not represent a commitment on any part of IAR Systems. While the information herein is assumed to be accurate, IAR Systems assumes no responsibility for any errors or omissions.

In no event shall IAR Systems, its employees, its contractors, or the authors of this document be liable for special, direct, indirect, or consequential damage, losses, costs, charges, claims, demands, claim for lost profits, fees, or expenses of any nature or kind.

TRADEMARKS

IAR Systems, IAR Embedded Workbench, C-SPY, visualSTATE, From Idea To Target, IAR KickStart Kit, IAR PowerPac, IAR YellowSuite, IAR Advanced Development Kit, IAR, and the IAR Systems logotype are trademarks or registered trademarks owned by IAR Systems AB. J-Link is a trademark licensed to IAR Systems AB.

Microsoft and Windows are registered trademarks of Microsoft Corporation.

FreescTM and the Freescale logo are trademarks of Freescale Semiconductor, Inc.

All other product names are trademarks or registered trademarks of their respective owners.

EDITION NOTICE

First edition: June 2008

Part number: CSS08HW-1

This guide applies to version 1.x of IAR Embedded Workbench® for S08.

Internal reference: IMAE.

Contents

Preface	v
Who should read this guide	v
How to use this guide	v
What this guide contains	vi
Other documentation	vi
Document conventions	vi
Typographic conventions	vi
Naming conventions	vii
Introduction to C-SPY® hardware debugger systems	1
The IAR C-SPY hardware debugger systems	1
Differences between the C-SPY drivers	3
Getting started	3
Running the demo program	3
Debugging using hardware systems	5
C-SPY options for debugging using hardware systems	5
Download options	7
PEmicro BDM Setup options	8
Communication options	9
PEmicro BDM menu	9
Using breakpoints	10
Hardware and software breakpoints	10
Breakpoint Usage dialog box	11
Resolving problems	11
Using flash loaders	13
The flash loader	13
Setting up the flash loader(s)	13
Build considerations	14
The flash loading mechanism	14

Flash Loader Overview dialog box	15
Flash Loader Configuration dialog box	16

Preface

Welcome to the *IAR C-SPY hardware debugger systems User Guide for S08*. The purpose of this guide is to provide you with detailed reference information that can help you use the features of the IAR C-SPY® Hardware Debugger System for S08.

Who should read this guide

You should read this guide if you want to get the most out of the features in the C-SPY hardware debugger systems. In addition, you should have working knowledge of:

- The C programming language
- Application development for embedded systems
- The architecture and instruction set of the S08 microcontroller (refer to the chip manufacturer's documentation)
- The operating system of your host machine.

This guide also assumes that you already have 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 to the IAR C-SPY Debugger, see the tutorials available in the *IAR Embedded Workbench® IDE User Guide*.

How to use this guide

This guide describes the C-SPY interface to the target system you are using; not the general features available in the IAR C-SPY debugger or the hardware target system. To take full advantage of the whole debugger system, you must read this guide in combination with:

- The *IAR Embedded Workbench® IDE User Guide* which describes the general features available in the C-SPY debugger
- The documentation supplied with the target system you are using.

Note that additional features may have been added to the software after the *IAR C-SPY hardware debugger systems User Guide for S08* was printed. The release note `css08e.htm` contains the latest information.

What this guide contains

Below is a brief outline and summary of the chapters in this guide.

- *Introduction to C-SPY® hardware debugger systems* introduces you to the available C-SPY hardware debugger system. The chapter briefly shows the difference in functionality provided by the different C-SPY drivers.
- *Debugging using hardware systems* describes the additional options, menus, and features provided by the debugger system.
- *Using flash loaders* describes the flash loader, what it is and how to use it.

Other documentation

The complete set of IAR development tools for the S08 microcontroller are described in a series of guides. These guides can be found in the `s08\doc` directory or reached from the **Help** menu.

All of these guides are delivered in hypertext PDF or HTML format on the installation media. Some of them are also delivered as printed books.

Recommended web sites:

- The Freescale web site, www.freescale.com, contains information and news about the S08 microcontrollers.
- The IAR Systems web site, www.iar.com, holds application notes and other product information.

Document conventions

When referring to a directory in your product installation, for example `s08\doc`, the full path to the location is assumed, for example `c:\Program Files\IAR Systems\Embedded Workbench 5.0\s08\doc`.

TYPOGRAPHIC CONVENTIONS

This guide uses the following typographic conventions:

Style	Used for
<code>computer</code>	<ul style="list-style-type: none"> • Source code examples and file paths. • Text on the command line. • Binary, hexadecimal, and octal numbers.

Table 1: Typographic conventions used in this guide

Style	Used for
<i>parameter</i>	A placeholder for an actual value used as a parameter, for example <i>filename.h</i> where <i>filename</i> represents the name of the file. Note that this style is also used for <i>s08</i> , <i>configfile</i> , <i>libraryfile</i> , and other labels representing your product, as well as for the numeric part of filename extensions—78.
[option]	An optional part of a command.
{option}	A mandatory part of a command.
a b c	Alternatives in a command.
bold	Names of menus, menu commands, buttons, and dialog boxes that appear on the screen.
<i>italic</i>	<ul style="list-style-type: none"> • A cross-reference within this guide or to another guide. • Emphasis.
...	An ellipsis indicates that the previous item can be repeated an arbitrary number of times.
	Identifies instructions specific to the IAR Embedded Workbench® IDE interface.
	Identifies instructions specific to the command line interface.
	Identifies helpful tips and programming hints.
	Identifies warnings.

Table 1: Typographic conventions used in this guide (Continued)

NAMING CONVENTIONS

The following naming conventions are used for the products and tools from IAR Systems® referred to in this guide:

Brand name	Generic term
IAR Embedded Workbench® for S08	IAR Embedded Workbench®
IAR Embedded Workbench® IDE for S08	the IDE
IAR C-SPY® Debugger for S08	C-SPY, the debugger
IAR C Compiler™ for S08	the compiler
IAR Assembler™ for S08	the assembler
IAR XLINK™ Linker	XLINK, the linker
IAR XAR Library builder™	the library builder
IAR XLIB Librarian™	the librarian

Table 2: Naming conventions used in this guide

Brand name	Generic term
IAR DLIB Library™	the DLIB library

Table 2: Naming conventions used in this guide (Continued)

Introduction to C-SPY® hardware debugger systems

This guide introduces you to the IAR C-SPY hardware debugger system for S08 and to how it differs 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 additional features may have been added to the software after this guide was written. The release notes available in the file `css08e.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 S08 microcontroller supports evaluation boards that have a standardized 6-pin BDM (background debug mode) connector. Background debug mode is used for system development, in-circuit testing, field testing, and programming.

The IAR C-SPY Debugger for the S08 microcontroller is available with a driver for the following hardware debug interfaces:

- USB Multilink (P&E Microcomputer Systems)
- Cyclone PRO (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 serial, Ethernet, or USB connection.

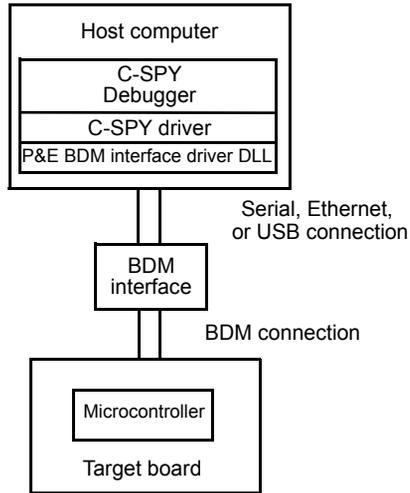


Figure 1: C-SPY BDM debugger communication overview

DIFFERENCES BETWEEN THE C-SPY DRIVERS

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

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

Table 1: Differences between available C-SPY drivers

* Cycle counter statistics are not available.

† Profiling works provided that enough breakpoints are available. One breakpoint per function must be set, which in reality limits profiling availability to applications where all code is located in RAM. Moreover, enabling profiling in the BDM debugger results in decreased performance.

Getting started

In this example, a demo project is set up for the Freescale DEMOQE128 development system, where the LEDs are connected to two general purpose I/O ports. When you run the demo program, the LEDs will blink.

You can find a ready-made workspace, `DemoQE.eww`, together with source files in the `s08\src\examples\QE` directory.

The workspace contains the LEDs project with the `Leds.c` file, which sends out data to `PORT C`.

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.

- I 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 **DemoQE**.

- 2 In the workspace window, select the **LEDs** project. Choose **Project>Options**. Verify the following critical options:

Category	Page	Option/Setting
General Options	Target	Device: MC9S08QE128 ¹⁾
Debugger	Setup	Driver: PEmicro BDM
PEmicro BDM	Communication	Communication type: 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 11.

- 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 system. 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
- PEmicro BDM 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:
 - **PEmicro BDM** for any of the interfaces from P&E.

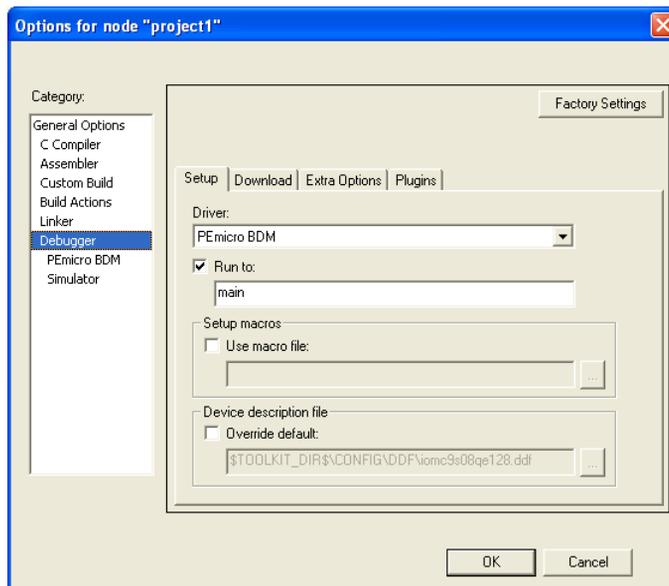


Figure 2: Project options dialog box

For information about the settings **Run to**, **Setup macros**, and **Device description file**, and 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.
- 6 To set options for using a flash loader with 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.

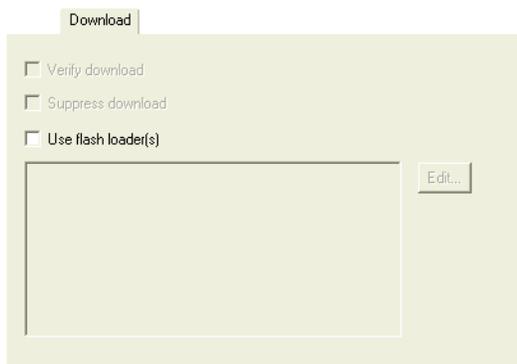


Figure 3: Download page

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. 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 13.

PEMICRO BDM SETUP OPTIONS

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

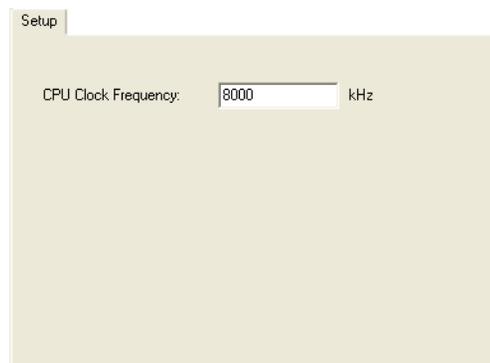


Figure 4: PEmicro BDM setup page

CPU Clock frequency

The S08 `FCDIV` register is used to control the length of timed events in program and erase algorithms executed by the flash memory controller. The flash loader configures the `FCDIV` register based on the bus and CPU clock frequencies, where the bus clock frequency is one half of the CPU clock frequency. The CPU clock frequency must be within the range 300 kHz to 204,800 kHz (204.8 MHz) and is by default set to 8000 kHz (8 MHz).

If you are not using a flash loader with the debugger, the CPU clock frequency value will be disregarded.

COMMUNICATION OPTIONS

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

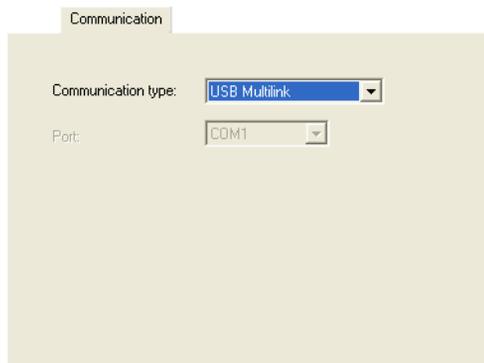


Figure 5: Communication page

The following settings are available:

Setting	Description
Communication type	Selects one of the supported communication types: <ul style="list-style-type: none"> • USB Multilink • Cyclone PRO Serial • Cyclone PRO USB • Cyclone PRO Ethernet.
Port	Selects one of the supported ports if Cyclone Pro Serial is selected. Choose between COM1–COM4.

Table 3: Communication options

PEmicro BDM menu

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



Figure 6: PEmicro BDM 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* or *software*.

This table summarizes the characteristics of hardware and software breakpoints:

Type	Number	Execution overhead
Hardware	Three	No
Software (statement located in RAM memory)	Unlimited	Yes

Table 4: Hardware and software breakpoints

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 RAM 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 three.

Software breakpoints

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

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

BREAKPOINT USAGE DIALOG BOX

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

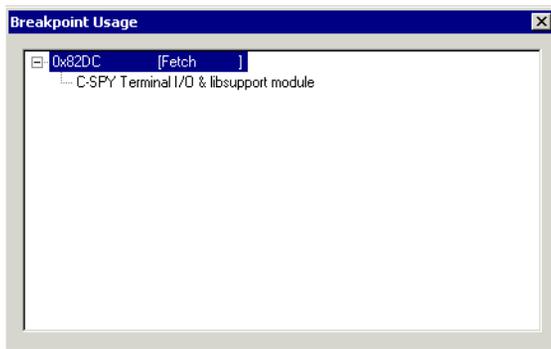


Figure 7: Breakpoint Usage dialog box

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 might 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.

- Make sure you have installed the P&E USB Multilink and Cyclone PRO communication drivers in the `s08\drivers\PEmicro` directory 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 range.

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 15. You can either select a flash loader or click **New** or **Edit** to 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 16.

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. The macro is useful for disabling flash array protection.

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 when debugging while the flash loader is running as an application, the setup macro `execUserPreload` must be used instead of `execUserFlashInit`.

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 (d78) that contains the entire user application and 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. All supplied linker command files are configured to automatically generate a simple-code output file, in addition to any other output files that you specify.

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 executes the setup function `execUserFlashInit` from the optional macro file.
- 2 C-SPY downloads the flash loader into target RAM.
- 3 C-SPY starts execution of the flash loader.
- 4 The flash loader opens the simple-code file containing the application code and programs it into flash memory.
- 5 The flash loader terminates.
- 6 C-SPY opens the UBROF file containing the application as well as debug and symbol information. Any parts of the application linked to RAM will be written to RAM by C-SPY. Any parts of the application linked to flash memory will be ignored, unless the **Verify download** option is selected.
- 7 C-SPY switches context to the user application.

The steps 1 to 6 are performed for each selected flash loader.

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.

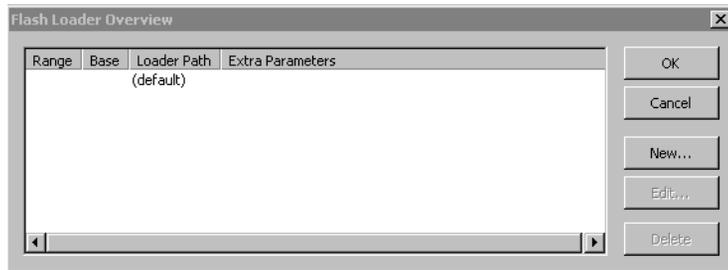


Figure 8: Flash Loader Overview dialog box

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 <i>Flash Loader Configuration dialog box</i> , page 16.
Edit	Opens the Flash Loader Configuration dialog box where you can modify the settings for the selected flash loader; see <i>Flash Loader Configuration dialog box</i> , page 16.
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.

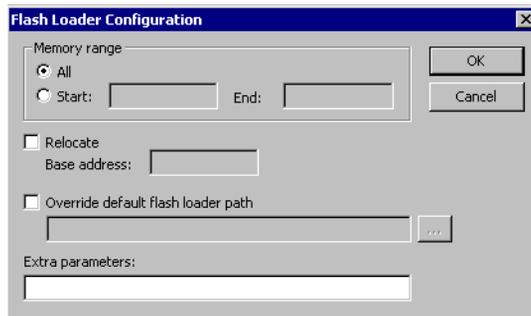


Figure 9: Flash Loader Configuration dialog box

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 base address

It is not possible to relocate the default flash base address in the S08 microcontroller. Therefore, any attempts to relocate the flash base address using this dialog box will have no effect.

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.

